

**Smart Object, not Smart Environment:
*Cooperative Augmentation of Smart Objects
Using Projector-Camera Systems***

**David Molyneaux
April 2008**

Lancaster University

Submitted in part fulfilment of the requirement for the degree of
Doctor of Philosophy in Computer Science

Abstract

Smart objects research explores embedding sensing and computing into everyday objects - augmenting objects to be a source of information on their identity, state, and context in the physical world. A major challenge for the design of smart objects is to preserve their original appearance, purpose and function. Consequently, many research projects have focussed on adding input capabilities to objects, while neglecting the requirement for an output capability which would provide a balanced interface.

This thesis presents a new approach to add output capability by smart objects cooperating with projector-camera systems. The concept of Cooperative Augmentation enables the knowledge required for visual detection, tracking and projection on smart objects to be embedded within the object itself. This allows projector-camera systems to provide generic display services, enabling spontaneous use by any smart object to achieve non-invasive and interactive projected displays on their surfaces. Smart objects cooperate to achieve this by describing their appearance directly to the projector-camera systems and use embedded sensing to constrain the visual detection process.

We investigate natural appearance vision-based detection methods and perform an experimental study specifically analysing the increase in detection performance achieved with movement sensing in the target object. We find that detection performance significantly increases with sensing, indicating the combination of different sensing modalities is important, and that different objects require different appearance representations and detection methods.

These studies inform the design and implementation of a system architecture which serves as the basis for three applications demonstrating the aspects of visual detection, integration of sensing, projection, interaction with displays and knowledge updating.

The displays achieved with Cooperative Augmentation allow any smart object to deliver visual feedback to users from implicit and explicit interaction with information represented or sensed by the physical object, supporting objects as both input and output medium simultaneously. This contributes to the central vision of Ubiquitous Computing by enabling users to address tasks in physical space with direct manipulation and have feedback on the objects themselves, where it belongs in the real world.

Preface

This dissertation has not been submitted in support of an application for another degree at this or any other university. It is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated.

Excerpts of this thesis have been published in conference and workshop manuscripts, most notably:

[Molyneaux and Gellersen 2006]

[Molyneaux, Gellersen et al. 2007]

[Molyneaux, Gellersen et al. 2008]

Acknowledgments

This work would not have been possible without the support and facilities offered by my supervisors Prof. Hans Gellersen and Dr. Gerd Kortuem. I thank them both for their never-ending encouragement, inspiration and guidance throughout my time as a PhD student. I consider myself lucky to have been given the opportunity to work in such an interesting field and honoured to have been part of the exceptional research group they have created at Lancaster. The unique lab environment with the large international crowd of people makes it a special place, and one that I will cherish forever.

I wish to thank all those other colleagues in the Ubicomp group at Lancaster who, either through direct assistance, or in discussions have helped this work in some way:

Henoc Agbota, Mohammed Alloulah, Bashar Al Takrouri, Urs Bischoff, Florian Block, Carl Fischer, Roswitha Gostner, Andrew Greaves, Yukang Guo, Robert Hardy, Mike Hazas, Henrik Jernström, Serko Katsikian, Chris Kray, Kristof van Laerhoeven, Rene Mayrhofer, Matthew Oppenheim, Faizul Abdul Ridzab, Enrico Rukzio, Dominik Schmidt, Jennifer Sheridan, Martin Strohbach, Vasughi Sundramoorthy, Nic Villar, Jamie Ward and Martyn Welsh.

Plus all the visitors and other alumni: Aras Bilgen, Martin Berchtold, Andreas Bulling, Clara Fernández de Castro, Manuel García-Herranz del Olmo (especially for all the boiled eggs), Alina Hang, Pablo Haya, Paul Holleis, Matthew Jervis, Russell Johnson, Yasue Kishino, Matthias Kranz, Masood Masoodian, Christoph März, Michael Müller, Kavitha Muthukrishnan, Albrecht Schmidt, Sara Streng, Tsutomu Terada and all those other people I have encountered at Lancaster during my time here.

Special thanks go to some of the most important people at Lancaster - the secretaries and support staff who have helped me during my time here Aimee, Ben, Cath, Chris, Gillian, Helen, Ian, Jess, Liz, Sarah, Steve and Trish. Also thanks to Gordon for being a superb head of department and to CAKES for interesting presentations and away-days.

Others outside of Lancaster have also helped greatly with this work, most notably Bernt Schiele in Darmstadt – without his support and guidance much of this work would not have been possible. I am greatly indebted to him. Thanks also to everyone from Multimodal Interactive Systems in Darmstadt for the great visits: Krystian Mikolajczyk, Bastion Leibe, Gyuri Dorko, Edgar Seemann, Mario Fritz, Andreas Zinnen, Nicky Kern, Tam Huynh, Ulrich Steinhoff, Niko Majer and Ursula Paeckel.

The FLUIDUM project and specifically Andreas Butz and Mira Spassova deserve a big mention, as that is where it all started. Without your initial support, Java code and spark of enthusiasm for the steerable projector work I would never have travelled as far as I did during my PhD journey. Thanks also to those whose code was incorporated in some form in the demos, most notably Mark Pupilli from Bristol for the basic Particle Filter and Rob Hess for the SIFT implementation I modified for my work.

Thanks to both my examiners – Mike Hazas and Andrew Calway from the University of Bristol for the interesting discussions and insights during the viva that have enhanced this thesis greatly beyond its initial submission.

Most importantly, I thank my family – Mum, Dad, Jane and grandparents - for all the love and support (in every sense of the word) they have given me over the years. Rose also deserves a special mention. Without her I would still be stuck in my second year with no demo or paper. She has inspired me, kept me on track, put up with my grumpy mornings and read probably more about steerable projectors and Cooperative

Augmentation in the last two years than she would otherwise want to... Thank you for being there for me Rose, I hope I can do the same for you.

Finally, I would like to thank the projects and organisations that funded this work, namely the EPSRC, the Ministry of Economic Affairs of the Netherlands through the BSIK project Smart Surroundings under contract no. 03060 and Lancaster University through the e-Campus grant.

David Molyneaux

Lancaster, September 2008

Contents

CHAPTER 1	INTRODUCTION	1
1.1	SMART OBJECT OUTPUT	1
1.2	COOPERATIVE AUGMENTATION	2
1.3	CHALLENGES	3
1.4	CONTRIBUTIONS	4
1.5	THESIS STRUCTURE.....	5
CHAPTER 2	RELATED WORK.....	6
2.1	INTRODUCTION	6
2.2	UBIQUITOUS COMPUTING	8
2.2.1	<i>Sensor Nodes</i>	9
2.2.2	<i>Smart Objects</i>	10
2.2.3	<i>Tangible User Interfaces</i>	12
2.2.4	<i>Input-Output Imbalance</i>	14
2.3	PROJECTOR-BASED AUGMENTED REALITY	14
2.3.1	<i>Projector-Camera Systems</i>	16
2.3.2	<i>Mobile, Handheld and Wearable Projector-Camera Systems</i>	17
2.3.3	<i>Multi-Projector Display Systems</i>	18
2.3.4	<i>Steerable Projector-Camera Systems</i>	19
2.3.5	<i>Interaction with Projector-Camera System Displays</i>	22
2.3.6	<i>Projected Display Geometrical Calibration</i>	26
2.3.7	<i>Projection Photometric and Colorimetric Calibration</i>	31
2.3.8	<i>Issues with Projector-based Augmented Reality</i>	33
2.3.9	<i>Mobile Objects</i>	36
2.4	DETECTION AND TRACKING OF OBJECTS	37
2.5	FIDUCIAL MARKER DETECTION AND TRACKING.....	37
2.6	NATURAL APPEARANCE DETECTION AND TRACKING	40
2.6.1	<i>Tracking</i>	40
2.6.2	<i>Detection</i>	42
2.6.3	<i>Multi-Cue Detection and Tracking</i>	45
2.6.4	<i>Camera Model</i>	46
2.6.5	<i>Pose calculation</i>	48
2.6.6	<i>Hybrid Detection</i>	50
2.7	VISION-BASED DETECTION WITH PHYSICAL SENSING	50
2.7.1	<i>Sensing for Pose and Object Motion Prediction</i>	51
2.7.2	<i>Structured Light Sensing for Location and Pose</i>	51
2.8	SUMMARY	53
CHAPTER 3	COOPERATIVE AUGMENTATION CONCEPTUAL FRAMEWORK	54
3.1	COOPERATIVE AUGMENTATION ENVIRONMENT	55
3.2	THE OBJECT MODEL	55
3.3	THE PROJECTOR-CAMERA SYSTEM MODEL	56

3.4	COOPERATIVE AUGMENTATION PROCESS	57
3.4.1	<i>Detection</i>	59
3.4.2	<i>Projection</i>	60
3.5	CONCLUSION	61
CHAPTER 4 VISION-BASED OBJECT DETECTION		62
4.1	NATURAL APPEARANCE DETECTION	62
4.2	OBJECT DETECTION METHODS	63
4.3	EVALUATION DATASET	64
4.3.1	<i>Object Appearance Library</i>	64
4.4	SCALE AND ROTATION EXPERIMENTS	65
4.4.1	<i>Design</i>	66
4.4.2	<i>Procedure</i>	67
4.4.3	<i>Apparatus</i>	69
4.4.4	<i>Results</i>	69
4.5	DISCUSSION	74
4.6	CONCLUSION	76
CHAPTER 5 COOPERATIVE DETECTION		77
5.1	VIDEO TEST LIBRARY	77
5.2	COOPERATIVE DETECTION EXPERIMENTS	78
5.2.1	<i>Design</i>	78
5.2.2	<i>Training</i>	79
5.2.3	<i>Procedure</i>	79
5.2.4	<i>Results</i>	80
5.3	DISCUSSION	85
5.3.1	<i>Single Cue Detection</i>	86
5.3.2	<i>Multi-Cue Combination</i>	88
5.3.3	<i>Limitation of Movement Sensing</i>	89
5.3.4	<i>Use of Sensing in smart objects for Pose Calculation</i>	89
5.4	CONCLUSION	90
CHAPTER 6 SYSTEM ARCHITECTURE		92
6.1	ARCHITECTURE DESIGN OVERVIEW	92
6.1.1	<i>Architecture Novelty</i>	94
6.2	ARCHITECTURE COMPONENTS	95
6.2.1	<i>Detection and Tracking</i>	95
6.2.2	<i>Projection</i>	98
6.2.3	<i>Pan & Tilt Tracking Component</i>	99
6.2.4	<i>Smart Objects</i>	102
6.2.5	<i>Object Proxy</i>	105
6.2.6	<i>Database Server</i>	105
6.2.7	<i>Knowledge Updating</i>	107
6.3	IMPLEMENTATION	108
6.3.1	<i>Detection and Tracking</i>	109
6.3.2	<i>Projection</i>	114
6.3.3	<i>Pan and Tilt Control</i>	117
6.3.4	<i>Object Proxy</i>	118
6.3.5	<i>Database Server</i>	119
6.3.6	<i>Networking Protocol Implementation</i>	119
6.4	PROJECTOR-CAMERA SYSTEM CALIBRATION	122
6.5	DISCUSSION	124
6.6	DETECTION METHOD MEMORY EXPERIMENTS	125
6.6.1	<i>Design</i>	126
6.6.2	<i>Results</i>	126
6.6.3	<i>Discussion</i>	127
6.7	POSE CALCULATION, POSE JITTER AND PROJECTION ACCURACY EXPERIMENTS	128
6.7.1	<i>Design</i>	128
6.7.2	<i>Procedure</i>	129
6.7.3	<i>Apparatus</i>	130

6.7.4	<i>Results</i>	130
6.7.5	<i>Discussion</i>	133
6.8	CONCLUSION.....	134
CHAPTER 7 DEMONSTRATION APPLICATIONS		136
7.1	SMART COOPERATIVE CHEMICAL CONTAINERS	136
7.1.1	<i>Object Model</i>	136
7.1.2	<i>Registration</i>	137
7.1.3	<i>Detection</i>	137
7.1.4	<i>Projection</i>	138
7.1.5	<i>Manipulating the Object</i>	138
7.1.6	<i>Knowledge Updating</i>	140
7.1.7	<i>Objects Departing the Environment</i>	140
7.1.8	<i>Discussion</i>	141
7.2	SMART PHOTOGRAPH ALBUM.....	142
7.2.1	<i>Object Model</i>	143
7.2.2	<i>Registration and Detection</i>	144
7.2.3	<i>Projection and Manipulation</i>	144
7.2.4	<i>Discussion</i>	144
7.3	SMART COOKING	148
7.3.1	<i>Hard-Boiled Egg Recipe</i>	148
7.3.2	<i>Object Model</i>	149
7.3.3	<i>Task Procedure</i>	149
7.3.4	<i>Discussion</i>	152
7.4	CONCLUSION.....	155
CHAPTER 8 CONCLUSION		156
8.1	CONTRIBUTIONS AND CONCLUSIONS	156
8.2	BENEFITS OF OUR APPROACH	158
8.3	LIMITATIONS.....	159
8.4	FUTURE WORK.....	160
BIBLIOGRAPHY.....		162
APPENDIX A STEERABLE PROJECTOR-CAMERA SYSTEM CONSTRUCTION		176
A.1	STEERABLE PROJECTOR-CAMERA SYSTEM DESIGN	176
A.2	STEERING MECHANISM.....	176
A.2.1	<i>Steering Mechanism Characteristics</i>	177
A.2.2	<i>Moving Mirror and Moving Head comparison</i>	180
A.3	VIDEO PROJECTOR	181
A.4	CAMERA	184
A.5	COMMERCIAL STEERABLE PROJECTORS	187
A.6	CONSTRUCTION OF STEERABLE PROJECTOR SYSTEMS.....	188
A.7	STEERABLE PROJECTOR HARDWARE	189
A.7.1	<i>Projector Selection</i>	189
A.7.2	<i>Steering Mechanism Selection</i>	190
A.7.3	<i>Camera Selection</i>	190
A.8	MOVING HEAD STEERABLE PROJECTOR CONSTRUCTION.....	191
A.8.1	<i>Moving Head Control</i>	193
A.9	MOVING MIRROR STEERABLE PROJECTOR CONSTRUCTION.....	194
A.10	CHARACTERISATION OF MOVING HEAD MECHANISM.....	195
A.10.1	<i>Design</i>	195
A.10.2	<i>Procedure</i>	196
A.10.3	<i>Results</i>	197
A.10.4	<i>Discussion</i>	197
A.11	CHARACTERISATION OF PROJECTOR	198
A.12	STEERABLE PROJECTOR SYSTEM COMPARISON	200
A.13	CONCLUSION.....	202
APPENDIX B SMART OBJECT PROGRAMMING EXAMPLES.....		204
B.1	ROUGH HANDLING DETECTION	204

B.2	SMART BOOK.....	205
B.3	SMART FURNITURE ASSEMBLY.....	206
B.4	SMART COOKING OBJECT MODEL	208

List of Figures

Figure 1.1 Cooperative Augmentation of smart objects with Projector-Camera Systems	3
Figure 2.1 This work draws from 4 main areas of computing: Ubiquitous Computing, Augmented Reality, Computer Vision and Tangible User Interfaces (TUI)	6
Figure 2.2 The "Weiser Continuum" of ubiquitous computing [Weiser 1996] to monolithic computing, taken from [Newman, Bornik et al. 2007]	6
Figure 2.3 The Milgram and Kishino "Virtuality" Continuum of Reality to Virtual Reality [Milgram and Kishino 1994], taken from [Newman, Bornik et al. 2007]	7
Figure 2.4 The Milgram-Weiser Continuum [Newman, Bornik et al. 2007]	8
Figure 2.5 (left) Smart-Its design (centre) Particle Smart-Its Device main board, (right) Add-on sensor boards [Decker, Krohn et al. 2005]	9
Figure 2.6 (left) Crossbow Technology Motes (right) Motive Corporation (now Sentilla Corporation) Telos Mote [Inc 2007]	10
Figure 2.7 (left) Mediacup Object [Gellersen, Beigl et al. 1999], (centre) Cooperative Chemical Containers, (right) Hazard warning display using 3 LEDs [Strohbach, Gellersen et al. 2004]	11
Figure 2.8 (left) Proactive Furniture Assembly [Antifakos, Michahelles et al. 2004], (centre) Display Cube Object [Terrenghi, Kranz et al. 2006], (right) Tuister Object [Butz, Groß et al. 2004]	11
Figure 2.9 Using a sensor enabled Cubicle for interaction [Block, Schmidt et al. 2004]	13
Figure 2.10 (left) Prototype mock-up display cube using static front-projection, (centre) The current LED-matrix display Z-agon interactive cube, (right) The envisioned fully interactive cube with colour displays [Matsumoto, Horiguchi et al. 2006]	13
Figure 2.11 (left) A chessboard with memory (centre) Interactive optics-design, (right) A prototype I/O bulb projector-camera system in the Luminous Room [Underkoffler, Ullmer et al. 1999]	15
Figure 2.12 Spatially Augmented Reality (SAR) [Raskar, Welch et al. 1999]	16
Figure 2.13 The Projector-Camera system family decomposed by mobility and display size, compared to traditional desktop monitors	17
Figure 2.14 (left and centre) The iLamps Project developed a handheld projector-camera system, (right) Detecting circular fiducial markers to augment a wall scene with projection	18

Figure 2.15 SONY's Handheld spotlight projector [Rapp, Michelitsch et al. 2004], (centre left) Multi-user interaction using handheld projectors [Cao, Forlines et al. 2007], (right) Wearable Projector-camera system [Karitsuka and Sato 2003].....	18
Figure 2.16 (left) The PixelFlex reconfigurable multi-projector display [Yang, Gotz et al. 2001], (right) Ad-hoc projector clustering and geometric correction for seamless display with iLamps [Raskar, VanBaar et al. 2005]	19
Figure 2.17 IBM's Everywhere Display demonstration at SIGGRAPH [Pinhanez 2001]	19
Figure 2.18 (left) Messaging notification, (centre) dynamic navigation signage, (right) augmented filing cabinet object [Pinhanez 2001].....	20
Figure 2.19 (left and centre) Portable Projected Display Screen [Borkowski 2006], (right) Personal Interaction Panel [Ehnes, Hirota et al. 2004].....	20
Figure 2.20 IBM's Steerable Interface EDML Framework.....	21
Figure 2.21 The IBM Steerable Interfaces Characterisation [Pingali, Pinhanez et al. 2003]	22
Figure 2.22 Interaction Techniques for Projected Displays	22
Figure 2.23 Fingertip matching in (a) Camera image, (b) Motion mask, (c) Finger tip template match [Pinhanez, Kjeldsen et al. 2001].....	25
Figure 2.24 (far left) Polar Motion Map, (left) the corresponding segments unrolled to a rectangle, (right) PMM in use [Kjeldsen 2005].....	25
Figure 2.25 (left) Using a red paper slider widget [Kjeldsen 2005], (right) Increasing camera exposure to compensate for projector light [Letessier and Bérard 2004] ..	26
Figure 2.26 (left) On-axis projection gives a rectangular image, (right) Geometric distortion due to horizontal projector rotation when projecting onto a planar screen	26
Figure 2.27 Classification of projector-camera system geometrical calibration methods, based on [Borkowski, Riff et al. 2003]	27
Figure 2.28 Model-based distortion correction based on known surface geometry and pose relative to the projector [Pinhanez 2001]	28
Figure 2.29 Projecting dots onto a planar screen to recover the projector-camera homography	28
Figure 2.30 (left) Creating the projector-camera homography, (right) Using the combined projector-screen transformation to project an undistorted image [Sukthankar, Stockton et al. 2001].....	29
Figure 2.31 (left) Gray-code structured light patterns (centre) Geometrically corrected multi-projector display on curved display [Raskar, VanBaar et al. 2005], (right) Automatic Projector Display Surface Estimation Using Every-Day Imagery [Yang and Welch 2001]	30
Figure 2.32 (left) Colour correction to project on any surface [Bimber, Emmerling et al. 2005], (right) Real-time adaptation for mobile objects [Fujii, Grossberg et al. 2005]	32
Figure 2.33 (left and centre) Uneven illumination on a non-planar surface, (right) uncompensated and intensity compensated image [Park, Lee et al. 2006].....	33
Figure 2.34 VRP used to overcome occluding light shadows [Summet, Flagg et al. 2007]	34

Figure 2.35 (left) Interactive buttons projected on a variety of everyday surfaces, (right) Dynamic Shader Lamps projecting on mobile tracked object [Bandyopadhyay, Raskar et al. 2001].....	35
Figure 2.36 (left) Examples of different Fiducial Marker systems [Fiala 2005], (right) Using ARToolkit Markers with a HMD to view a virtual character [Kato and Billinghurst 1999]	38
Figure 2.37 ARToolkit Recognition and Overlay method [Kato and Billinghurst 1999]	38
Figure 2.38 (left and centre) Occlusion of ARToolkit marker prevents tracking, (right) Occlusion of ARTag marker [Fiala 2005].....	39
Figure 2.39 (left) AR Toolkit fails to detect markers under different illumination conditions in the same frame [Fiala 2005], (right) Ehnes et al. track ARToolkit markers with a steerable projector and project on modelled surfaces in the marker area [Ehnes, Hirota et al. 2004].....	40
Figure 2.40 (left) Drummond and Cipolla's RAPiD-like algorithm, (right) robustness to partial occlusion [Drummond and Cipolla 2002].....	41
Figure 2.41 (left) Two dissimilar objects and their Mag-Lap histograms corresponding to a particular viewpoint, image plane rotation and scale. [Schiele and Crowley 2000], (right) Shape Context 2D log-polar histogram based on relative point locations [Belongie, Malik et al. 2002]	43
Figure 2.42 (left) SIFT local feature based AR method, (right, top) SIFT Features detected on a mug, (right, bottom) AR teapot added to camera display [Gordon and Lowe 2004]	45
Figure 2.43 (left) Pinhole Perspective Camera Model, (right) Checkerboard pattern for camera calibration with overlaid pattern coordinate system.....	47
Figure 2.44 (left) Projection on mobile planar surfaces with single light sensor [Summet and Sukthankar 2005], (right) Projection onto surfaces with sensors at each corner for rotation information [Lee, Hudson et al. 2005].....	52
Figure 3.1 Detection Sequence Diagram.....	58
Figure 3.2 Projection Sequence Diagram.....	58
Figure 3.3 Knowledge flow in the detection process	59
Figure 4.1 Experiment Objects (left to right, top to bottom): a football, a chemical container barrel, a book, a product box, a smart cube, a chair, a cup, a notepad, a cereal box and a toaster.	64
Figure 4.2 (left) Box Object Scale images at 1m, 3m, 6m from camera, (right) Notepad object rotation images at -40° , 0° , $+40^\circ$	64
Figure 4.3 Different numbers of corner features (yellow dots) are detected and in different locations on the Cereal box object at 1m, 3m and 6m distance with the single-scale Harris algorithm [Harris and Stephens 1988].....	65
Figure 4.4 Camera and Object Detection Coordinate System Transformations	65
Figure 4.5 (left) Number of features detected on Mediacube object by Harris-based algorithms, (right) Harris-based algorithm detector repeatability when trained at 4m.....	70
Figure 4.6(left) Harris detector with multiple training images 1.1m,1.85m,3m,5m, (right) Mean repeatability with varying training distance for Harris-based detector algorithms.....	71

Figure 4.7 (left) SIFT Descriptor matching percentages with 2D rotation, (right) Number of features detected with 3D rotation.....	71
Figure 4.8 (left) Detector repeatability for 3D rotation, (right) Descriptor matching percentages for 3D rotation.....	72
Figure 4.9 (left) Detection performance over all objects for SIFT local features at 1m, 2m and 6m training distances, (right) Detection performance over all objects for Shape Context at 1m, 3m and 6m training distances.....	73
Figure 4.10 (left) Detection performance over all objects for Texture (Mag-Lap) Multidimensional Histograms at 1m, 2m and 6m training distances, (right) Detection performance over all objects for LAB Colour Histograms at 1m, 3m and 6m training distances.	73
Figure 4.11 (left) Detection percentage repeatability of SIFT (DoG) over all objects when varying 3D rotation angle, (right) Detection performance over all objects when varying 3D rotation angle, for Texture, Colour and Shape, at 3m training and test distances.	74
Figure 5.1 Video Test library images of chemical container (top) and cereal box (bottom).	77
Figure 5.2 Graph of detection algorithm results without sensing (orange) and with movement sensing (blue), averaged over all objects. Error bars show 95% Confidence level (CI) of mean.....	80
Figure 5.3 (left) SIFT Local Feature Detection Performance with and without sensing for each object in test library, (right) Mag-Lap Texture Detection Performance with and without sensing for each object in test library	82
Figure 5.4 (left) Lab Colour Detection Performance with and without sensing for each object in test library, (right) Shape Context Detection Performance with and without sensing for each object in test library	82
Figure 5.5 Detection Performance improvement using multiple cues, No Sensing.....	85
Figure 5.6 Detection Performance improvement using multiple cues and Movement Sensing.....	85
Figure 6.1 Distributed System Architecture Overview	93
Figure 6.2 Detection method selection based on smart object knowledge.....	96
Figure 6.3 The 4 coordinate systems: Camera, Projector and smart object Local Coordinate Systems, and the arbitrary World Coordinate System.....	97
Figure 6.4 Search Methods: (left) Creeping Line Search over the whole Pan and Tilt Field Of View, (right) Expanding-Box search from previous object location	100
Figure 6.5 Two views of a Lab Environment: (left) South-West Elevation, (right) North-East Elevation	108
Figure 6.6 The moving-head steerable projector-camera system.....	109
Figure 6.7 View down into a background model re-projected into a hemisphere, captured by rotating a moving head steerable projector through a 360x90° FOV.	111
Figure 6.8 (left) Image of detected of Book object with overlaid 3D model and object coordinate system axes, (right) Four detected objects tracked simultaneously - the Chemical Container, the Book , the Card and the Cup (green lines denote the maximum extent of the 3D model).....	113

Figure 6.9 (left) 3 surface projection on box object, (right) Sensed temperature projected on the non-planar smart mug surface - the blue wire at the right is the antenna of the Smart-Its device.....	116
Figure 6.10 Architecture Message Protocol and Routing.	120
Figure 6.11 (left) Composite image of ARToolkit marker aligned with 5 far calibration locations and handheld for one of the near calibration locations, (right) Projector Calibration using projected pattern on planar surface [Park, Lee et al. 2006].....	124
Figure 6.12 The location accuracy and jitter experiment object grid locations, orthogonal to the camera. Object test locations are the green circles.	129
Figure 6.13 Error in Z-axis location for objects at a large distance from the camera causes smaller error in the X,Y location error of the projection.....	134
Figure 7.1 (left) New objects arrives in environment, (centre) An employee walks with containers, (right) The employee places one object on the floor.....	137
Figure 7.2 (left) Detected container with green wireframe 3D Model superimposed on the camera view using calculated pose, (right) A warning message projection on two chemical containers.....	138
Figure 7.3 Partial Object Model representation of Chemical Container Demonstrator	139
Figure 7.4 (left) Scale and rotation invariant local features detected on chemical containers, (right) A container leaves the environment with the employee.....	140
Figure 7.5 Photograph Album smart object with Projection (left) front cover, button state 10 (centre) being opened, (right) inside, button state 5.....	143
Figure 7.6 Partial Object Model representation of Photo Album Demonstrator.....	146
Figure 7.7 Photograph Album Content to Project.....	147
Figure 7.8 (left) Recipe State 0, (right) Add Egg Projection inside pan.....	150
Figure 7.9 Recipe State 8.....	151
Figure 7.10 (left) Example salt 3D model, (right) 3D model with added base projection area.....	153
Figure A.1 (left) Moving mirror display light, (right) Moving head display light [SteinigkeShowtechnikGmbH 2005].....	177
Figure A.2 (left) The IBM Everywhere Display Steerable Projector Design, (right) Everywhere Display Projection Cone [Pinhanez 2001].....	178
Figure A.3 (left) Single chip DLP projector optics [TexasInstruments 2005], (right) Mitsubishi Pocket Projector [MitsubishiElectricCorporation 2008].....	183
Figure A.4 (left) A Co-axial projector-camera system [Fujii, Grossberg et al. 2005] (centre) Camera Resolution Test Chart [Geoffrion 2005], (right) Bayer Pattern Colour Filter Array [Geoffrion 2005].....	185
Figure A.5 (left) Futurelight PHS150 Single arm Yoke, (right) Compulite “Luna” dual fork Yoke.....	190
Figure A.6 The pan and tilt yoke uncovered.....	191
Figure A.7 Steerable Projector Cabling Layout.....	192
Figure A.8 (left) The fabricated projector bracket attached to the yoke, (centre) FLUIDUM room projector mounting direct to concrete [Butz, Schneider et al. 2004], (right) Wooden Bracing of the four threaded rods from which the projector is hung.....	193

Figure A.9 (left) The operational moving head steerable projector system, (centre) The tripod-mounted moving mirror steerable projector system, (right) Close-up of operational moving mirror yoke and pan-tilt camera	194
Figure A.10 Moving Mirror Steerable Projector Cabling.....	195
Figure A.11 (left) Image Size versus Distance at Zoom 0, (right) Image Size versus Distance at Zoom 20 (mid-zoom).....	199
Figure A.12 (left) Display resolution versus distance to projection surface at Zoom 0, (right) Display resolution versus distance to projection surface at Zoom 20	199
Figure A.13 (left) Focus steps versus distance to display surface at Zoom 0, (right) Focus steps versus distance to display surface at Zoom 20	200
Figure B.1 State Machine Program for Detecting Rough Handling of a smart object.	205
Figure B.2 State Machine Program for Articulated Smart Book with Force Sensors on Each Page.....	206
Figure B.3 Smart Furniture Assembly Program for 3 Individual Pieces.....	207
Figure B.4 Hard-Boiled Egg Recipe States 0-2	208
Figure B.5 Hard-Boiled Egg Recipe States 3-5	209
Figure B.6 Hard-Boiled Egg Recipe States 6-8	210
Figure B.7 Hard-Boiled Egg Recipe States 8-10	211
Figure B.8 Hard-Boiled Egg Recipe States 10-12.....	212

List of Tables

Table 5.1 Mean algorithm runtime per frame, averaged over 200 frames.....	81
Table 5.2 Mean time to detection from first entry to the environment.....	81
Table 5.3 Ranking of Cues by Detection Results for Each Object, No Movement Sensing.....	84
Table 5.4 Ranking of Cues by Detection Results for Each Object, with Movement Sensing.....	84
Table 5.5 Guidance for which method to use, based on object appearance type.....	87
Table 6.1 Appearance knowledge levels and detection methods with associated processing cost.....	95
Table 6.2 Geometric correction methods for projection based on object geometry [Bimber and Raskar 2005].....	98
Table 6.3 Memory requirements for Appearance Knowledge and the total Object Model with single viewpoint (minimum) and full viewing sphere (maximum).....	127
Table 6.4 Median Rotation Error results for each object, over -70° to $+70^{\circ}$ Out-of-plane and 10° to 350° in-plane.....	131
Table 6.5 Median Location Calculation Error in X,Y,Z Camera Coordinate System for each object, averaged over all grid locations.....	131
Table 6.6 Median Projection Location Error on the Object X,Y front surface plane for each object, averaged over all grid locations.....	131
Table 6.7 Median 3D Object Location Jitter from Median Location for each object, averaged over all grid locations.....	132
Table 6.8 Median Projection Location Jitter from Median Location on the Object front surface plane for each object, averaged over all grid locations.....	132
Table A.1 Four commercial steerable projectors.....	187
Table A.2 Projector Specifications.....	189
Table A.3 Yoke Selection Specifications.....	190
Table A.4 Moving Head Steerable Projector Characterisation.....	197
Table A.5 Steerable Projector Steering Mechanism Comparison.....	201
Table A.6 Steerable Projector Video Projector Comparison.....	201
Table A.7 Steerable Projector Camera Comparison.....	202

Chapter 1 Introduction

One of the central visions of Ubiquitous Computing is that the environment itself becomes the user interface [Ishii and Ullmer 1997]. Interaction will be significantly different than with traditional computing – physical objects, surfaces and spaces themselves will allow us to perform our tasks, while the technology itself becomes transparent, disappearing into the background [DisappearingComputer 2002]. Interaction will no longer be device centric, but information centric, allowing both implicit and explicit interaction with the information represented and sensed by physical objects [Schmidt, Kranz et al. 2005].

Smart objects research explores embedding sensing and computing into everyday objects - augmenting objects to be a source of information on their identity, state, and context in the physical world. While many research projects have focussed on adding input capabilities to objects, less attention has been paid to adding output, creating an imbalance in the interface. Giving an object output capability allows users to address tasks in physical space with direct feedback on the objects themselves. This thesis investigates how we can support physical objects simultaneously as input and output medium, redressing the imbalance in the interface.

The first chapter presents a short motivation for the work, a section on the specific problems we identified, describes our approach and the contributions we make.

1.1 Smart Object Output

As the interest in creating smart objects grows they are expected to bridge the gap between the physical and digital world, becoming part of our lives in economically important areas such as retail, supply chain or asset management [Lampe and Strassner 2003; Siegemund and Flörkemeier 2003; Decker, Beigl et al. 2004] and health and safety monitoring in work places [Strohbach, Gellersen et al. 2004; Kortuem, Alford et al. 2007]. A major challenge for the design of smart objects is to preserve their original appearance, purpose and function, thus exploiting natural interaction and a user's familiarity with the object [Schmidt, Strohbach et al. 2002].

In many cases we do want to augment an object with digital information, for example, to reveal otherwise hidden information stored inside the object or give direct visual feedback to a user from object manipulation. An ability for objects to function as both input and output medium simultaneously enables scenarios such as objects that monitor their physical condition and visually display warnings if these are critical [Strohbach, Gellersen et al. 2004]. However, delivering this visual information to the user by adding output capability to objects is a challenge.

One approach is to routinely embed displays in objects. For example, thin, flexible displays such as e-paper are expected to become common in a few years time. However, this approach is expensive, especially if the object is disposable, and inflexible, as the product designer must choose at design-time which surfaces are to be augmented. An additional problem is that embedded displays change an object's appearance and can change the way an object is used. For example, consider that adding a display to a smart cup would prevent it being put in the dishwasher unless the display was removable. Either way, we have changed the object's natural appearance and function.

Another way we could visualise this information is by using Augmented Reality display devices, such as head mounted displays (HMD), tablet PCs, PDAs or mobile phones. However, carrying special purpose devices is encumbering for a user and limits interaction to a single person, whereas there are many times when it would be beneficial for a group of users to see the same display simultaneously.

In contrast, the recent availability of small, cheap and bright video projectors makes them practical for augmenting objects with non-invasive projected displays. By adding a camera and using computer vision techniques, a projector-camera system can also dynamically detect and track objects [Borkowski, Riff et al. 2003; Ehnes, Hirota et al. 2004], correct for object surface geometry [Pinhanez 2001; Borkowski, Riff et al. 2003; Ehnes, Hirota et al. 2004; Bimber and Raskar 2005], varying surface colour and texture [Fujii, Grossberg et al. 2005] and allow the user to interact directly with the projected image [Kjeldsen, Pinhanez et al. 2002]. The use of a projector-camera system does not rely on adding to or modifying the hardware of the object itself, instead creating temporary displays on objects in the environment without permanently changing their appearance or function.

The traditional way of augmenting objects with projector-camera systems (such as that taken by Pinhanez [Pinhanez 2001] and Raskar et al. [Bandyopadhyay, Raskar et al. 2001; Raskar, Beardsley et al. 2006]) is to store all information about the object in the projector system itself. Usually such systems are installed as infrastructure in the environment, creating a smart environment. This approach reduces flexibility, as it requires a-priori knowledge about all possible objects which can enter the environment. This results in large databases of objects and consequently higher possibility of inter-object confusion during detection.

1.2 Cooperative Augmentation

The core contribution of this thesis is the development of a new approach called Cooperative Augmentation to support physical objects simultaneously as input and output medium, as illustrated in Figure 1.1.

Instead of storing knowledge about objects in a smart environment, the Cooperative Augmentation concept distributes knowledge into smart objects. Hence, the intelligence becomes embodied in the smart objects inhabiting a space, not the space itself.

By moving the knowledge and intelligence from the environment to the smart object a projector-camera system no longer needs a-priori knowledge of all objects. This allows us to make projector-camera systems ubiquitous, as they merely offer a generic display service to all smart objects.

In the Cooperative Augmentation approach cooperation between smart object and projector-camera system plays a central role in detection, tracking and projection. The objects themselves become pro-active clients of the environment, allowing spontaneous

use of projection for output capability by requesting use of the projector-camera display service. Continued cooperation allows smart objects to describe information to the projector-camera system vital to the visual-detection and projection process, such as knowledge of their appearance. This knowledge is used to dynamically tailor the projector-camera services to each object.

Many objects possess embedded sensors designed for specific purposes. In the Cooperative Augmentation approach sensor information from the object can be serendipitously integrated in the detection and tracking process. The additional information from sensing allows projector-camera systems to dynamically constrain the detection process and increase visual detection performance.

After detection the smart object controls the interaction with projector-camera systems. The smart object issues projection requests to the projector-camera system, controlling how the projected output on its surfaces changes and allowing direct visual feedback to interaction.

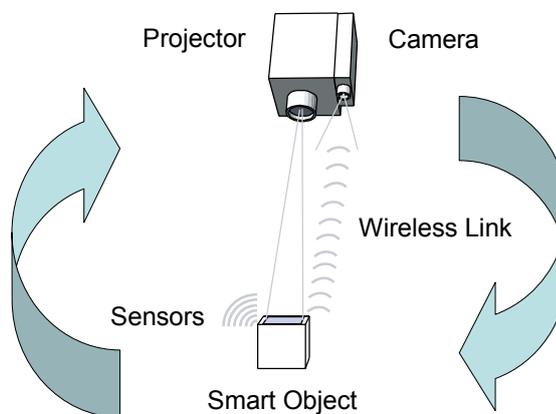


Figure 1.1 Cooperative Augmentation of smart objects with Projector-Camera Systems

1.3 Challenges

The key concepts of Cooperative Augmentation discussed above raise questions which we address in this thesis.

The initial question is how we can model the object and projector-camera system so that objects can cooperate by describing relevant aspects of their knowledge. This modelling must allow applications using the Cooperative Augmentation approach to be written independent of a particular system, but adapt themselves to it.

A central problem to achieving displays on everyday objects (smart or not) is also their detection and tracking. Only when an object is detected can the projector-camera system align its projection so the image is registered with the object's surfaces. In this thesis we use vision-based detection as it does not require adding dedicated location system hardware to every object to enable detection.

In experimental prototypes, vision-based detection is commonly achieved with fiducial markers [Ehnes, Hirota et al. 2004]. However, with a view to ubiquitous augmentation of objects using the Cooperative Augmentation approach, it is more realistic to base detection on the natural appearance of objects. This detection is a significant challenge in real-world environments, as objects naturally vary in their appearance. For example, one book could have a red cover, while a second book has a

blue cover. Similarly, one book could be open and one closed. Here the book objects are fundamentally identical in concept and use, but vary in both appearance and form.

Another challenge is that objects can appear at any distance and orientation with respect to a tracking camera. For example, we can see detail on a book cover when it is close to the camera, but this detail disappears when it is far away. Similarly, when we change our viewpoint from looking at the cover of the book to looking at its side we now see white pages instead of a colourful cover. To understand how best to detect and track smart objects using natural appearance detection methods we must therefore study the impact of scale and rotation.

One of the limitations of vision-based detection is that it has many failure modes. Some common reasons for detection failure include changes in object appearance, changing illumination, occlusion of the object, distractions in the image and fast movement of the camera or object. The Cooperative Augmentation approach enables integration of sensor information in the detection process, but to best understand how to integrate this information we must study how this sensing can be best exploited to increase detection performance.

1.4 Contributions

Giving an object output capability is valuable as it allows users to address tasks in physical space and receive direct feedback on the objects themselves, where it belongs in the real world. However, adding output capability to smart objects while preserving their original appearance and functions is challenging.

To address this challenge we present the Cooperative Augmentation approach, contributing the following to the area of ubiquitous computing:

1. The novel concept of Cooperative Augmentation, which enables:
 - a. Generic projection services in the environment.
 - b. Spontaneous use of projection capability by smart objects.
 - c. Detection and tracking of mobile objects in the environment.
 - d. Non-invasive output capability on smart objects.
 - e. Use of smart object as input and output interface simultaneously.
2. Validation of the Cooperative Augmentation concept through:
 - a. A system architecture specifying the different cooperating components.
 - b. Implementation of the system architecture.
 - c. Three demonstration applications.
3. An investigation of natural appearance vision based detection, providing:
 - a. A study to understand the impact of object scale and rotation on different natural appearance detection methods.
 - b. Insight into training requirements of different detection approaches.
4. An investigation of the integration of embedded sensing in the detection process, providing:
 - a. A study to analyse the increase in detection performance achieved with movement sensing in the target object.
 - b. Insight into the requirement for multiple detection approaches.

1.5 Thesis Structure

This thesis explores the cooperative augmentation of smart objects with displays using projector-camera systems:

Chapter 2 provides an overview of related work relevant for understanding the thesis in the four areas of Ubiquitous Computing, Augmented Reality, Computer Vision and Tangible User Interfaces. Here we aim to place our work relative to other research and explain some of the reasoning behind our approach.

Chapter 3 presents the Cooperative Augmentation conceptual framework in more detail, specifically the Object Model, the projector-camera system model and the cooperation process to achieve interactive displays on smart object surfaces.

Chapter 4 investigates four natural appearance computer vision object detection approaches and presents an experimental study exploring the issues of scale and rotation to enable detection of objects at any distance from the camera and in any pose. We also create an object appearance library for use in our studies.

Chapter 5 presents an experimental study exploring cooperative detection between the vision detection system and smart object, specifically analyzing the increase in detection performance achieved with basic movement sensing in the target object. We also create a video test library for use in our studies.

Chapter 6 presents the architectural design and implementation of the conceptual framework developed in Chapter 3, and evaluates the implementation as a whole.

Chapter 7 presents three demonstration applications created using the architecture. These applications are developed to demonstrate three different areas of the architecture, specifically scenarios presenting the whole system from an object entry to exit; interaction methods and a scenario with multiple-projectors and multiple-objects.

Finally, **Chapter 8** presents a summary of the thesis, the limitations of our approach and implementation, our conclusions and our future work plans.

Appendix A additionally describes the design and assembly of two steerable projector-camera systems, constructed for our experimental studies and demonstration applications. We identify characteristics of typical systems and present recommendations for building similar equipment. We characterise our system and compare the performance with other related research projects.

Appendix B provides additional examples of programming smart object state models for use with the Cooperative Augmentation framework.

Chapter 2 Related Work

This chapter does not provide an exhaustive list of related research projects; instead it aims to cover the most relevant related work for understanding the thesis and to outline this thesis' contribution in terms of other projects and similar research.

2.1 Introduction

As shown in Figure 2.1, this work draws on four areas of computing: Ubiquitous Computing, Augmented Reality, Computer Vision and Tangible User Interfaces (TUI). The main area is Ubiquitous Computing, as seen on the far left in Figure 2.2, which was defined by Mark Weiser as the third wave of computing after monolithic and personal computing. This third wave involves the technology receding into the background and many computing devices being unobtrusively embedded in the environment to enrich our lives [Weiser 1996].

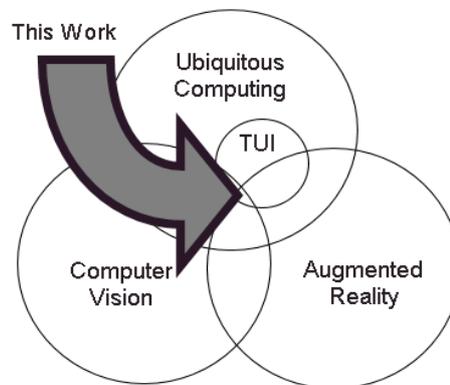


Figure 2.1 This work draws from 4 main areas of computing: Ubiquitous Computing, Augmented Reality, Computer Vision and Tangible User Interfaces (TUI)

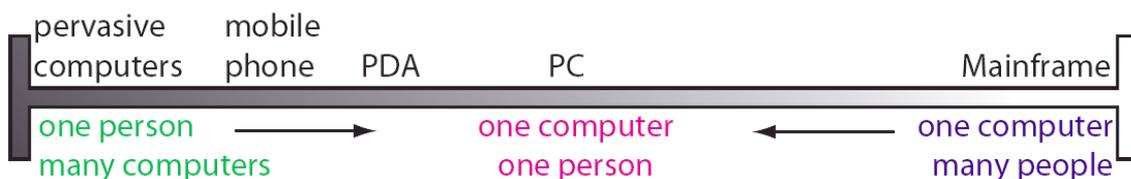


Figure 2.2 The "Weiser Continuum" of ubiquitous computing [Weiser 1996] to monolithic computing, taken from [Newman, Bornik et al. 2007]

The second area is augmented reality, which is a type of the Mixed-Reality (MR) visual display. Milgram and Kishino describe MR as a general set of technologies which involve the merging of real and virtual worlds [Milgram and Kishino 1994]. These technologies can be decomposed with respect to the amount of reality present in the interface to create the “virtuality” continuum, as shown in Figure 2.3. The continuum ranges from a zero virtuality interface in the real world (such as a pen and paper interface) to fully virtual environments where a user is immersed in surrounding virtual world (such as CAVEs).

In contrast, Augmented Reality (AR) refers to cases where either a display of the real environment, or objects in the real world themselves are augmented by means of virtual objects, using computer graphics [Milgram and Kishino 1994]. This allows a shift in focus for human-computer interaction from a static interaction with a user sat at a desktop display, to one where the surrounding environment and physical objects in this environment become the interface. Typical applications for AR include navigation, visualisation and annotation for medical, assembly, maintenance and repair tasks, entertainment, education, mediated reality, collaboration of distributed users and simulation. In this thesis we concentrate on projector-based AR, however, a good general introduction to all aspects of AR can be found in [Azuma 1997; Azuma, Baillet et al. 2001].



Figure 2.3 The Milgram and Kishino “Virtuality” Continuum of Reality to Virtual Reality [Milgram and Kishino 1994], taken from [Newman, Bornik et al. 2007]

To achieve an AR display so it appears correct for a user requires that the registration between the real and virtual world is exact. Consequently, for mobile users or dynamic environments the relative positions of the real and virtual must be continuously determined. This can be accomplished either by detecting and tracking objects, users, or both, depending on the scenario. Different approaches to tracking are possible, including using dedicated hardware such as mechanical, electro-magnetic and optical tracking systems [Rolland, Baillet et al. 2001]. In contrast, vision-based tracking using cameras allows a relatively low-cost and non-invasive approach, either using fiducial markers or markerless computer vision techniques [Lepetit and Fua 2005].

Traditionally, head mounted displays (HMD) have been used to view augmented reality displays [Kato and Billinghurst 1999]. However, HMD typically suffer from a number of drawbacks, such as limited Field Of View (FOV) due to their optics, low resolution due to the miniature displays and heavy weight. More recently, PDAs, tablet PCs and mobile phones have been used [Wagner and Schmalstieg 2006; Schmalstieg and Wagner 2007]. In addition to the encumbrance of the physical devices, these technologies share the problem of accommodation, as the augmentation appears at a different location and depth than the objects in real world. These technological and ergonomic drawbacks prevent them from being used effectively for many applications [Bimber and Raskar 2005]. In contrast, projector-camera systems and projector based AR offers a possible solution, enabling displays directly on the surfaces of objects in the real world.

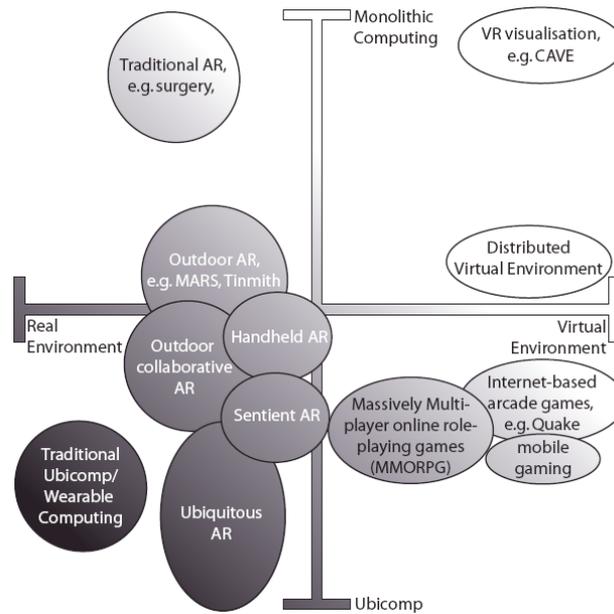


Figure 2.4 The Milgram-Weiser Continuum [Newman, Bornik et al. 2007]

Although Weiser believed ubiquitous computing to be the opposite of Virtual Reality (VR), Newman et al. point out that VR is merely at one extreme of the Milgram continuum, and propose merging the two continuums to create the Milgram-Weiser Continuum (as shown in Figure 2.4).

In this thesis we use projection to augment objects with interactive interfaces. The objects themselves are smart and cooperate with projector-camera systems to help with their detection and projection task. This places the work firmly in the Ubiquitous AR category of the Milgram-Weiser Continuum.

To achieve the augmentation of the smart objects we use markerless computer vision techniques to detect and track the objects in the environment. Through our cooperative augmentation framework the objects themselves can become Tangible User-Interfaces (TUI), where location, orientation, object geometry, projected interfaces and sensors become methods for interacting directly with the projected displays, and hence, otherwise hidden knowledge in the object and environment.

The following sections of this chapter investigate these areas in more detail.

2.2 Ubiquitous Computing

This section describes the closely related fields of ubiquitous computing (responsible for the drive to make everyday objects smarter) and tangible user interfaces. In recent years the ubiquitous computing field has become broader; however, in this thesis we concentrate on augmenting physical-tangible smart objects with projected displays. Here we examine what smart objects are, investigate some typical uses and concentrate on understanding how the potential for output capability can benefit objects and users.

Recent technological advances have allowed computing devices to be miniaturised enough to be embedded into everyday objects. The Smart-Its project envisioned these

could be attached to objects to augment them with a "digital self" [Holmquist, Gellersen et al. 2004]. So called "smart objects" would have the ability to perceive their environment through sensors and provide resources to nearby users and objects via peer-to-peer communication and customisable behaviour. However, the computing itself is secondary to the object, with the computer placed in the background of a users' interaction with the physical and social environment [Beigl and Gellersen 2003].

Mark Weiser proposed that "the real power of the concept comes not from any one of these devices: it emerges from the interaction of all of them. The hundreds of processors and displays are not a 'user interface' like a mouse and windows, just a pleasant and effective 'place' to get things done" [Weiser 1991], hence, collections of smart objects become a collaborative interactive experience. Sensors providing objects with awareness of physical context and communication allows objects to promote a digital presence and to become part of networked applications.

2.2.1 Sensor Nodes

There are many experimental sensor node platforms in use for augmenting everyday objects with sensing and computation, however here we concentrate on two typical devices – Smart-Its and Motes.

The Smart-Its platform for embedded computing has developed over the years, with early DIY versions [Strohbach 2004] and more-recent versions such as the Particle computer [Decker, Krohn et al. 2005]. A basic objective of Smart-its platform is to be generic, to enable operation in mobile settings and have ad-hoc peer-to-peer interoperation, while allowing customisation of sensors, perception and context-awareness methods. This is achieved using a flexible system with hardware consisting of main-boards with a PIC18F6720 microcontroller for processing, a TR1001 transceiver for communication on 868MHz and extendable pluggable sensor boards, as shown in Figure 2.5. Powered is provided by a 1.2V AAA rechargeable battery.



Figure 2.5 (left) Smart-Its design (centre) Particle Smart-Its Device main board, (right) Add-on sensor boards [Decker, Krohn et al. 2005]

The Particle device has a ball switch on the main board, which can be used for applications such as movement detection or power saving. Based on an analysis of typical sensor nodes applications the Smart-Its design developed two sensor boards. Sensor board (a) in Figure 2.5 (right) and Sensor board (b) which adds on-board processing with a PIC18F452 microcontroller, enabling applications that require more processing power. Both boards have a range of sensors available, including accelerometer sensors (2D or 3D), temperature sensor, light sensor (visible and IR), microphone, and force sensor. For actuation, the main board carries 2 controllable LEDs and a speaker for audio notification. More information on Smart-Its can be found in related publications [Beigl and Gellersen 2003; Holmquist, Gellersen et al. 2004; Decker, Krohn et al. 2005].

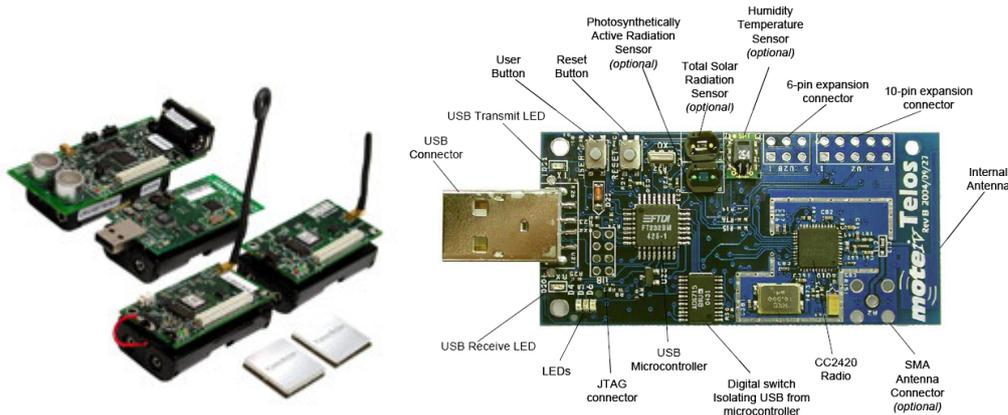


Figure 2.6 (left) Crossbow Technology Motes (right) Motive Corporation (now Sentilla Corporation) Telos Mote [Inc 2007]

Motes were initially developed in a collaboration between the University of California, Berkley and Intel Research [Nachman, Kling et al. 2005]. Motes are small, self-contained battery-powered devices with embedded processing and communication, enabling them to exchange data with one another and self-organize into ad hoc networks. Hence, similar to Smart-Its, the Motes form the building blocks of wireless sensor networks. The Motes typically run a free, open source component-based operating system called TinyOS. Motes are manufactured by Crossbow Technology, Berkley and Sentilla Corporation. The Motes family have a large number of designs, with an equally large number of pluggable daughter-boards. They support many of the same sensors as Smart-Its devices such as the integrated temperature, light and humidity sensors shown in Figure 2.6 (right). More information can be found on the related company and university websites.

2.2.2 Smart Objects

The Mediacup was one of the first projects to demonstrate sensing and computation embedded unobtrusively in an everyday object, as shown in Figure 2.7 (left). Here a coffee-cup was made smart by means of an attachable rubber foot containing computing to sense context with movement and temperature sensors, for example, sensing if the user drinks, or plays with the cup [Gellersen, Beigl et al. 1999]. It had the ability to communicate with other devices, such as smart watches and smart door plates to share its context. In this case, if a smart door-plate detected several Mediacups inside the room it displayed a “Meeting” sign, warning others.

Through this embedded computing the system gave an added value in the backend, allowing applications such as a coffee cup that provided the location of the user and a map which visualised building activities. This was accomplished without changing the appearance or function of the cup itself or the behaviour of the user. For example, rather than adding a display to the cup (which would change its appearance and prevent it being put in the dishwasher), a smart watch displayed the cup temperature on its LCD.



Figure 2.7 (left) Mediacup Object [Gellersen, Beigl et al. 1999], (centre) Cooperative Chemical Containers, (right) Hazard warning display using 3 LEDs [Strohbach, Gellersen et al. 2004]

Strohbach et al. present a scenario for embedded computing in industrial environments [Strohbach, Gellersen et al. 2004]. In this scenario large industrial storage facilities potentially contain thousands of chemical containers with different contents. Health and Safety rules apply to where and for how long these containers can be stored. Instead of augmenting the environment to track all the objects, Strohbach et al. propose embedding the rules directly into the containers and using embedded sensing to detect their state and their location relative to nearby containers. The containers can now cooperate, sharing their context to determine whether they comply with the rules, and detect hazard conditions, such as potentially reactive chemicals stored together, critical mass of containers stored together and when containers are stored outside of approved areas [Strohbach, Gellersen et al. 2004]. Such potentially risky situations require action from the facility employees to avert any danger, but the employees are faced with potential problems – such as how to find one container that has been placed in the wrong place out of thousands, and where to return this particular container to. Strohbach et al. provided visual feedback to employees as three LEDs on the top of the container barrel – green for no hazard, yellow for a warning (e.g. container stored outside the approved area) and red for critical hazards (e.g. reactive chemicals in proximity), as shown in Figure 2.7 (right).



Figure 2.8 (left) Proactive Furniture Assembly [Antifakos, Michahelles et al. 2004], (centre) Display Cube Object [Terrenghi, Kranz et al. 2006], (right) Tuister Object [Butz, Groß et al. 2004]

The Smart Furniture project developed furniture with embedded computing, sensing and actuation to proactively guide the purchaser with the task of assembly. By attaching computing devices and sensors onto each individual piece of the furniture, the system can both recognise a user's actions and determine the current state of the assembly [Antifakos, Michahelles et al. 2002; Antifakos, Michahelles et al. 2004; Holmquist, Gellersen et al. 2004]. The task of furniture assembly is inherently complex for an

embedded system, both as the task itself can be accomplished in different ways, and as it must cater for different user experience. For example, beginners may need a full walk-through, while experts may only need to be rescued if they assemble a piece incorrectly.

The system augments both tools and different pieces of the furniture with different sensors, for example, a screwdriver with a gyroscope to sense rotation, a horizontal board with an accelerometer for orientation sensing, and a side board equipped with a force sensing resistor (FSR) to measure when the two boards are joined. Hence, the system monitors its state and suggests the next most appropriate action using strips of LEDs for visual feedback, as shown in Figure 2.8 (left).

This feedback was a green pattern on both edges of the boards when correctly assembled, a red pattern when a mistake is made, or a flashing green when there are multiple alternatives. After alignment, individual green LED direct the user to tighten screws with the screwdriver. From their user study it was determined that augmented furniture with even simple LED notification allowed both a measurable decrease in assembly time and reduction in assembly errors [Antifakos, Michahelles et al. 2004].

2.2.3 Tangible User Interfaces

While differing from ubiquitous computing, Tangible User Interfaces (TUI) research holds a common concern for physically contextualised interaction. In contrast to ubiquitous computing, TUI research is rarely interested in augmenting everyday objects, but instead producing new artificial objects that fuse together input and output within a single device. TUI propose using these devices to provide an explicit mapping between the physical and virtual worlds, enabling tactile sense exploration and spatial reasoning which exploits the human senses of touch and kinaesthesia, and allows familiar physical actions to be used as input [Ishii and Ullmer 1997].

Central to TUI is the concept of embodying information in tangible objects, where objects serve as tokens or containers for digital information (as demonstrated by Underkoffler et al. in the Luminous Room project, described in section 2.3). Similarly, physical objects can serve as input primitives or tools to manipulate digital information, creating the possibility of representing abstract entities and concepts with physical ones, potentially enabling more efficient cognition of the relationships involved [Valli 2005].

Physical objects can also be used to gather and infer information about the context of the user and their intentions – for example, if a user picks up a tennis racket, they may also be interested in the location of their tennis balls. Lamming and Bohm propose such a system using embedded computing to detect relative proximity between objects and warn a user if they forget items from their bag [Lamming and Bohm 2003]. Such context information can be useful even when not being grasped by a user, for example, as objects may be placed in a particular spatial arrangement for an activity. Here, adding more objects allows more degrees of freedom, and hence gives the user more control over the representation of complex spatial relationships.

The physical object properties and affordances (which Norman define as the qualities of an object that allows an individual to perform an action [Norman 1988]) themselves can also be used as part of the user interface – for example, a graspable object such as a cube can provide more than a six degree of freedom input by sensing interactions such as squeezing or twisting in addition to movement and rotation [Sheridan, Short et al. 2003]. The use of such affordances were demonstrated in the Cubicle project, where a tangible cube was augmented with embedded computing and sensors [VanLaerhoven,

Villar et al. 2003; Block, Schmidt et al. 2004]. 3D accelerometers detected orientation and allows detection gestures such as shaking or placing the object down on a surface to control a home entertainment centre, as shown in Figure 2.9.

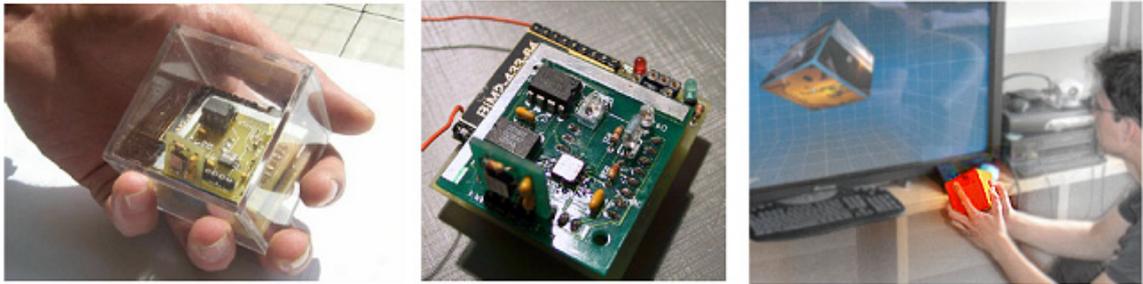


Figure 2.9 Using a sensor enabled Cubicle for interaction [Block, Schmidt et al. 2004]

The Display cube, shown in Figure 2.8 (centre), was designed by Terrenghi et al. as a child's learning appliance that exploited the familiar physical affordances of a cube, while augmenting it with embedded sensing, computing and displays [Terrenghi, Kranz et al. 2006]. The internal cube hardware uses a Smart-Its platform, 3D accelerometers to sense orientation, an LCD display on each face and a speaker for audio feedback.

The user interface was designed for games and quizzes, for example, a matching task where a letter was shown on the top face and children had to select the matching character from the other faces. Applications had to be specially designed to cope with the low resolution display and the lack of electronic compass sensors, which prevented sensing of the rotation around the gravity vector. However, in the study Terrenghi et al. found that users could still easily read letters and numbers in any orientation, suggesting that lack of orientation sensing is not a major issue for very simple text and graphics.

The user study also illustrated additional benefits of a tangible cube, with children quickly engaging with the games due to the shaking and turning, and cooperative play with children demonstrating to each other the solutions to the tasks.



Figure 2.10 (left) Prototype mock-up display cube using static front-projection, (centre) The current LED-matrix display Z-agon interactive cube, (right) The envisioned fully interactive cube with colour displays [Matsumoto, Horiguchi et al. 2006]

Matsumoto et al. envision a device similar to the Display cube, but with seamless displays covering each side of their Z-agon cube. They believe that users will be able to interact more easily with digital information when presented with an intuitive tangible interface [Matsumoto, Horiguchi et al. 2006]. For their initial prototype they front projected onto a large-scale mock-up, as shown in Figure 2.10 (left). Their current prototype is a 2.5 inch cube using low-resolution LED-matrix displays and 3D

accelerometers; however, they envision a fully interactive high-resolution video-player and communication device with colour displays, as shown in Figure 2.10 (right).

The Tuister was designed as one-dimensional input and output device based on discussions with cognitive psychologists which indicated that cubes may have too many degrees of freedom for users to locate and remember a display position (Sheridan et al. identified 17 non-verbal input affordances [Sheridan, Short et al. 2003]). Instead, the psychologists believe people were likely to lose track of their movements in a complex series of motions [Butz, Groß et al. 2004]. Consequently, Butz et al. designed the Tuister for browsing hierarchical nested menu structures with hardware consisting of two parts – a handle and display part which rotates to scroll through the menus, as shown in Figure 2.8 (right). Embedded in the display part are 6 organic LED displays each with 64x16 pixel resolution to display small symbols or short text. Sensors in a Smart-Its device detect the rotation of the display part with respect to the handle. Butz et al. envision Tuister as a personal multi purpose device, carried to interact directly with pervasive environments and serving as a universal remote control.

2.2.4 Input-Output Imbalance

The increasing amount of technology embedded into the environment enables new interaction metaphors beyond the traditional GUI paradigm. In the ubiquitous computing field there has been much research on input to smart objects, for example, on embedding sensing, location systems and methods for accumulating and distributing knowledge. However, there is little research on output methods to allow the user to visualise this knowledge and interact with it. Similarly, tangible user interfaces work allow physical manipulation and spatial arrangement of objects, but visual feedback to the user is mostly still provided by displays in the environment instead of the device itself [Butz, Groß et al. 2004]. This leads to an indirection in the interaction and an input-output imbalance.

As we have seen from sections 2.2.2 and 2.2.3, there is a real potential for objects and users to benefit from the delivery of complex and variable visual information on the object itself. This potential can be seen in diverse situations such as hazard monitoring in safety-critical workplaces, for assembly instruction, or for visual feedback to object manipulation in tangible user interfaces. The objects we examined attempted to redress the input-output imbalance either by using other displays in the environment or embedding displays in the physical objects themselves. These embedded displays took different forms, such as simple LEDs (e.g. the Chemical Containers and Smart Furniture) or multiple graphical displays (e.g. the Display cube, Z-agon and Tuister).

However, the display technology currently used has many limitations. For example, LED displays can only convey a small amount of information to users (such as flashing to indicate an error condition), while graphical displays are expensive, have high power requirements (so are difficult to integrate in low-power mobile objects) and change the appearance and function of an object. Hence, this motivates our approach in this thesis using non-invasive projected displays.

2.3 Projector-based Augmented Reality

The approach we investigate in this thesis uses projector-based augmented reality techniques to dynamically annotate physical and tangible smart objects with interactive projected information, solving their output problem. In this section we aim to

understand how we can create interactive, undistorted and visible projected displays on objects of varying shapes and appearances.

Projector-based Augmented Reality uses front-projection to project images directly onto physical object's surfaces, not just in a user's visual field. [Raskar, Welch et al. 1999]. Unlike physical displays such as computer monitors, projection-based displays allow integration with the existing appearance of an object to create seamless displays [Pinhanez and Podlaseck 2005]. The displays they create are non-invasive, as they do not require any hardware in the object being augmented. This feature allows projected displays to be used almost anywhere, in situations where physical displays would not be used, for example, due to cost, vandalism concern or hazardous environments. Projection can be used to change or supplement the functionality of the object - most commonly by transforming a non-augmented object "into an access point to the virtual information space" [Pinhanez 2001].

Projectors are able to create images much larger than the display device itself, allowing even small portable projectors to augment large objects. The traditional AR display technologies (Head Mounted Displays (HMD), PDA, tablets or mobile phones) are inherently encumbering and limited to a single user. In contrast, projected displays have scalable resolution, improved ergonomics, easier eye accommodation (as the graphics appear at the same distance in the real world as the objects) and a wide field of view, so are visible to multiple people. These properties enable a greater sense of immersion and have the potential to increase the effectiveness of multi-user interaction or co-located group-working in an object's physical space [Bimber and Raskar 2005].

Some of the earliest work in Projector-based Augmented Reality was Underkoffler, Ullmer and Ishii's Luminous Room project [Underkoffler, Ullmer et al. 1999]. This project developed a concept for providing graphical display and interaction on each surface of an environment. Co-located two-way optical transducers –called *I/O bulbs*– that consist of projector and camera pairs capture the user interactions and display the corresponding output. The Luminous Room also demonstrated the possibility of embodying information in tangible objects. Here, objects could "save" associated digital information, such as the chess board state, allowing the user to remove the object from the projected surface. The projected interface would then re-appear in the saved state whenever the object was placed back on the surface.

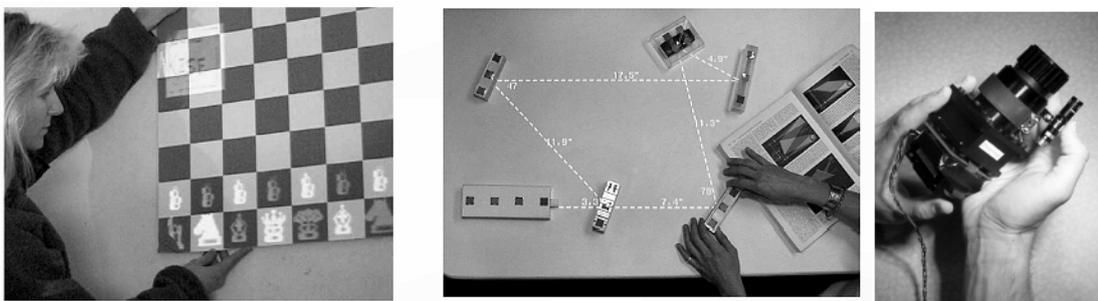


Figure 2.11 (left) A chessboard with memory (centre) Interactive optics-design, (right) A prototype I/O bulb projector-camera system in the Luminous Room [Underkoffler, Ullmer et al. 1999]

Underkoffler et al. present several design principles based on their experience in the luminous room. For example, people were often unable to distinguish between the real object and virtual projection. Consequently, Underkoffler et al. recommended subtle

animation was applied to the projections to make people aware of the virtual nature of the projections.

This reduced ability to distinguish between real and virtual was used deliberately in the Spatially Augmented Reality project [Raskar, Welch et al. 1999], which captured a physical object's inherent colour, texture and material properties (such as reflectance) with a camera, replaced them with a white object and projected imagery exactly reproducing the original appearance of the object. Raskar et al. showed that visually there was no difference between the original object illuminated with white light and the white object illuminated with the original appearance, indicating the ability to separate an object's appearance from its form. Using this separation Raskar et al. demonstrate alternate appearances, lighting effects and animation by simply changing the projection. This was also one of the first projects to demonstrate that projection is not restricted to a planar surface or a single projector, by augmenting complex 3D objects such as a model representation of the Taj Mahal mausoleum in India with multiple static projectors.

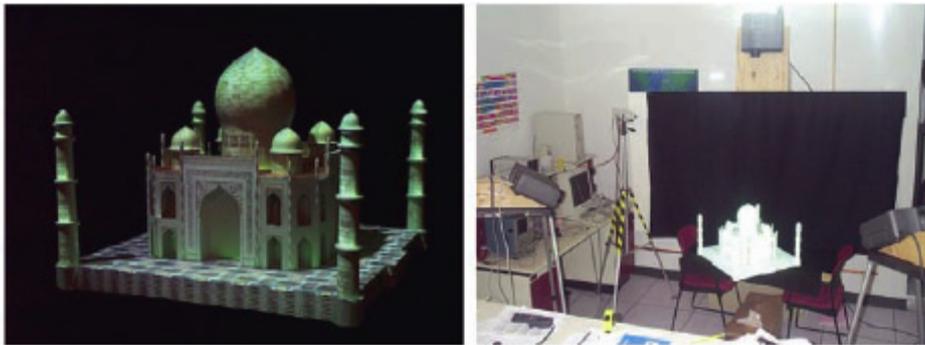


Figure 2.12 Spatially Augmented Reality (SAR) [Raskar, Welch et al. 1999]

As demonstrated by the I/O bulb and Spatially Augmented Reality project, the addition of a camera to the projector creates a feedback-loop and allows the display to become interactive. Projector-camera systems enable un-encumbered and un-tethered interactive displays on any surface, at any orientation. No special hardware is required in the display surface itself to support this interaction.

2.3.1 Projector-Camera Systems

The projection-based AR technology discussed in section 2.2 form part of a larger family of projector-camera systems. Here we decompose the family into three categories with respect to display mobility:

1. Static projector-camera systems (such as multi-projector display walls)
2. Mobile, handheld and wearable projector-camera systems
3. Steerable projection from a static system with pan and tilt hardware

These categories can be seen more clearly in Figure 2.13, when compared to a traditional desktop monitor which is a static, single user technology with a small display size.

All types of projector-camera system have been used for augmenting objects with projection, however, static [Bandyopadhyay, Raskar et al. 2001], mobile, handheld [Raskar, Beardsley et al. 2006] or wearable [Karitsuka and Sato 2003] (shown in Figure

2.15) projector-camera systems can only opportunistically detect and project on objects passing through the field of view of the projector and camera.

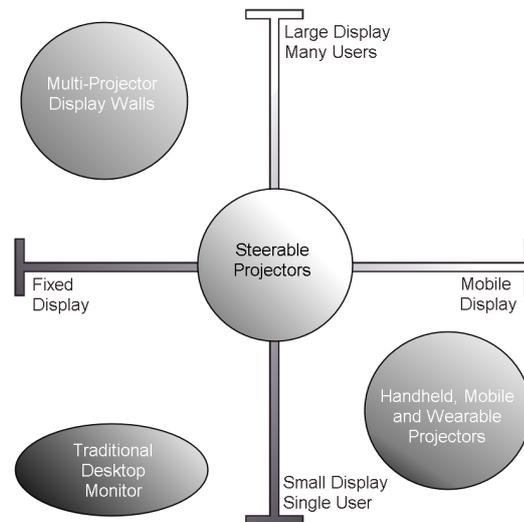


Figure 2.13 The Projector-Camera system family decomposed by mobility and display size, compared to traditional desktop monitors

In contrast, projector-camera systems in the third category, with computer controlled steerable mirrors or pan and tilt platforms [Pinhanez 2001; Borkowski, Riff et al. 2003; Butz, Schneider et al. 2004; Ehnes, Hirota et al. 2004] allow a much larger system field of view and the ability to track objects moving in the environment. We use the generic term “Steerable Projectors” for these systems.

2.3.2 Mobile, Handheld and Wearable Projector-Camera Systems

Raskar et al. initially developed the concept of handheld projector-camera systems in the “intelligent Locale-Aware Mobile Projector” (iLamps) project [Raskar, VanBaar et al. 2005]. As seen in Figure 2.14, the handheld projector-camera system included on-board computing, a tilt-sensor and network access. The projector-camera system was initially calibrated to determine the intrinsic (optical) parameters and extrinsic pose (relative locations and orientations) of projector and camera. Projection-based object adaptive displays were then demonstrated using circular fiducial markers to allow the system to calculate the camera pose (hence the projector pose). With projector pose known relative to an object (in this case the black rectangle with a fiducial marker in Figure 2.14), a projection can be registered with the object so it is overlaid on its surfaces.

Due to the fixed field of view of a projector, mobile and handheld projectors can only be used to reveal information in the environment in a similar way to a flashlight being used to illuminate a surface. This “peephole” metaphor [Butz and Krüger 2003] allows the user to see the windows and interfaces by “painting them with light”, onto an area of the environment. Sony first presented a prototype handheld projector (shown in Figure 2.15) containing accelerometers to measure the hand movement and paint a small world stabilised image on surfaces [Rapp, Michelitsch et al. 2004]. This concept was extended by Raskar et al. to project onto photo-sensing objects in the environment with their Radio-Frequency Id and Geometry (RFIG) project [Raskar, Beardsley et al. 2006].



Figure 2.14 (left and centre) The iLamps Project developed a handheld projector-camera system, (right) Detecting circular fiducial markers to augment a wall scene with projection

Handheld projectors are now widely researched due to recent technology developments in micro displays and cheap, long-life LED and LASER light sources. These developments have caused a drastic reduction in the size of projectors, which will soon enable projectors to be carried in a pocket or embedded in mobile devices. Some recent research concentrates on calibration of handheld projectors [Dao, Hosoi et al. 2007], while other projects make use of the ability of projectors to create displays larger than the device itself, focussing on collaborative interaction techniques using multiple handheld projectors [Cao and Balakrishnan 2006; Cao, Forlines et al. 2007].

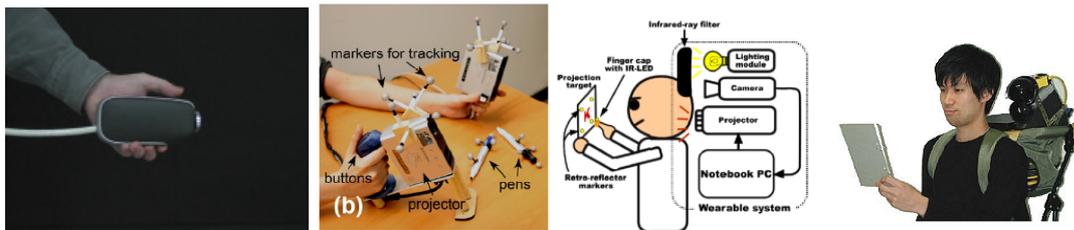


Figure 2.15 SONY's Handheld spotlight projector [Rapp, Michelitsch et al. 2004], (centre left) Multi-user interaction using handheld projectors [Cao, Forlines et al. 2007], (right) Wearable Projector-camera system [Karitsuka and Sato 2003]

2.3.3 Multi-Projector Display Systems

Traditionally, multi-projector displays were created by time-consuming mechanical alignment of the individual projectors to abut the images. Periodic re-calibration was often necessary, due to factors such as vibration which caused increasing alignment inaccuracies over time.

As part of the iLamps project [Raskar, VanBaar et al. 2005], Raskar et al. demonstrate the first ad-hoc clustering of overlapping projectors, using camera-based registration and image blending to create a single geometrically seamless display (see Figure 2.16, right). The project also developed shape adaptive displays, where the display was geometrically corrected to appear undistorted on multi-planar and curved quadric surfaces using the least-squares conformal mapping approach proposed by Levy et al. [Levy, Petitjean et al. 2002].

For perceptually seamless displays, multi-projector displays require photometric and colourmetric calibration in addition to geometric calibration to ensure uniformity of brightness and colour across the display. This is discussed further in section 2.3.7.



Figure 2.16 (left) The PixelFlex reconfigurable multi-projector display [Yang, Gotz et al. 2001], (right) Ad-hoc projector clustering and geometric correction for seamless display with iLamps [Raskar, VanBaar et al. 2005]

2.3.4 Steerable Projector-Camera Systems

Pinhanez at IBM proposed creating “interactive displays everywhere in an environment by transforming any surface into a projected touch screen” [Pinhanez 2001]. The system, which Pinhanez named the Everywhere Display (ED), uses a video projector and steering mechanism together with a camera to enable vision-based interaction with the projected imagery. The steering mechanism increases the area in which the display can be used, while the combination of projector and camera forms a powerful closed-loop visual control system, allowing display adjustment and unencumbered user interaction directly with the display.

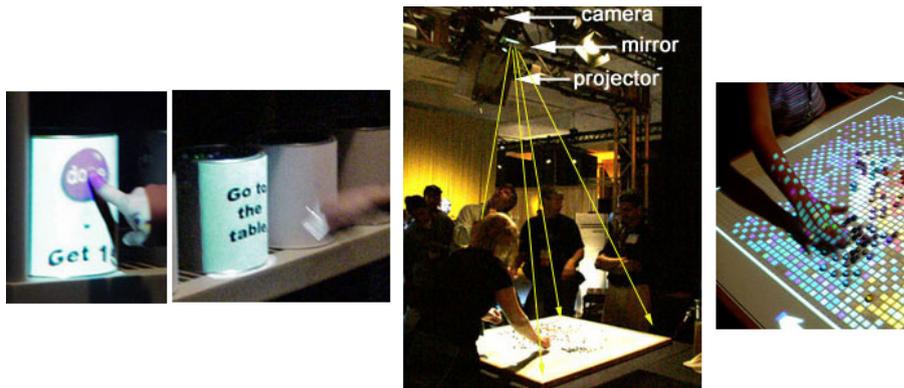


Figure 2.17 IBM's Everywhere Display demonstration at SIGGRAPH [Pinhanez 2001]

Pinhanez proposed steerable projector-camera systems for two classes of applications: interactive computer displays and spatial augmented reality. The steering mechanism enables ubiquitous interactive displays, as they can be situated on any surface or object within the field of view of the projector [Pingali, Pinhanez et al. 2003].

IBM first demonstrated simple augmented objects in their Everywhere Display demonstration at SIGGRAPH, seen in Figure 2.17. Here a table was augmented with a simple image display onto which the user could place individual coloured M&M candy “pixels” to build up a picture. To place the correct colour candy the steerable projector directed the user to a series of paint tins containing M&Ms, onto which were projected virtual buttons. The camera system detected user touch of this virtual button, transforming the paint cans into interactive interfaces.

With steerable projection, the information required in any situation can be brought to the location it is required. Pinhanez demonstrates the concept of projected information that is “attached” to a spatial location [Pinhanez 2001], allowing phone books to appear at the location of the phone when the user picked up the handset, however, this location

was static and had to be programmed into the system. Pingali et al. approach the idea of location based interaction from a different direction, by creating a display that followed the user, allowing important information to be constantly visible [Pingali, Pinhanez et al. 2002]. In this project a series of static display areas were pre-calibrated around the room, then a camera used to track the user. The steerable projector calculated the nearest visible display area to the detected user, and used this as the display.

As shown in Figure 2.18, Pinhanez et al. envision other uses, such as ambient displays in a user's periphery for messaging notification, dynamic navigation signage and augmenting objects for ubiquitous computer access.



Figure 2.18 (left) Messaging notification, (centre) dynamic navigation signage, (right) augmented filing cabinet object [Pinhanez 2001]

The steerable projector concept has been developed further, with the FLUIDUM intelligent environment project using fiducial markers (see section 2.5) tagged onto objects such as books to locate them [Butz, Schneider et al. 2004]. Following a pre-scan of the environment, the system could successfully find and highlight objects that had not moved.

Dynamic displays can also be created on mobile objects, such as the Portable Display Screen demonstrated by Borkowski et al. [Borkowski, Riff et al. 2003]. The display screen was a sheet of white card, modified by adding a black border to allow tracking with computer vision. The system allowed the user to carry a fully interactive display without any power or infrastructure requirements, apart from the projector-camera system. Ehnes et al. demonstrate a similar system [Ehnes, Hirota et al. 2004], where the white Personal Interaction Panel (PIP) was tracked using a fiducial marker.



Figure 2.19 (left and centre) Portable Projected Display Screen [Borkowski 2006], (right) Personal Interaction Panel [Ehnes, Hirota et al. 2004]

2.3.4.1 Existing Steerable Projector Frameworks

Levas et al. presented a framework for steerable projector-camera systems to project onto objects and surfaces in their Everywhere Display EDML framework [Levas, Pinhanez et al. 2003]. As shown in Figure 2.20, the EDML architecture comprised three

layers – the lower services level containing the actual hardware dependant implementation, the middle integration layer which abstracts and synchronises the hardware, and the high level application layer.

The lower levels provided applications for explicit user modelling of displays using a 3D world modelling tool and provided user localisation and geometric reasoning for use in applications such as the user following display [Pingali, Pinhanez et al. 2002]. The integration layer provided the main classes of the API to build applications with the framework, event management, geometric distortion correction and handling of interactive content to be rendered on the virtual displays.

While supporting a distributed architecture, the framework has a number of limitations. It is reliant on the user to explicitly model their world and pre-calibrate displays, hence is limited to known, static environments. Although dynamic occlusion detection of the pre-calibrated displays is possible, this is implemented by vision-based head tracking. Hence, the system cannot detect any non-human occlusion of pre-calibrated display locations (for example, by furniture). The framework also does not address any methods for multiple projectors to work cooperatively, assuming only a single projector in any environment.

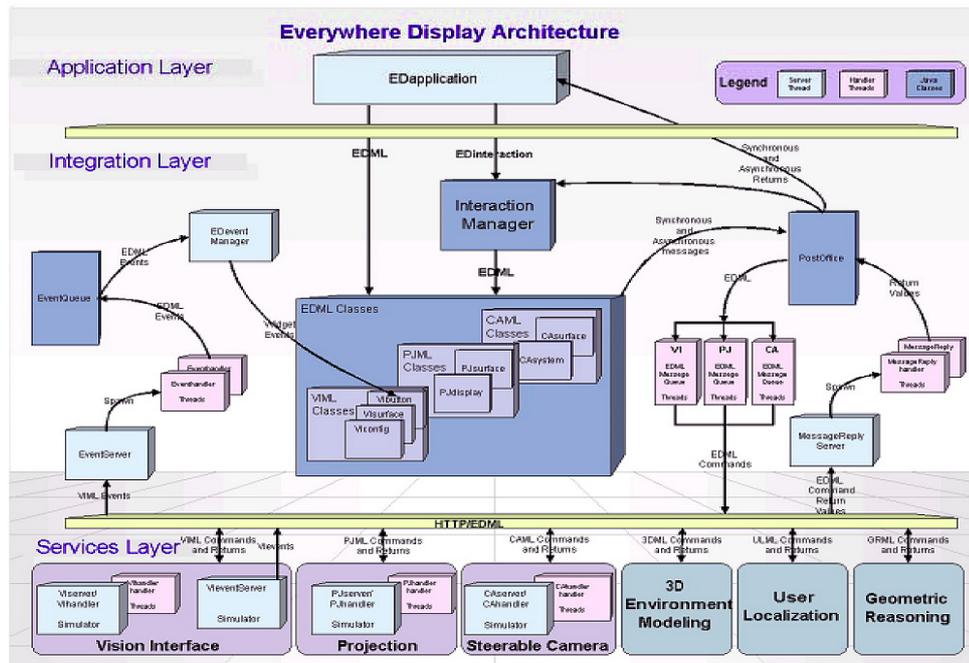


Figure 2.20 IBM's Steerable Interface EDML Framework

Pingali et al. [Pingali, Pinhanez et al. 2003] build on the EDML framework design to characterise steerable interfaces as exhibiting the six following qualities:

1. Moveable output interface - the ability to move video and audio around spatially.
2. Moveable input interface – such as steerable cameras and directional microphones.
3. Adaptation to user context - for example, the user's location and orientation.
4. Adaptation to environment - reasoning about the geometry and properties of surfaces, adapting to dynamic conditions such as occlusions and ambient noise.
5. Device-free interaction - using multi-modal input techniques for interaction.

6. Natural interaction - Intuitive and useable interaction, sensitive to user context.

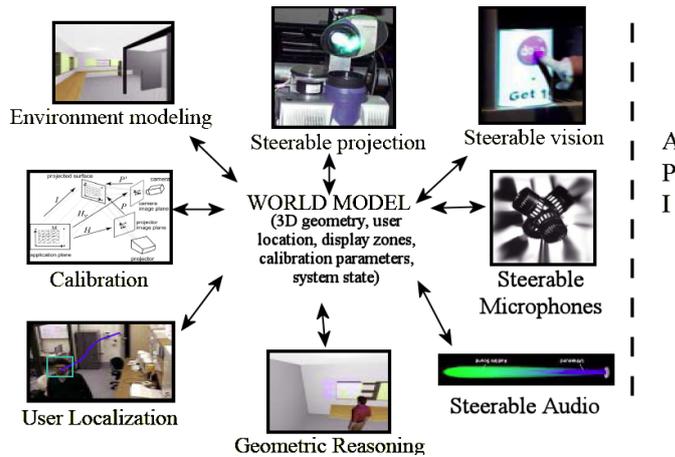


Figure 2.21 The IBM Steerable Interfaces Characterisation [Pingali, Pinhanez et al. 2003]

Although the characteristics Pingali et al. propose are a good way to describe steerable interfaces, they do not directly address the Ubiquitous Computing vision of the future, which proposes computing embedded into everyday objects. It is conceded that “special purpose” devices for interaction could be accommodated; however, there is no support for projection on mobile smart objects in their interaction paradigm.

We construct a steerable projector-camera system to enable vision-based detection, tracking and projection onto smart objects in our work. This steerable projector-camera system is discussed further in Appendix A.

2.3.5 Interaction with Projector-Camera System Displays

Without input techniques, the ability to create displays on objects is analogous to having a set of portable televisions which can be moved around at will, but only display one channel. To enable the most functionality for users requires a balanced interface, hence in this section we look at how we can interact with projected displays.

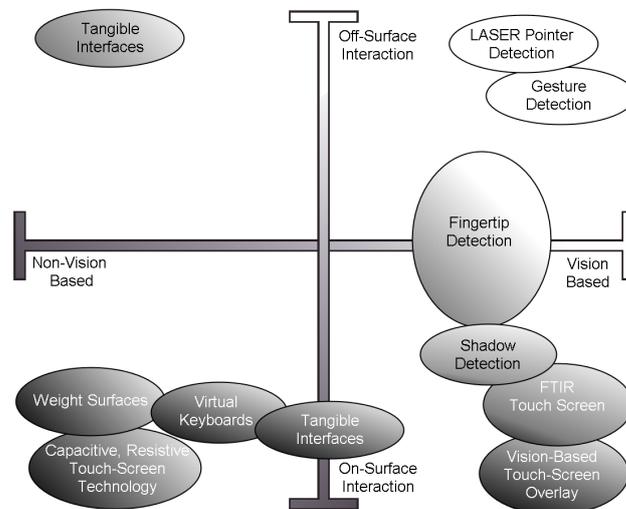


Figure 2.22 Interaction Techniques for Projected Displays

Many interaction techniques can be used with projected displays. Some of the main techniques in use are shown in Figure 2.22. Interaction can be primarily decomposed into on-surface techniques (where the user has physical contact with the projection surface) and off-surface (the user is remote). We also decompose the interface technology itself into vision-based and non-vision-based sensing.

The challenge with all interaction techniques is to distinguish the difference between pointing or hovering, versus activating [Buxton 1990]. For control activation one issue is that the interaction techniques discussed here all have little tactile feedback. Users cannot feel the edges of keys and the force when pressed, so performance will likely be much lower than normal keyboards or mice. Consequently, users will not be able to achieve eyes-free interaction [Lewis and Potosnack 1997]. However, common solutions to increase performance can be used, such as generating an audio or visual feedback for the user whenever a control is activated (e.g. simulating a key press click).

2.3.5.1 Off-Surface Interaction

Off-surface interaction techniques are primarily using pointing interfaces (LASER pointing [Kirstein and Müller 1998] or finger pointing) with vision detection. Pointer interaction techniques are similar to mouse interaction in that they allow target selection and gestures. However, Oh and Stuerzlinger's experiments reported a laser pointer is only approximately 75% as fast as a mouse at selection [Oh and Stuerzlinger 2002]. Similarly, Laberge and Lapointe report that by using a homography based camera to screen calibration (see section 2.3.6), an typical accuracy of only ± 2 pixels could be achieved [Laberge and Lapointe 2003]. Cheng and Pulo [Cheng and Pulo 2003] also state that laser pointer interaction suffers from three additional problems:

1. Pointer instability due to jitter in hand movement.
2. On-screen pointer latency relative to LASER due to low camera frame rates
3. Selection of objects is restricted to gestures, pointer on/off events or dwell times

They propose hiding the hand jitter and latency issues by either not showing an on-screen cursor (relying on the laser pointer spot itself as visual position feedback), or by using an invisible Infra Red laser pointer and showing only an on-screen cursor.

2.3.5.2 On-surface Interaction

On-surface interaction techniques based on non-vision techniques use a variety of mechanical sensing methods to detect the location of fingers in real-time. The main drawback with all these techniques is the requirement for sensing equipment to be embedded in the display surface and the need to then calibrate the display-to-surface relationship to achieve correct sensor-projector alignment.

Capacitive and resistive touch sensors are widely used in products such as smart whiteboards, laptop trackpads and touch-screen overlays. However, this technology is typically restricted to a single user and a maximum of one or two points of contact.

Systems such as the Mitsubishi DiamondTouch system [Dietz and Leigh 2001] and Rekimoto's SmartSkin [Rekimoto 2002] are able to support multiple simultaneous users with multiple points of touch, however, only DiamondTouch can differentiate between users. This relies on capacitive coupling and requires each user to remain in contact with a signal receiver to be visible to the system, limiting the interaction serendipity.

Weight surfaces triangulate the location of objects and fingers placed on their surfaces using weight sensors, allowing them to be used as generic pointing devices [Schmidt, Strohbach et al. 2002]. This approach has several benefits – it is robust,

reliable and can preserve the original functionality of the surface, allowing many surfaces to be augmented cheaply. The technique can also successfully determine active and passive loads, for example, allowing it to differentiate between a static mug on a table and a user's finger. However, it cannot differentiate between multiple users.

Virtual Keyboard devices are designed to be used with mobile phones and PDAs to allow fast, convenient text input without the bulk and portability issues of a physical keyboard. The devices project an image of a keyboard on a planar surface and use optical sensing (typically LASER or infra red break-beams) to detect when a user's finger enters the area of a key, generating a key press event. The reader is referred to [Kölsch and Turk 2002] for a full survey of virtual keyboards.

Examples of tangible interfaces are discussed separately in section 2.2.3.

Vision-based detection is attractive as it is non-invasive, requiring nothing more than a remote camera to detect interaction. Hence, hand and finger tracking is a large area of its own in Computer Vision research. Researchers typically employ a number of approaches, such as correlation tracking, model-based contour tracking, foreground segmentation and shape filtering [Borkowski, Letessier et al. 2004]. However, vision-based interfaces must contend with many challenges, such as variable and inconsistent ambient lighting, reflections, strong shadows and camera occlusion.

Pinhanez et al. [Pinhanez, Kjeldsen et al. 2002] proposed that the two goals for a vision-based finger detection system are to detect when a user touches a projected "button" and to track where a user is pointing on a projected screen. For some rear-projection screens, vision can be used directly through the surface of the screen itself, such as with Frustrated Total Internal Reflection (FTIR) sensing [Han 2005] or stereo-vision for detecting finger location and distance from the surface [Wilson 2004].

However, for interaction with front-projected displays typical vision approaches perform poorly due to the potential for projected light to radically change the appearance and colour of a hand, as was shown in Figure 2.17 (right). This can render traditional techniques that track the hand's shape or skin colour unreliable. Background subtraction also will not work if dynamic imagery is being projected significantly brighter than the background. As a result, some researchers have used Infra Red (IR) for detection. However this approach typically requires removing IR-blocking filters from consumer cameras, then physically fitting an IR-pass filter (preventing detection of visible light, and hence, colour) and separate IR illumination sources.

Some projects make use of shadows from separate IR illuminators or the projection light itself, using off-axis cameras to detect a finger's shadow on the projection surface. The location of the shadow relative to the detected finger allows calculation of the distance to the surface and the location when on the display [Kale, Kwan et al. 2004; Wilson 2005]. However, for on-axis cameras (such as a camera attached to a projector) Pinhanez et al. propose using a motion based approach to overcome the majority of the problems associated with front projection displays [Pinhanez, Kjeldsen et al. 2001].

The motion-based approach involves subtracting each frame from the frame before to create a motion mask. This approach can create a large amount of noise on the image, due to changes in the projected image or movement of objects and surfaces. Hence, morphological erode and dilate operations are performed to reduce noise and analyse neighbourhoods rather than individual pixels.

Pinhanez et al. first tried a fingertip template matching approach in their Everywhere Display interfaces, however, this approach was easily fooled as any occlusion of the

projected interactive area generated false-positives [Kjeldsen 2005]. The small fingertip template used could also easily match at multiple locations on an image, so a concept of user location had to be introduced and the match furthest from a user was assumed to be the correct fingertip (requiring a tracked user).

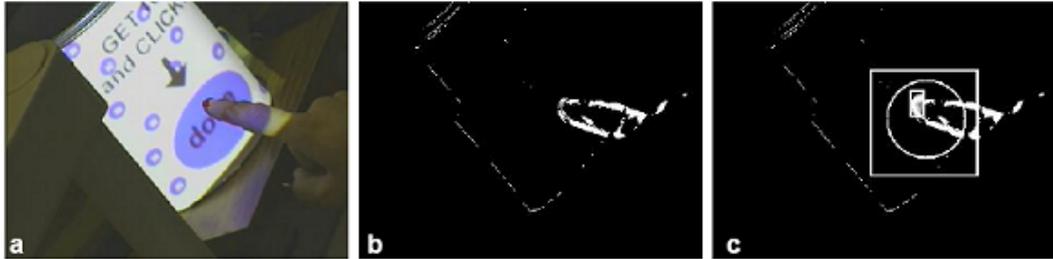


Figure 2.23 Fingertip matching in (a) Camera image, (b) Motion mask, (c) Finger tip template match [Pinhanez, Kjeldsen et al. 2001]

Kjeldsen proposes using a new technique called “Polar Motion Maps” [Kjeldsen 2005], which splits the circular area surrounding an interactive button (R_w on Figure 2.24) into segments like a cartwheel and analyses each segment for the direction of movement of the energy in the motion mask.

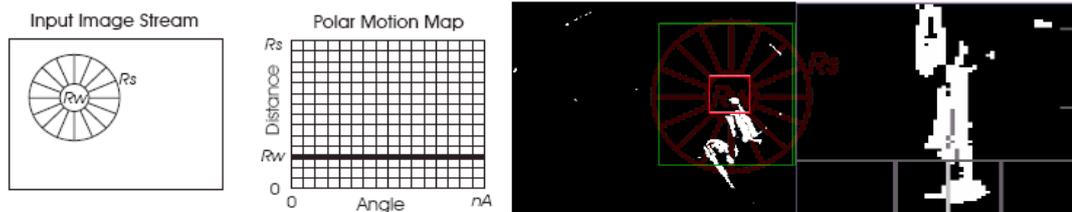


Figure 2.24 (far left) Polar Motion Map, (left) the corresponding segments unrolled to a rectangle, (right) PMM in use [Kjeldsen 2005]

The PMM segments can be converted from polar to Cartesian coordinates, and the graph analysed for long narrow objects that approach from a consistent direction, do not pass through the target (R_w) then retract from the direction in which they approached. This “lightning strike” movement was found by Kjeldsen to be indicative of touching movements, allowing events such as movement of a hand through the target or occlusion (which would not exhibit narrow profiles that a retracted in the same direction of arrival) to be disregarded.

Kjeldsen also proposes allowing users to define an interface arrangement themselves using tangible paper based representations of user interface widgets [Kjeldsen 2005], to. Here, the location of the paper objects defines the target for interaction and the identity of the object determines what interaction is possible. The Everywhere Display [Pinhanez 2001] vision system was then used to automatically generate an interactive interface, based on the spatial arrangement and identity of widgets it detects, as shown in Figure 2.25 (left).

Letessier and Bérard proposed another, simpler solution [Letessier and Bérard 2004]. By manually increasing the exposure of the camera until the projected light appears overexposed, the hand gains the correct exposure, as shown in Figure 2.25 (right). Using this method traditional finger tracking techniques can be used, but require the camera exposure to be set manually to optimise the finger visibility and assumes that the light levels in the whole environment remain constant.

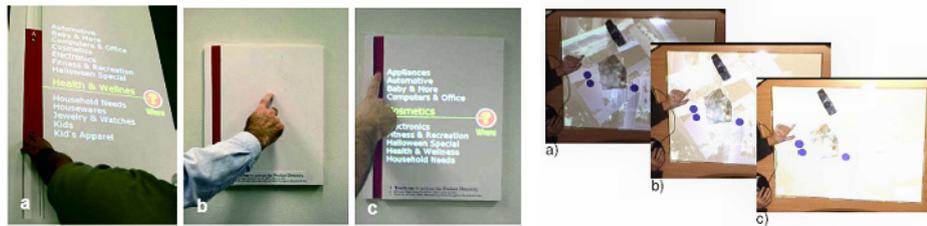


Figure 2.25 (left) Using a red paper slider widget [Kjeldsen 2005], (right) Increasing camera exposure to compensate for projector light [Letessier and Bérard 2004]

Letessier and Bérard’s method is not useful in our work; when we increase the exposure of the camera to remove the projection, this will over-expose the view of the objects, preventing us from tracking them with the camera. Consequently, in our work we use the method proposed by Pinhanez et al. [Pinhanez, Kjeldsen et al. 2002] to implement a non-invasive vision-based finger interaction system.

2.3.6 Projected Display Geometrical Calibration

Video projectors are designed to project light in a direction orthogonal to a planar projection surface [Pinhanez, Kjeldsen et al. 2002]. This location produces the best image on the screen, with no geometrical distortions. Projector or surface rotation away from this ideal orientation causes geometrical distortion of the projected image, often called “keystoning”. Similarly, non-planar display surfaces also distort the projection.

Many projectors allow compensation for small amounts of geometric distortion by digitally warping the image before projection so that it appears rectangular when displayed on the projection surface. A few projectors include an inclinometer sensor to perform automatic vertical distortion correction when a projector is rotated vertically [Raskar, VanBaar et al. 2005], however, an inclinometer on its own cannot correct for distortion caused by rotation in the horizontal plane or distortion due to rotation of the projection surface. Consequently, the correction must typically be specified manually by the user, using the remote control.

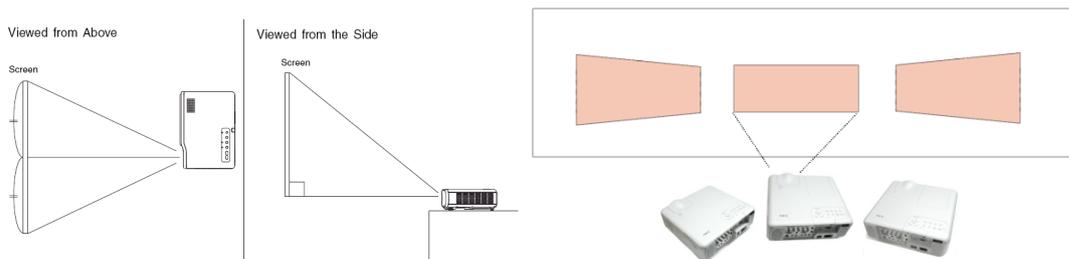


Figure 2.26 (left) On-axis projection gives a rectangular image, (right) Geometric distortion due to horizontal projector rotation when projecting onto a planar screen

It is possible to implement automatic geometric distortion correction in software by dynamically pre-warping the image to project. It is always possible to correct for oblique projection on simple geometries, as long as the projection surface does not have a shape that occludes the projected light [Pinhanez 2001]. This enables projectors to be mounted anywhere in the environment with respect to the projection surface. For example, in a meeting scenario the projector could be located at the side of the room, where a presenter is less likely to cast shadows on the screen and where the projector does not occlude the audience’s view [Sukthankar, Stockton et al. 2001].

Many different calibration strategies have been proposed to calculate the geometric correction required to allow an image to look un-distorted to an observer. Yang and Welch proposed an initial classification of calibration methods [Yang and Welch 2001], which was extended by Borkowski et al. [Borkowski, Riff et al. 2003]. The extended classification separates the methods based on whether they are on-line (real-time) calibration techniques performed while a system is operational, whether the system actively emits light or makes other active emissions to detect the surface, and whether the calibration is only valid for planar 2D surfaces or can be used for three dimensional surfaces. However, Borkowski et al. incorrectly classify the Laser Scanner, Structured Light and Stereo Vision calibrations and omit distortion correction based on projective texture mapping techniques. Our corrected classification is shown in Figure 2.27.

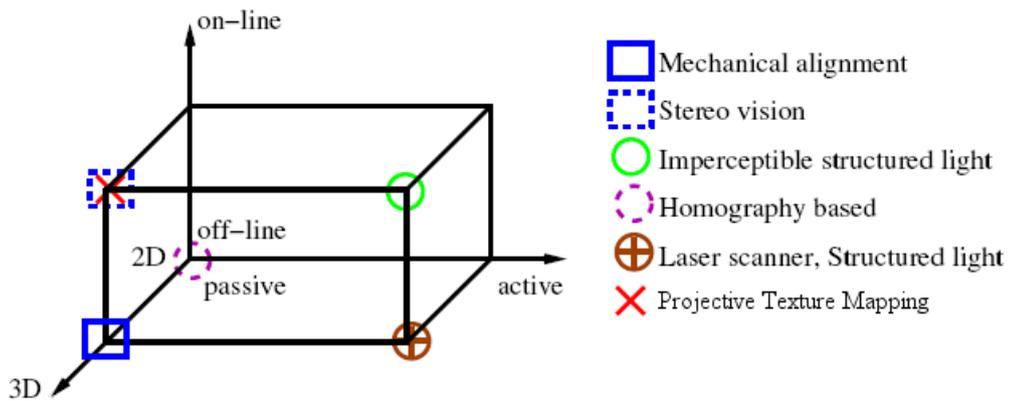


Figure 2.27 Classification of projector-camera system geometrical calibration methods, based on [Borkowski, Riff et al. 2003]

Of the geometrical calibration methods, the homography based and structured light techniques only require a projector and camera to be performed, however, the homography method is limited to planar surfaces. The projective texture mapping requires a-priori knowledge of the projection surface geometry and projector-camera pose (but can be used on-line when this is known), while the remaining techniques require additional hardware to enable calibration. The most common geometrical correction methods are discussed in more detail below:

2.3.6.1 Projective Texture Mapping

Geometrically, projection is the inverse of camera viewing [Pinhanez 2001]. Hence, it is possible to model the projection by creating a three-dimensional virtual representation of the projection surface and locating a virtual camera at the same position and orientation as the projector in the real world. By creating the virtual camera with identical optical parameters to the real projector, the image seen by the camera will exactly replicate what the projector sees in the real world. Distortion correction is then performed automatically by projective texture-mapping of an image onto a virtual surface with the orientation and size that it would appear in the real world. The virtual camera now views an image that will appear geometrically correct when projected.

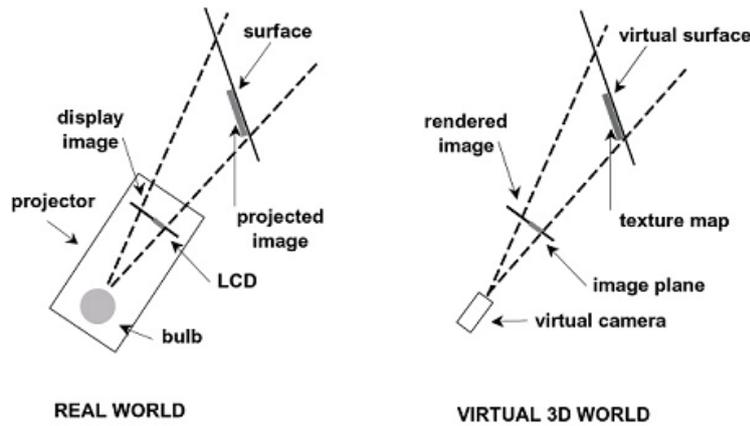


Figure 2.28 Model-based distortion correction based on known surface geometry and pose relative to the projector

[Pinhanez 2001]

This approach is conceptually simple, and can be easily implemented using 3D computer graphics hardware for real time performance when dynamically changing projector and display surface pose. However, the main weaknesses with this approach are that it requires an exact model of the environment to be created, and the exact pose of the projector relative to the display surface to be detected. This restricts its use to environments where the display geometry is known and tracked.

2.3.6.2 Planar Homography Calibration

Pinhanez et al. first developed a mathematical method for geometric calibration of projector-camera displays on planar screens using homographies [Pinhanez, Nielsen et al. 1999]. A homography is projective transform that links two planar surfaces in three-dimensional space. It is an exact mathematical description of the rotation, translation and scaling that maps each point between the two planes. To perform the calibration Sukthankar et al. first project a series of dots onto the screen and detect the location of the dots in the camera image, calculating the projector-to-camera transformation homography (T) [Sukthankar, Stockton et al. 2001]. By then detecting the location of the screen in the camera image (either using fiducial markers at the corners or the screen border edges) Sukthankar et al. calculate the camera-to-screen transformation (C). This allows the full projector image to screen transformation (P) to be recovered ($P=C^{-1}T$). These transformations were used to pre-warp an image before projection to align correctly with the screen, as shown in Figure 2.30.



Figure 2.29 Projecting dots onto a planar screen to recover the projector-camera homography

For example, to calculate the projector-camera homography we would first establish correspondences between the projector and camera (such as projecting and detecting the centre of the dots in Figure 2.29), then use the following formulas to calculate the conversion between the coordinates in the projector image (u, v) and camera image coordinates (u', v'):

$$u = \frac{au'+bv'+c}{gu'+hv'+1} \quad v = \frac{du'+ev'+f}{gu'+hv'+1} \quad (2.1)$$

These equations can be reformatted for homogeneous coordinates as follows:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \quad (2.2)$$

Given n non-collinear corresponding points on both the image and display surface (where $n \geq 4$), the correspondences can be reformatted as simultaneous equations in equation 3.3. By assuming the constraint $\|h\|=1$, the coefficients (h_{11} to h_{32}) can be determined using Gaussian elimination if $n=4$, or linear least-squares with Singular Value Decomposition (SVD) when $n > 4$.

$$\begin{pmatrix} u'_1 & v'_1 & 1 & 0 & 0 & 0 & -u'_1u_1 & -v'_1u_1 & -u_1 \\ 0 & 0 & 0 & u'_1 & v'_1 & 1 & -u'_1v_1 & -v'_1v_1 & -v_1 \\ \vdots & \vdots \\ u'_n & v'_n & 1 & 0 & 0 & 0 & -u'_nu_n & -v'_nu_n & -u_n \\ 0 & 0 & 0 & -u'_n & -v'_n & 1 & -u'_nv_n & -v'_nv_n & -v_n \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ \vdots \\ h_{31} \\ h_{32} \end{pmatrix} = 0 \quad (2.3)$$

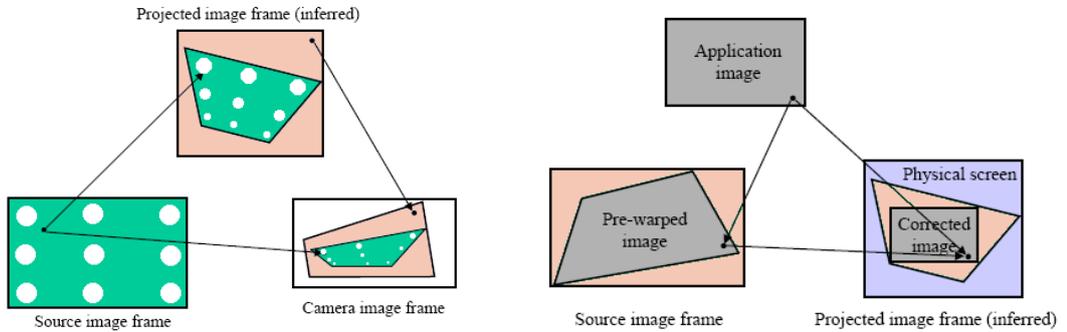


Figure 2.30 (left) Creating the projector-camera homography, (right) Using the combined projector-screen transformation to project an undistorted image [Sukthankar, Stockton et al. 2001]

The calibration performed by Sukthankar et al. was for a static system. However, assuming the projector-camera relationship is fixed (e.g. the camera is attached to the projector) and the camera can detect and track a minimum of 4 non-collinear points on the target planar surface, this method works in real time. Borkowski et al. use this method to demonstrate real-time projection geometrically aligned with their portable display screen, by determining the 4 corners of the black borders [Borkowski, Letessier et al. 2004].

In static environments where projection is unconstrained and no visually defined screen area exists, Borkowski et al. also proposed an off-line pre-calibration of environment [Borkowski, Letessier et al. 2004]. This calibration enables a projector-camera system and off-axis camera to detect planar surfaces and store them for future use. A world model is built by scanning the room while dynamically projecting a series

of straight lines. Lines which have a curved appearance in the camera image indicate a non-planar surface, while a step-like discontinuity is assumed to indicate the edge of a surface. Once all possible planar areas are detected, the projector can establish planar homographies and calculate the distance to the display surface.

The homography approach has been extended to support automatic geometric calibration of multi-projector display walls [Chen, Sukthankar et al. 2002; Brown, Majumder et al. 2005; Bhasker, Sinha et al. 2006] and multi-planar surfaces [Ashdown, Flagg et al. 2004].

2.3.6.3 Curved and Unknown Surface Calibration with Structured Light

Curved or spherical projection surfaces can be corrected using the off-line triangulation technique and quadric calibration method proposed by Raskar et al. [Raskar, VanBaar et al. 2005]. This approach projects a sequence of structured light patterns (see Figure 2.31, left) and detects left-right correspondences using a calibrated stereo camera system to triangulate the display surface geometry. A Quadric curve fitting approach is used with a minimum of 9 correspondences (compared to 4 for a homography) to enable geometric correction, as shown in Figure 2.31 (centre).



Figure 2.31 (left) Gray-code structured light patterns (centre) Geometrically corrected multi-projector display on curved display [Raskar, VanBaar et al. 2005], (right) Automatic Projector Display Surface Estimation Using Every-Day Imagery [Yang and Welch 2001]

As discussed by Raskar et al., the geometric correction of non-planar geometry requires a tracked observer to provide the correct undistorted view. It is always possible to generate a geometric distortion correction for an arbitrary viewpoint given known surface geometry and calibrated cameras and projectors [Park, Lee et al. 2006]. Hence, if the observer's viewpoint is not tracked it is usually assumed their viewpoint is perpendicular to the projection surface.

To calibrate completely unknown geometry Park et al. [Park, Lee et al. 2006] propose an off-line calibration technique which projects a series of structured light images while using triangulation between the projector and a single camera (by assuming the projector is equivalent to another camera). This approach relies on both a calibrated projector and camera, but can recover a grid of points on an arbitrary surface. The surface is modelled from the recovered 3D points by assuming it is piece-wise planar in the small (i.e. between neighbouring points). The recovered geometry mesh is then used to warp the projected image using homographies to correct for geometric distortion.

Yang et al. use an iterative on-line approach to recovering display surface geometry by making use of the closed-loop of a projector-camera system [Yang and Welch 2001].

The system initially assumes an orthogonal planar screen, then detects features in each image before projection, observes these features when projected and continually refines an estimate of the display surface geometry based on the difference in location using a Kalman filter. Over a number of frames this will converge to approximate the real surface, as shown in Figure 2.31 (right). This approach is in effect a structured light approach; however, the only structure comes from the projected content.

Johnson and Fuchs present a similar on-line approach for planar surfaces with real-time feature detection and tracking [Johnson and Fuchs 2007]. Their approach additionally predicts the pose of the projector at each frame, allowing use of a mobile projector with a static off-axis camera.

Another on-line structured light approach using explicit patterns was proposed by Park et al. [Park, Lee et al. 2007]. Here the patterns were embedding into the display imagery in a way that is imperceptible to observers, by projecting a pattern in one frame, then its inverse in the next frame. The pattern is adapted to the colour distribution and spatial brightness variation of the original projected content to make it further imperceptible. This approach requires a camera synchronised to the projector for the pattern to be visible to the detection system and creates some visible flicker for humans with a typical 60Hz projector refresh rate. Grundhöfer et al. report the patterns are still visible with fast eye movements above 75Hz and propose a new approach which creates patterns which are below the human perceivable threshold, but still detectable by camera. This method estimates the threshold based on a human contrast sensitivity function and modifies the pattern contrast for each frame, based on the brightness and spatial frequencies of the projected image and original pattern [Grundhöfer, Seeger et al. 2007].

The temporal-coded structured light approaches are not useful in dynamic environments as they require several projected frames to recover geometry of the target display surface, hence cannot easily be used with mobile display surfaces. With spatially-coded approaches it is possible to recover geometry with a single projected pattern, however, these approach create lower resolution geometry models and tend to be much less robust [Salvi, Pages et al. 2004]. Additionally, while the structured light approaches can detect and model all geometry inside the projector's field of view, they cannot robustly identify and segment an arbitrary 3D object from the background on their own. Hence, they would have to be used with other computer vision methods when we wanted to detect a particular object or project at a specific location on an object.

Instead, in our work we assume each object carries knowledge of its 3D model, which removes the need for on-line geometry recovery and allows projection at specific locations on the object. Once objects are detected and their pose calculated, the known surface geometry is used by the projector system to dynamically configure which calibration method is used. For example, with planar surfaces we could use the homography or projective texture mapping methods, and for curved surfaces the quadric calibration proposed by Raskar et al. [Raskar, VanBaar et al. 2005].

2.3.7 Projection Photometric and Colorimetric Calibration

We do not limit the type, form or appearance of objects that can be used with our Cooperative Augmentation approach. Hence, it is likely many surfaces of everyday objects we would like to augment with displays are coloured, textured or non-planar, which is not ideal for projection. Similarly, we may want to make use of multiple

projectors to create seamless displays on objects. In this section we look at different ways to address these cases by calibrating the projection to make the image more visible.

Photometric and colorimetric calibration is used in two scenarios: the first is when creating a perceptually seamless display using multiple projectors, generally with a normal white projection screen. With ad-hoc multi-projector displays main problem encountered is that the characteristics of the projectors vary. For example, projectors from different manufacturers differ in terms of their photometric response, colorimetric response and uniformity. Even among projectors of the same make and manufacturer the projector lamps vary in both brightness and colour as they age, as can be seen in the variable brightness and hue of the blue backgrounds in Figure 2.16 (left). To appear as a seamless, uniform display this necessitates calibration, however, the complexity of geometric alignment and colour calibration increases with the number of projectors.

Several approaches have been presented for automatic calibration of multi-projector systems using a single camera [Rajj and Pollefeys 2004; Brown, Majumder et al. 2005; Bhasker, Sinha et al. 2006]. Typically these involve measuring the luminance and chrominance properties within each projector's image and the differences between the projectors. These intra-projector and inter-projector variations are used to calculate individual mappings to a shared brightness and colour range which can be used by all projectors to achieve a seamless display. Brown et al. found this approach provides a poor image, as the common range for all projectors can be very narrow, limiting contrast and colour gamut. Humans can detect luminance variations more easily than chrominance variations, hence Brown et al. propose trading-off chrominance uniformity for increased display quality by approximating perceptual uniformity [Brown, Majumder et al. 2005].

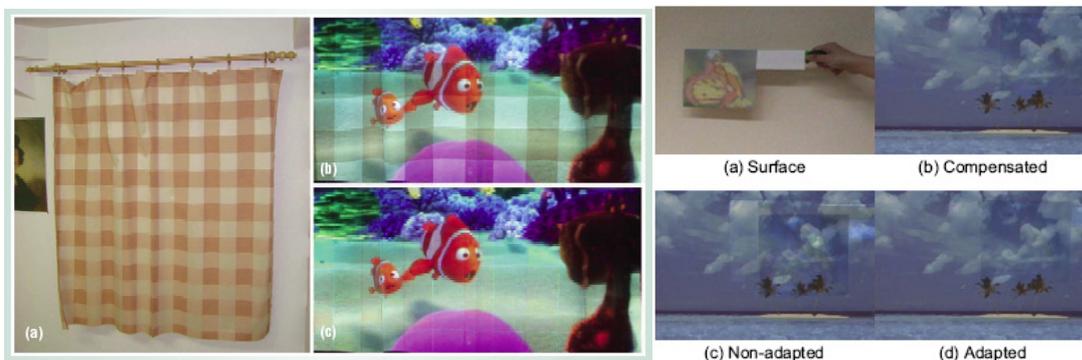


Figure 2.32 (left) Colour correction to project on any surface [Bimber, Emmerling et al. 2005], (right) Real-time adaptation for mobile objects [Fujii, Grossberg et al. 2005]

The second scenario where photometric and colorimetric calibration has been demonstrated is to compensate for the display surface itself in the case where the surface is non-white, non-uniform or has non-lambertian reflectance. Again, a range of techniques has been proposed [Nayar, Peri et al. 2003; Grossberg, Peri et al. 2004; Bimber, Emmerling et al. 2005]. The calibration process typically includes capturing the natural surface appearance and its response to projected colour calibration images with a camera. The calibration techniques then modify the projected image by attempting to invert the natural colour of the object surface based on the recovered reflectance responses, making it more visible to an observer (as shown in Figure 2.32).

The methods cannot completely correct very dark or saturated surfaces, as the dynamic range of typical projectors is not sufficient to invert the natural surface appearance.

While these calibration techniques assume a static environment, Fujii et al. extend [Nayar, Peri et al. 2003] by proposing a real-time adaptation approach [Fujii, Grossberg et al. 2005] by making use of the projector-camera system closed loop feedback. They use a 3×3 matrix for each pixel to encode both the colour mixing between the channels of projector and camera and the surface reflectance. The values of the matrices are determined by projecting a series of uniform colour images and capturing the images of the illuminated surfaces with the camera. The compensation image is computed by multiplying the inverse of the colour mixing matrices with the RGB colour of the corresponding pixels in the input image. Following initial calibration, the method corrects the projection image based just on the image captured by the camera, optimising the projected image to look as much like the input image as possible, when seen by the camera. Although requiring a camera and projector with calibrated colour response, this approach allows dynamic changes in illumination or colour of the projection surface and a mobile object or projector-camera system, as shown in Figure 2.32 (right).

Similarly, Grundhöfer and Bimber extend their work to achieve real-time correction by using the GPU for calculation, while simultaneously adjusting the content of the input images before correction to reduce the perceived visual artefacts from limited projector dynamic range [Grundhöfer and Bimber 2008].

The two scenarios presented using the calibration methods generally assume the display surfaces are planar and orthogonal to the projector. In our work this is often not the case, as objects are mobile and need not be planar. The amount of light that arrives at the projection surface depends both on the distance and angle of the surface with respect to the projector, with oblique surfaces receiving less illumination and appearing darker. Hence we also need to incorporate intensity compensation based on the orientation of the object for non-planar objects, before the radiometric and colorimetric calibration is applied.



Figure 2.33 (left and centre) Uneven illumination on a non-planar surface, (right) uncompensated and intensity compensated image [Park, Lee et al. 2006]

Bimber et al. propose a formalisation of this compensation based on square distance to the surface and the angle of the projected light [Bimber, Coriand et al. 2005], however, generally we do not care about the absolute brightness as long as the projection is uniform. Hence, in our work, a simplified method proposed by Park et al. [Park, Lee et al. 2006] can be used as we know the surface geometry and pose of an object relative to the projector.

2.3.8 Issues with Projector-based Augmented Reality

Projector-based AR faces issues which do not occur with other display technologies, for example, the display brightness must overcome the ambient illumination, and that

image warping for geometric correction reduces the final achieved display resolution. However, there are three main problems which occur when using projection:

2.3.8.1 Display Occlusion

The most obvious problem with front-projection is occlusion of the projection, either by other objects in the environment, or by an interacting user. It was observed both by Pinhanez et al. and Summet et al. [Pinhanez, Kjeldsen et al. 2002; Summet, Flagg et al. 2007] that users had to develop coping strategies for occlusions with front projection, moving their bodies and limbs to minimise occlusion.

Summet et al. propose a solution using multiple redundant projectors in the environment and an off-axis camera to detect occlusions of the display. The projectors are separated spatially but their frustums overlap so that the projected image from one projector is able to “fill-in” areas occluded in the second projector. In the active system [Summet, Flagg et al. 2007], the “Virtual Rear Projection” (VRP) method can also suppress the projected light on dynamically occluding users and objects, removing the blinding effect when users look towards the projector. While this approach works well when front projection is required, experiments reported that users still preferred rear projection systems. The solution also requires multiple projector systems, doubling the equipment costs for each display. Real-time correction was also demonstrated using the GPU for the calculations [Flagg, Summet et al. 2005].

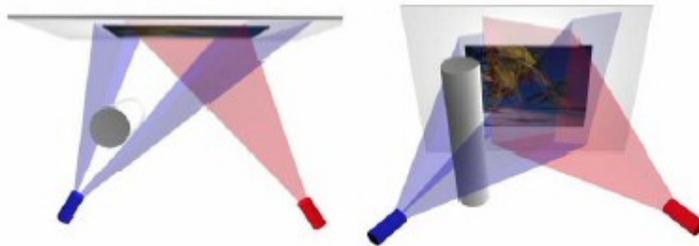


Figure 2.34 VRP used to overcome occluding light shadows [Summet, Flagg et al. 2007]

Both the VRP shadow removal techniques and multi-projector blending algorithms to blend of overlapped projections are only suited to static projectors and display surfaces. When projecting on mobile objects, any mis-registration between the projections (for example, due to error in object pose calculation or differing detection, rendering and projection lag times in multiple projector-camera systems) creates a blurred or unreadable projection.

Ehnes et al. propose another approach when using multiple steerable projectors and mobile objects [Ehnes, Hirota et al. 2005]. Similar to the VRP approach, multiple projectors have objects within their field of view simultaneously. However, instead of overlapped projection Ehnes et al. propose using a central application server, which assigns display rights based on the first projector-camera system to detect the object. A camera-reported quality measure (such as the distance and angle of the object surface with respect to the camera) is then used to determine when to change the display to another system [Ehnes and Hirose 2006].

This approach does not model occlusion directly, however, as the steerable projectors use an on-axis camera, any occlusion of the projection would also occlude the object in the camera, causing a low quality measurement. The display would then automatically switch to another projector with the object in view.

2.3.8.2 Projection Focus Distance

Typical projectors can only focus on a single focal plane located at a constant distance from the projector. Hence, projecting images onto non-planar surfaces, highly oblique surfaces or multiple surfaces at different distances causes blurring. If projectors possess computer controlled powered focus it is possible to calibrate the focus-distance relationship (as discussed in Appendix A.11) and dynamically focus on objects when their distance to the projector is known.

Recent work by Bimber et al. proposes using multiple projectors focused at different distances in static systems [Bimber and Emmerling 2006], however, future LASER projectors will not have this problem at all, appearing in focus at all distances.

2.3.8.3 Acceptability of Display

Although it is possible to project interactive interfaces onto objects in an environment, if users do not realise the objects and displays are interactive, then there is no benefit beyond a traditional display. At a demonstration of the Everywhere Display steerable projector Kjeldsen et al. found that users failed to recognise a paint can had an interactive projected button, despite having just successfully touched two similar buttons on a wall and table [Kjeldsen, Pinhanez et al. 2002]. To investigate this further, Podlaseck et al. performed a series of user studies to investigate the acceptability of projected interfaces on everyday objects [Podlaseck, Pinhanez et al. 2003]. These studies propose the possibility that people have difficulty cognitively perceiving high functional-fixedness objects as interfaces.

One illustrative example was a red virtual button projected onto the surface of a glass of milk, as seen in Figure 2.35 (left). At the end of the first experiment 10% of the subjects could not even remember seeing a glass of milk, which Podlaseck et al. believes was because they had mentally subtracted it from task at hand. Similar results were presented in a different context by Rensink as *inattentional blindness* (where observers closely watching a particular object fail to see other unexpected objects) [Rensink 2000]. They theorise that users may effectively be “blind” to interfaces on some objects. Consequently, some mechanism must be employed to increase their visibility and highlight an application’s connection to the object. This presents a challenge to the direct integration of information with objects in real-world, suggesting that the connection of an interface to an object may need to be made more salient and explicit.

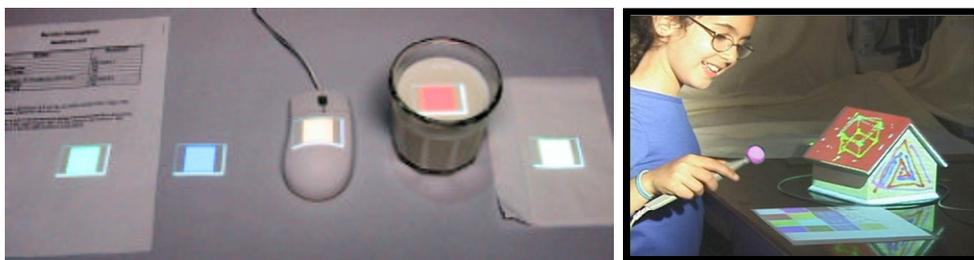


Figure 2.35 (left) Interactive buttons projected on a variety of everyday surfaces, (right) Dynamic Shader Lamps projecting on mobile tracked object [Bandyopadhyay, Raskar et al. 2001]

Weiser and Brown write “an affordance is a relationship between an object in the world and the intentions, perceptions and capabilities of a person [Weiser and Brown

1996]. The side of a door that only pushes out affords this action by offering a flat pushplate.” Similarly, hints to the user could be weaved into the display to suggest interaction modalities, such as a “touch me” label on a touch sensitive button. However, such explicit messages are at odds with the design goal of many interfaces aiming to be more natural and intuitive. Instead, subtle cues such as animation or interactive elements positioned specifically on objects close to a user could be used.

2.3.9 Mobile Objects

There has been much work on displaying projected content onto static areas of the environment [Pinhanez 2001], or static objects [Raskar, Welch et al. 1999; Butz, Schneider et al. 2004; Bimber, Coriand et al. 2005]. Similarly, there has been work on augmenting planar objects in very constrained scenarios such as “digital desk” scenarios [Wellner 1993; Robertson and Robinson 1999; Koike, Sato et al. 2001]. However, there has been relatively little work on augmenting mobile objects with projected displays in more unconstrained scenarios.

Morishma et al. first augmented mobile objects with projected displays [Morishima, Yotsukura et al. 2000]. Here pre-recorded videos of faces speaking were projected onto a planar mask worn by a live actor. The mask was tracked by a camera and a homography calibration used to warp the video to correct for geometrical distortion. The projector was located in a shopping trolley pushed by the actor; hence the possible range of movement of the actor was limited.

Increased working ranges have been demonstrated by using steerable projector-camera systems to track mobile objects, such as the Personal Interaction Panel discussed in section 2.3.4. Similarly, Ehnes et al. use visual markers to track objects [Ehnes, Hirota et al. 2004]. The use of a steerable projector allowed tracking to occur anywhere within its field of view and information or graphics to be projected at locations relative to the marker, as shown in Figure 2.39 (right).

Bandyopadhyay et al. investigated augmenting mobile objects with projected displays [Bandyopadhyay, Raskar et al. 2001]. Here 3D objects with planar surfaces were equipped with a magnetic and infra-red tracking system. Static projectors were used to augment the objects in real-time, as shown in Figure 2.35 (right). These projectors each required an initial calibration to match 3D points in the real world to 2D pixels in the projector image and hence, calculate the transformations between projector and tracking system’s coordinate system. A virtual 3D painting application was used to demonstrate the concepts of annotation and visualisation with projected displays. However, this work suffered from two key problems due to the tracking systems used. The first problem was that latency was around 110ms, which led to a 1cm distance lag even when the object was moved at very slow speeds. The Polhemus Fastrack magnetic tracker caused a second problem by limiting working range to 50-75cm.

In our approach we use a projector-camera system with a vision-based object detection system, allowing augmentation of objects anywhere within the field of view of the system at camera frame-rates, without relying on separate tracking hardware.

2.4 Detection and Tracking of Objects

To augment objects with projection, we must first calculate their location and orientation (i.e. their pose) with respect to the projector-camera system. There are many approaches to detecting and tracking camera or object pose in AR literature, for example, mechanical, magnetic, ultrasound, inertial, vision-based solutions, and hybrid systems combining different approaches [Azuma 1997; Rolland, Baillet et al. 2001]. Each approach has both advantages and disadvantages. For example, mechanical trackers are accurate but require a direct attachment, hence limit working volume. Magnetic trackers again have limited working volume and are additionally affected by metal objects in the environment. Ultrasound is susceptible to noise (for example, from jangling keys) and drift from temperature variations. Similarly, inertial systems (gyros and accelerometers) are expensive and again subject to drift over time [Lepetit and Fua 2005].

In contrast, while vision-based detection and tracking has a higher processing cost, it generally provides more accurate AR, as the pose is calculated directly from features in the image to be augmented. Vision-based detection and tracking in AR is split into fiducial marker-based and marker-less approaches. The marker-based approaches present a robust, low-cost solution to 3D pose estimation in real-time. They require no initialisation by hand, are robust to marker occlusion and provide a pose accurate enough for AR in good illumination conditions. However, they require changing the appearance of objects or engineering the scene to achieve this detection.

In contrast, while markerless approaches use the natural appearance of an object, they typically require an off-line training phase to learn the object appearance. The processing cost is also generally higher than marker-based approaches, as detection algorithms tend to be more complex due to the difficulty in achieving a robust detection.

2.5 Fiducial Marker Detection and Tracking

Fiducial marker based visual tracking approaches are widely used in the AR community, especially in experimental prototypes due to the availability of open-source tracking toolkits (such as ARToolkit [Kato and Billinghurst 1999] and ARTag [Fiala 2005]) allowing easy development. Many toolkits exist, typically each with their own marker appearance (such as rectangles, circles, semi-circles, and chessboard-like patterns), as shown in Figure 2.36 (left). Readers are referred to [Zhang, Frnz et al. 2002] for an in-depth comparison.

The fiducial markers themselves are printed planar patterns (similar to 2D barcodes) that a computer vision system can easily detect, track and decode, enabling information to be associated with the marker. Augmented Reality systems make use of this functionality to overlay computer generated information on top of detected markers using the detected marker pose for registration of the information with the real world. Traditionally a user wears a head mounted display (HMD), or carries a portable viewer to see the overlaid computer graphics, as shown in Figure 2.36 (right).

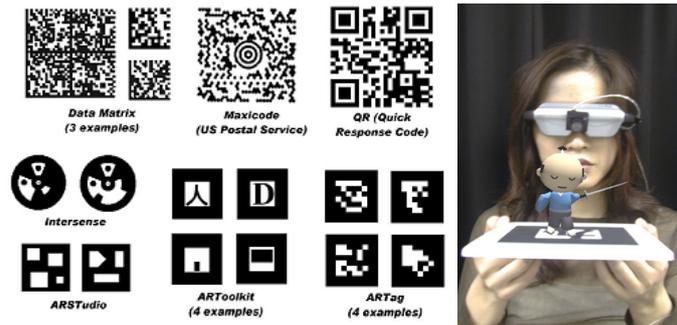


Figure 2.36 (left) Examples of different Fiducial Marker systems [Fiala 2005], (right) Using ARToolkit Markers with a HMD to view a virtual character [Kato and Billinghurst 1999]

The augmentation process for ARToolkit is shown in Figure 2.37. Other marker systems employ a conceptually similar process, but may detect the marker outline using edge based techniques for robustness, and decode information encoded in the marker rather than using cross-correlation to identify the marker [Fiala 2005].

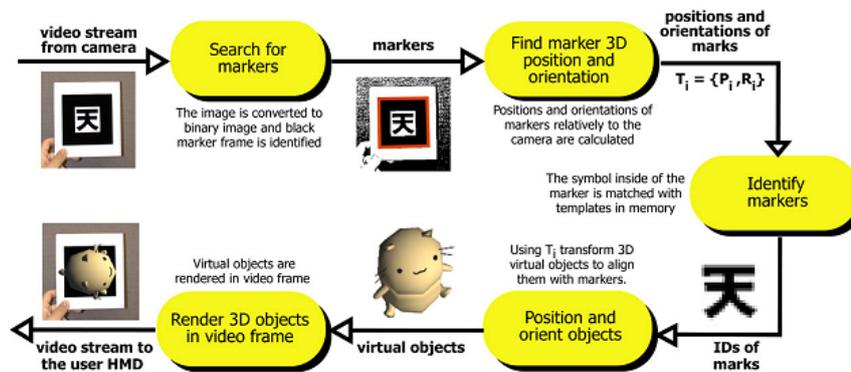


Figure 2.37 ARToolkit Recognition and Overlay method [Kato and Billinghurst 1999]

Unfortunately, fiducial marker approaches exhibit six main limitations:

1. **Minimum Marker Size in Camera Images.** The distance from the camera at which a marker can be detected decreases with both decreasing marker size and decreasing camera angular resolution. Hence, small markers, low resolution cameras or wide Field-Of-View (FOV) cameras will typically have a small tracking range. Fiala stated that for ARToolkit markers, a 75% marker recognition accuracy required a minimum marker edge size of 13 pixels in the image with a greyscale VGA camera and 18 pixels with a colour camera [Fiala 2005].
2. **Poor Location Accuracy.** Malbezin et al. characterised the accuracy of ARToolkit marker tracking with a calibrated camera in room sized environments, between 1m and 2.5m distance from the camera [Malbezin, Piekarski et al. 2002]. It was found that the reported position error increased with distance, from 6 to 12% in the marker X axis and 9 to 18% in the marker Y axis, with the accuracy being further affected by the angle of the marker to the camera. They conclude that once calibrated, any systemic inaccuracy could be corrected by applying a filter to the detection results; however, the filter would require explicit calibration for every camera and lens combination used.
3. **Limited Number of Useful Marker.** Some fiducial marker toolkits encode an ID to differentiate between different markers (e.g. ARTag toolkit [Fiala 2005]). The limit

on the number of markers that can be detected is the number of visually distinct markers that can exist with the encoding pattern used. For example, ARTag has a library of 2002 rectangular markers with an encoded 11 bit unique numerical identity (ID). To increase the number the number of unique objects that can exist in the world the number of bits allocated to the ID (and hence the physical marker size) must be increased. Similarly, circular marker systems such as TRIP [Ipiña, Mendonça et al. 2002] are generally limited only by their physical size – extra rings with more bits can always be added to the outside of the marker.

ARToolkit takes a different approach, with no fixed encoding scheme. This allows users to generate unique and visually distinct patterns inside the rectangular black border used for initial detection. However, the user must then pre-calibrate the system and explicitly specify which patterns the system should load and search for at runtime.

4. **Marker Occlusion.** Different marker toolkits exhibit different robustness to partial occlusion of the markers [Fiala 2005]. For example, as shown in Figure 2.38 (left and centre), ARToolkit detection will fail if even a small portion of the external marker border is obscured. In contrast, ARTag detection remains robust as long as three sides remain visible. This issue with ARToolkit can be used as a feature in multi-marker tracking scenarios, allowing failure in detection for one marker to be inferred as deliberate user interaction, for example, finger activation of a virtual button.

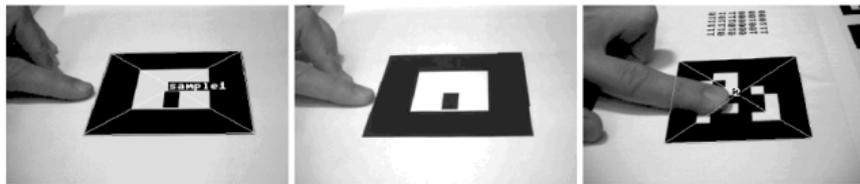


Figure 2.38 (left and centre) Occlusion of ARToolkit marker prevents tracking, (right) Occlusion of ARTag marker [Fiala 2005]

5. **Marker Illumination.** As shown in Figure 2.39 (left), some AR toolkits lack robustness under varying illumination conditions due to use of a global image threshold [Naimark and Foxlin 2002]. Solutions to this include local or adaptive thresholding, or the use of more robust marker systems, such as ARTag, which use edge detection methods [Fiala 2005]. This is a serious issue for our work, as any projector light overlapping a marker may prevent detection.
6. **Visual Intrusiveness.** As adding fiducial markers on objects is visually intrusive, Park and Park propose using Infra Red (IR) absorbing markers [Park and Park 2004]. These markers appear black to an IR camera in environments illuminated with IR light, but are invisible to the human eye. Similarly, Nakazato et al. and Santos et. al demonstrate IR retro-reflective (but visually translucent) markers [Nakazato, Kanabara et al. 2004; Santos, Stork et al. 2006]. While such techniques remove visual intrusion, they still share the limitations of visual markers and additionally rely on the presence of IR light sources for detection.



Figure 2.39 (left) AR Toolkit fails to detect markers under different illumination conditions in the same frame [Fiala 2005], (right) Ehnes et al. track ARToolkit markers with a steerable projector and project on modelled surfaces in the marker area [Ehnes, Hirota et al. 2004]

With a view to augmentation in ubiquitous computing environments the marker based approaches are not ideal, as they require physical modification or engineering an objects external appearance to enable detection. To achieve robust detection at distance requires a large marker, which in turn reduces our available projection area, as illumination change on the marker due to projection may prevent detection. Additionally, use of markers is problematic for objects with non-planar surfaces, and 3D objects typically require multiple markers (e.g. a cube requires at least one marker per surface = 6) to achieve detection in all poses. Hence, our approach in this work uses the natural appearance of an object for markerless detection, allowing augmentation with displays without permanently changing an object's appearance.

2.6 Natural Appearance Detection and Tracking

The use of natural appearance for marker-less detection in AR is generally a wide-baseline matching problem, where natural features in an unknown scene have to be matched to a model of the object pre-built from training images, despite changing viewpoint and illumination conditions. In this work we consider only monocular (i.e. single camera) model-based detection and tracking approaches. We also distinguish between detection and tracking. Detection is identifying a known object in an unknown scene, whereas tracking is typically limited to following a previously detected object in an unknown scene. Tracking algorithms either require an initial detection step, or require that the object to be tracked is manually manipulated into a pose close to the starting pose assumed by the tracker. In this thesis we concentrate on object detection, however to understand the difference more clearly we first look briefly at tracking approaches before investigating detection in more detail.

The reader is directed to [Lepetit and Fua 2005] for a full survey of 3D model-based detection and tracking.

2.6.1 Tracking

Recursive tracking algorithms only face a narrow-baseline matching problem, as they typically just consider the last few camera frames to predict the current pose. Over such small timescales any object or camera movement is generally small and object appearance is not likely to change. However, object or camera occlusion, fast motion, or appearance and illumination changes between consecutive frames can cause loss of tracking, requiring re-initialisation either by another detection step, or manual re-alignment. Recursive tracking in general is also susceptible to error accumulation

[Lepetit and Fua 2005]. The advantages of tracking approaches are that a predictor-corrector framework constrains the search for matching features between frames based on their recent location, reducing image processing requirements and hence allowing faster, more real-time performance.

The two most common frameworks on which trackers are based are the Kalman filter [Kalman 1960] and Particle filter [Isard and Blake 1998]. Both these frameworks use a Bayesian formulation to estimate the probability density of successive poses in the space of all possible camera or object poses. Kalman filters only consider a Gaussian distribution, whereas Particle Filters use a more general representation with a set of weighted pose hypotheses. Each particle is an individual hypothesis, allowing multiple hypotheses to be supported simultaneously. However, to describe complex motion a large set of particles is required, with consequent high processing cost. Particle filters are also more prone to jitter in the predicted pose, which can be smoothed using a filter [Isard and Blake 1998].

Objects with strong edge contours also allow use of edge tracking methods such as the popular RAPID algorithm [Harris 1993]. RAPID based algorithms generally project lines from the object's known 3D model into camera image coordinates, based on an estimate of the object's pose, as shown in Figure 2.40 (a). At intervals along visible lines a number of control points are generated (b). A 1D search is performed at each control point for edges in the camera image, perpendicular to the projected lines (c). The 3D motion of the object between consecutive frames can be calculated from the 2D displacement of the control points and used to predict both the pose in the next frame and hence, which control points will be visible [Drummond and Cipolla 2002; Klein and Drummond 2003].

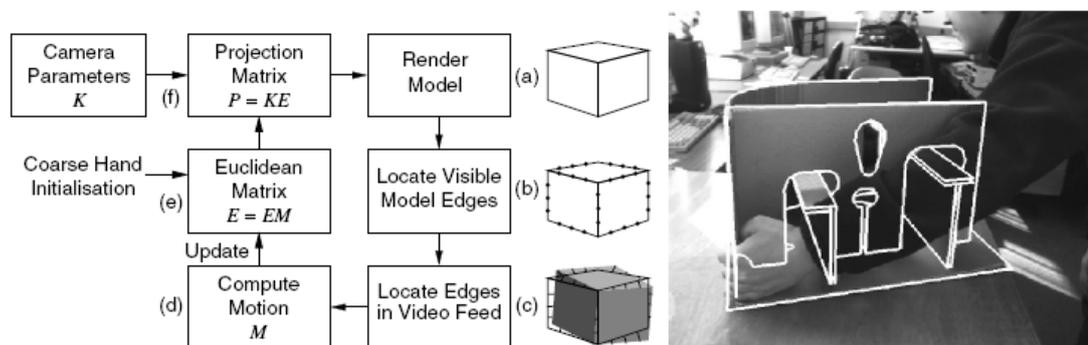


Figure 2.40 (left) Drummond and Cipolla's RAPID-like algorithm, (right) robustness to partial occlusion [Drummond and Cipolla 2002]

The main problem with algorithms based on RAPID is their lack of robustness; as incorrectly matched edges from occlusion, shadow, object texture or background clutter cause incorrect pose computation. Hence, a number of extensions have been proposed to make RAPID more robust, such as grouping control points to form primitives (such as a line or quadrilateral), the use of robust estimators such as RANSAC to detect outliers or integration of RAPID into a Kalman filter [Lepetit and Fua 2005].

Approaches based on extracting and matching line segments to a 3D model are also possible, trading-off generality for robustness [Deriche and Faugeras 1990; Lowe 1992; Koller, Danilidis et al. 1993]. However, the edge based methods are generally fast, simple, robust to changing illumination and object scale. Their main problem is failure due to mismatching when backgrounds become cluttered or when rapid change in pose occurs.

2.6.2 Detection

There are several challenges when detecting objects using their natural appearance. For example, objects themselves have widely differing appearances, and we need to detect them over a large scale range and with the object in any pose. Many approaches have been proposed using cues of an objects appearance, such as colour, texture, shape and features on the object. A selection of popular approaches to detecting these cues are described below:

2.6.2.1 Colour

Colour is a powerful cue for humans, used in many ways everyday in the real-world; for example, in traffic lights, as identifying features in man-made products, in advertisements or warning signs. In computer vision, colour histograms, first proposed by Swain and Ballard [Swain and Ballard 1991], have been shown to be invariant to rotation and robust to appearance changes such as viewpoint changes, scale and partial occlusion and even shape. Hence, a 3D object can be represented using a small number of histograms, corresponding to a set of canonical views (Swain and Ballard recommend 6 views). However, in their original work Swain and Ballard reported the need for a sparse colour distribution in the histogram to distinguish different objects, which can be achieved using a high dimensional histogram. Similarly, colour histograms are illumination variant, so illumination intensity, temperature and colour will all affect the final histogram. In contrast, histogram matching techniques are generally robust, as the histogram representation uses the entire appearance of the object rather than just a small number of interest points.

Histograms are created by dividing a colour-space into discrete units (bins) and filling those bins with each pixel of that colour from the source image. The result will be a set of bins which represent the approximate colour distribution in the image. Histograms can be multi-dimensional (e.g. 3D Red-Green-Blue histograms) and bins can be larger than a single value (e.g. for an 8-bit RGB colour image each bin could be 16 colour values wide, giving a 16x16x16 bin histogram).

As most objects have surfaces composed of regions of similar colour, peaks will be formed in the histogram. Objects can be detected by matching a colour histogram from a camera image region to a histogram from a training sample of the object using either the histogram intersection measurement (for two histograms V and Q):

$$\cap(Q, V) = \sum_i \min(q_i, v_i) \quad (2.4)$$

or statistical divergence measurements, such as chi-square (χ^2):

$$\chi^2(Q, V) = \sum_i \frac{(q_i - v_i)^2}{q_i + v_i} \quad (2.5)$$

With histograms a high match value will still be obtained, even if individual pixel colour matches are not exact, because the regions are well matched.

Swain and Ballard also propose histogram back-projection as a way of locating an object in an image. Here the colour values in the source image are replaced by matching values from a ratio histogram calculated between the object model and image histogram. Swain and Ballard then propose convolving the result image with a mask of

the estimated size of the object and extracting maxima which correspond to an expected location.

2.6.2.2 Texture

Many objects cannot be described by colour alone (for example, black objects), hence objects with visible texture on their surfaces allow a wider range of techniques to be employed. For example, template matching approaches are fast and can be easily used to track 3D objects with planar surfaces (assuming a calibrated camera), although this method is susceptible to failure with occlusion and must be explicitly trained with varying illumination to become robust to illumination change [Jurie and Dhome 2002].

The histogram approach of Swain and Ballard was generalised to multidimensional histograms of receptive fields by Schiele and Crowley to detect object texture [Schiele and Crowley 2000]. The histograms encode a statistical representation of the appearance of objects based on vectors of joint statistics of local neighbourhood operators such as image intensity Gaussian derivatives (D_x, D_y) or gradient magnitude and the local response of the Laplacian operator (Mag-Lap), as shown in Figure 2.41. Multidimensional histograms are used to provide a reliable estimate of the probability density function without being computational expensive.

Experimental results show the histograms are robust to partial occlusion of the object and are able to recognise multiple objects in cluttered scenes in real-time using the probabilistic local-appearance hashing approach proposed by Schiele and Crowley. When matching, Schiele and Crowley found Chi-squared divergence function provides better detection results than intersection, with respect to appearance changes, additive Gaussian noise and blur [Schiele and Crowley 2000].

Another approach to presented by Schaffalitzky and Zisserman develops a region descriptor for texture detection, using a class of statistical descriptors which is invariant to affine viewpoint and photometric transformations [Schaffalitzky and Zisserman 2001]. Their method was demonstrated directly in wide-baseline matching to calculate the epipolar geometry between two views despite significant changes in viewpoint; however, their work makes the assumption that all regions are planar.

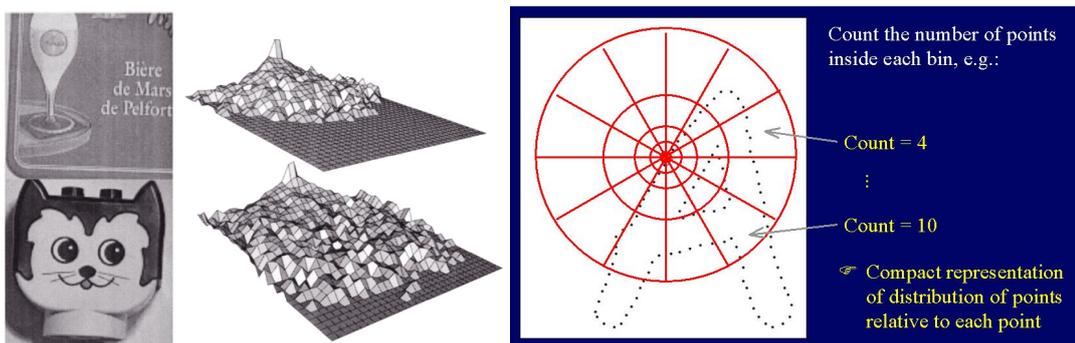


Figure 2.41 (left) Two dissimilar objects and their Mag-Lap histograms corresponding to a particular viewpoint, image plane rotation and scale. [Schiele and Crowley 2000], (right) Shape Context 2D log-polar histogram based on relative point locations [Belongie, Malik et al. 2002]

2.6.2.3 Shape

Object shape can be described using either a global or local shape description. For global shape typical approaches use PCA-based methods [Turk and Pentland 1991;

Murase and Nayar 1995]. Here a global eigenspace is built as training images are projected into it. PCA analysis reduces the dimensionality of the training image set, leaving only those features in the images that are critical for detection. To train the representation typically a set of images of each object in different poses with varied lighting (to make it illumination invariant) are used. This image set is compressed to a low-dimensional sub-space by computing eigenvectors and eigenvalues on the covariance matrix of the training images and the highest eigenvectors kept. These eigenvectors represent a manifold. When detecting an object in an unknown image, the image is projected into the eigenspace and recognised based on the manifold it lies on. The pose is determined by the location of the projection on the manifold [Murase and Nayar 1995]. PCA based methods require the image projected to the eigenspace to be the same size as the training images, hence a scale-space must be created for scale-invariant detection [Lindeberg 1990]. Turk and Pentland also propose using a 2D gaussian centred on the training and test images to reduce image intensity at the borders, as cluttered backgrounds negatively affect the detection [Turk and Pentland 1991].

Local shape can be detected using many methods. Here we look at methods that describe the silhouette contours of an object, as these can be directly matched to a pre-computed database of object appearances with the object in different poses. For objects with known exact 3D models this database of object appearances can be calculated directly by rendering the model in different poses and extracting the silhouette contour using an edge detection algorithm, such as the Canny algorithm [Canny 1986].

The Curvature Scale Space (CSS) algorithm is part of the MPEG7 standard, for use in 2D shape detection [F.Mokhtarian 1995]. CSS is a “multi-scale organization of the inflection points of a 2D contour in an image”, i.e. zero-crossing points on the contour are detected when the contour changes direction. The multi-scale segmentation renders the system robust to edge noise and local shape differences. This algorithm has been demonstrated in 3D object detection [F.Mokhtarian, Khalili et al. 2001] and because the silhouettes of objects will be very similar with only small changes in pose, Mokhtarian and Abbasi also proposed a way of determining how many unique training views are required in an appearance database to maximise detection rates [F.Mokhtarian and Abbasi 2005]. This allows objects with complex and non-rotationally symmetric geometries to estimate pose by direct matching with CSS to the appearance database.

The Shape Context method described by Belongie et al. [Belongie, Malik et al. 2001; Belongie, Malik et al. 2002] is related to deformable templates [Aixut, Meneses et al. 2003; Felzenszwalb 2005] and has achieved success in character recognition. The contours of an object are first extracted using an edge detection algorithm and a set of points detected on the contours, either equally or randomly spaced. The shape context algorithm then captures the relative distribution of points in the contours relative to each point on the shape. As shown in Figure 2.41, a histogram of the number of contour points in each “bin” of the log-polar coordinates can be created to describe each point. Descriptors are similar for homologous (corresponding) points and dissimilar for non-homologous points, hence objects can be detected by matching the log-polar histograms and generating point correspondences. As the number of points in training and test images is identical, the matching process is a linear assignment problem, with the goal of assigning the best match to each point. Here, the histograms are used to calculate and minimise the cost for each match. Belongie et al. then align training and test images using a thin-plate-spline method of point-set alignment to model the deformation required to align the two images.

2.6.2.4 Features

Another approach using the natural appearance of an object is local features. Local feature based detection algorithms aim to uniquely describe (and hence detect) an object using just a few key points. By extracting a set of interest points such as corners or blobs from training images of an object we can use the local image area immediately surrounding the interest point to calculate a feature vector which we assume serves to uniquely describe and identify that point. A database of local features registered to a 3D model of the object is constructed off-line. At runtime interest points are detected in the camera image. Object detection now becomes a problem of matching features between the training set and camera image by comparing feature descriptors. Once 2D image to 3D model correspondences are established they can be directly used to calculate the object's 3D pose. Readers are referred to [Mikolajczyk and Schmid 2005; Mikolajczyk, Tuytelaars et al. 2005] for an in-depth comparison of different feature detection and descriptor algorithms.

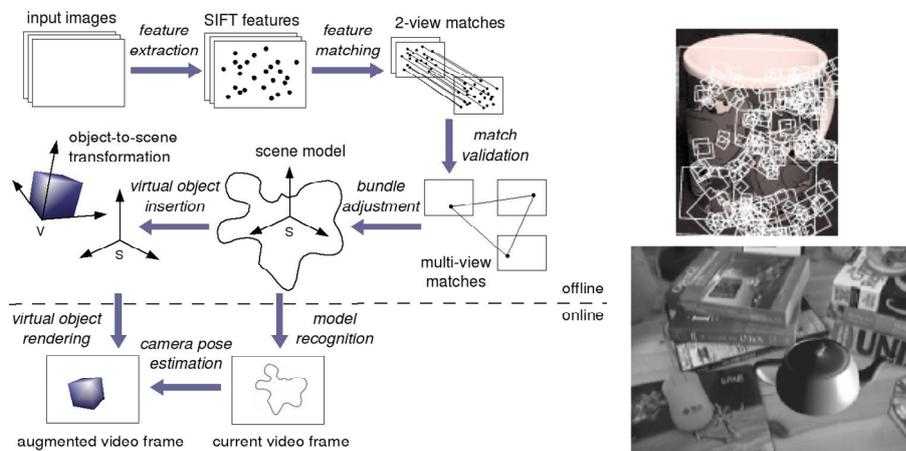


Figure 2.42 (left) SIFT local feature based AR method, (right, top) SIFT Features detected on a mug, (right, bottom) AR teapot added to camera display [Gordon and Lowe 2004]

Local features are demonstrated for marker-less real-time object detection and tracking [Gordon and Lowe 2004; Gordon and Lowe 2006]. Here Gordon and Lowe use an off-line metric model-building phase to initially acquire scene geometry with SIFT scale and rotation-invariant features [Lowe 2004]. The 3D feature model can then be used on-line for near real-time detection and tracking with local features for AR, as show in Figure 2.42.

Affine viewpoint-transform invariant features which deform their shape to the local region orientation have also been proposed [Matas, Chum et al. 2002; Mikolajczyk and Schmid 2002]. The geometry and intensity based affine region trackers proposed are also invariant to linear changes in the illumination, increasing robustness for real-world environments. However, the number of actual features detected is less as only invariant features are kept and (due to their invariance) these features are also less discriminative, hence and the possibility of mis-matches increases.

2.6.3 Multi-Cue Detection and Tracking

As objects vary significantly in their appearance, approaches based on a single cue such as just colour or shape can perform poorly in real-world environments. No single visual cue is general enough but also robust enough to cope with all possible

combinations of object appearance and environment, hence, multi-cue detection systems have been proposed. The goal of multi-cue approaches is to increase robustness for detection and tracking in dynamic environments. In theory, a combination of complementary cues leads to an enlarged working domain, while a combination of redundant cues leads to an increased reliability in detection [Spengler and Schiele 2001].

Popular cue combinations are edges and vertices of the 3D model [Hirose and Saito 2005], edges and texture [Vacchetti, Lepetit et al. 2004], colour and edges [Li and Chaumette 2004], colour and texture [Brasnett, Mihaylova et al. 2005], intensity, shape and colour [Spengler and Schiele 2001], shape, texture and depth [Giebel, Gavrilu et al. 2004]. However, these combinations are fixed at runtime.

Extended or unscented Kalman filters have been widely used for multi-sensor fusion [Welch and Bishop 1997; You and Neumann 2001; Foxlin and Naimark 2003]; however, Kalman filters require a good measurement model. As objects and cameras can be mobile or handheld in AR, it is difficult to define a model suitable for such unpredictable motion [VanRhijn and Mulder 2005].

Typically, multiple detection cues are fused in particle filter tracking frameworks, allowing multiple target hypotheses to exist simultaneously [Spengler and Schiele 2001; Giebel, Gavrilu et al. 2004; Li and Chaumette 2004; Brasnett, Mihaylova et al. 2005]. In this case the detection of each cue is often assumed to be independent and they are democratically integrated to contribute to the overall measurement density. Hence, when different image cues are fused simultaneously, the failure of one cue will not affect the others [Li and Chaumette 2004].

Several other methods have been proposed in multi-cue tracking for AR, for example using edges and vertices of the 3D model [Hirose and Saito 2005]. Similarly, Vacchetti et al., use edges of the 3D model and texture [Vacchetti, Lepetit et al. 2004]. Here, they extend the RAPiD tracker to consider multiple contour candidates instead of just the closest, to solve edge ambiguities. The correct match is then selected during pose optimization using a robust estimator. Texture is detected as Harris corners [Harris and Stephens 1988] on the surface of the object and both are fused using the method described [Vacchetti and Lepetit 2004]. This fusion makes both the registration accuracy more stable and allows the system to handle textured and un-textured objects, however, it does not take into account rapid camera or object movement.

In our approach we also use multiple detection algorithms to detect smart objects, but cues are chosen at runtime by a selection step based both on the appearance knowledge contained within a smart object and the object's context (such as the background).

2.6.4 Camera Model

Understanding how cameras and projectors are modelled is important to understand how an object's pose is calculated and how projectors and cameras are calibrated in this thesis.

In this work we use a standard pinhole camera model for both projectors and cameras, formulating the projection of light rays between 3D space and the image plane of the camera or projector. We assume this projection is a perspective projection. As can be seen in Figure 2.43 (left), the camera image plane (where the projection of 3D points is formed) is modelled as a 2D plane in the X,Y axes with the Z axis towards the object. $P(x,y)$ is the principle point, where the Z axis (representing the optical axis of the camera) intersects the image plane.

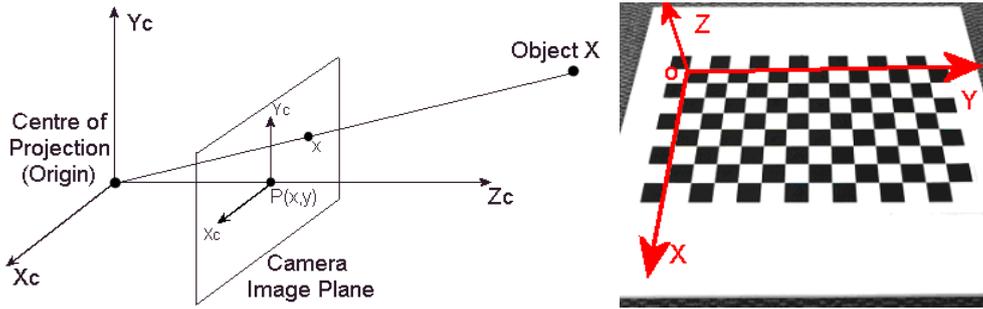


Figure 2.43 (left) Pinhole Perspective Camera Model, (right) Checkerboard pattern for camera calibration with overlaid pattern coordinate system

The 3D coordinates of Object X are defined as $[X,Y,Z]^T$ and the corresponding projection on the camera image plane (x) is $[x_c,y_c]^T$. These are related by the equation $sx = PX$, where s is a scale factor and P is a 3×4 projection matrix defined up to scale.

The projection matrix P can be decomposed as:

$$P = K[R|t] \quad (2.6)$$

where K is a 3×3 matrix of the intrinsic optical parameters of the camera (the camera calibration matrix) and $[R|t]$ is a 3×4 matrix of extrinsic parameters (R represents a 3×3 rotation matrix and t a translation) defining the transformation of the object (or camera) from the world coordinate system to camera coordinate system [Lepetit and Fua 2005].

$$K = \begin{bmatrix} f_x & skew & P_x \\ 0 & f_y & P_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.7)$$

The intrinsic parameter matrix (K) is composed of f_x, f_y which are the focal lengths of the lens in the X and Y axes respectively. These focal lengths are determined in terms of pixels per unit distance in the respective axes. P_x, P_y are the pixel coordinates of the principal point in the camera image. We assume rectangular pixels, hence $skew = 0$.

In this work we use the Zhang's camera calibration method [Zhang 2000] to recover the camera intrinsic parameter matrix (K) from images captured of a planar checkerboard pattern in different orientations and at different distances to the camera, as shown in Figure 2.43 (right). Additionally, as a camera lens is not optically perfect, we assume a second order lens distortion model with radial and tangential distortion [Hartley and Zisserman 2003]. Lens distortion is estimated at the same time as calibration with Zhang's method. All camera images captured in this work are initially corrected for lens distortion before use.

Readers are referred to [Hartley and Zisserman 2003] for a full discussion of camera calibration.

2.6.5 Pose calculation

As an object moves around an environment its appearance changes depending on its relative orientation and distance to the camera. As we are interested in detection of objects in unconstrained real-world scenarios we aim to recover the full 6 degree-of-freedom 3D location and orientation of the object relative to the camera when it is in any pose, or at any scale.

Pose Calculation involves calculating the transformation which allows the best fit of an object's 3D model to features detected in a 2D camera image. This is equivalent to calculating $[R|t]$, the extrinsic parameters of the object (or camera). Many different approaches have been proposed, but most approaches require correspondences to already have been established between the image and 3D model before the pose calculation step. Some of the most common methods are described below:

A 3D object pose can be calculated directly from correspondences using a Direct Linear Transform (DLT). This method is similar to the homography projector calibration process described in section 2.3.6, however, the formulation is now a linear system of equations for 3D-to-2D projection. A unique solution can be obtained to the equations using Singular Value Decomposition (SVD) when the intrinsic parameters are known and there are 6 or more correspondences. However, as the DLT algorithm minimises algebraic error in its solution, the calculated pose may not be the best geometric fit of the model. Instead, we can re-formulate the equations to give us the minimum re-projection error between the 3D points and their 2D coordinates:

$$[R|t] = \arg \min_{[R|t]} \sum_i dist^2(PX_i, x_i) \quad (2.8)$$

This can now be solved using linear least squares minimisation techniques or non-linear iterative approaches [Lepetit and Fua 2005]. The iterative approaches such as Gauss-Newton and Levenberg-Marquardt can also be used as a way to improve accuracy following an initial pose calculation such as DLT, or when we have an initial estimate of the pose (for example, from other non-vision-based sensors).

As any incorrect correspondences in the pose calculation process causes errors in the calculated pose, a robust estimator such as RANSAC [Fischler and Bolles 1981] or an M-estimator is typically used in model-based tracking. The two approaches are complimentary, with M-estimators producing accurate solutions but requiring an initial estimate, while RANSAC does not requiring an estimate but typically uses only a subset of all correspondences. RANSAC itself is a simple iterative algorithm, randomly extracting the smallest set of correspondences required to calculate a pose (Fischler and Bolles use 3 correspondences). After pose calculation the algorithm projects all 3D model points and measures the re-projection error to the corresponding 2D points detected in the camera image. If the points are projected close enough the corresponding points are treated as inliers. RANSAC finally returns the pose with the largest number of inliers, theoretically eliminating incorrectly matched correspondences.

DeMenthon and Davis' POSIT algorithm first calculates an approximate pose by assuming the camera model is a scaled orthographic projection [DeMenthon and L.S. Davis 1995]. With a simpler camera model there are less unknown components of the projection, hence with more than 4 corresponding points a pose can be calculated by

solving a set of linear equations. After the initial estimate each point is weighted based on the re-projection error (scaling the coordinates) and the process is iterated until it converges. The POSIT algorithm does not work when the model is planar; hence, Oberkampff et al. extend the original algorithm for the coplanar case [Oberkampff, DeMenthon et al. 1996]. As incorrect correspondences in the pose calculation process cause an incorrect pose to be calculated, in more recent work David et al. propose combining the POSIT algorithm with a softassign approach [Gold, Rangarajan et al. 1998]. This allows simultaneous determination of pose and correspondence with the softPOSIT method for points [David, DeMenthon et al. 2002], lines [David, DeMenthon et al. 2003] and line features when there are high-clutter backgrounds [David and DeMenthon 2005].

In contrast, voting approaches such as the Generalised Hough Transform (GHT) only represent object structure implicitly for efficient matching, avoiding the need for costly methods to group features for robust pose calculation (such as RANSAC). Instead, matching feature pairs (such as 3D model lines and image edges) are converted into votes for a rigid transformation which would align the corresponding features, assuming the matches are correct. Votes from multiple matches for a consistent transformation make peaks in a voting histogram; hence the transform is robust to outliers. This approach is used with Lowe's SIFT local features as an initial clustering step to reduce outliers before RANSAC [Brown and Lowe 2002; Lowe 2004].

Geometric hashing approaches use a similar approach, but use a hash table instead of a multi-dimensional voting space for performance reasons [Wolfson 1990; Lamdan and Wolfson 1998]. Hence, while the GHT quantises all possible transformations between a model and object into its bins, geometric hashing quantises only a set of discrete transformations represented by the hash basis.

In our work we first establish correspondences between features such as interest points or lines in the camera image and the object's 3D model, then use a robust approach combining DLT and RANSAC for initial pose estimation and a small number of Gauss-Newton iterations to refine the pose and increase accuracy.

Readers are referred to Hartley and Zisserman for a complete introduction to pose calculation [Hartley and Zisserman 2003].

2.6.5.1 Pose Jitter

One of the major problems with pose calculation is jitter, which can arise even with a static camera and object due to two main factors:

The first is that mismatched features or noise in the image affects the calculation, so when calculating pose from a few correspondences even small differences in the location of detected features in the camera image can have a large effect on the overall pose calculation. Robust matching techniques such as RANSAC can be used to reject outliers with large errors when the solutions are over-constrained with many correspondences, however, they can still include features with small reprojection errors.

The second factor is that numerical methods calculating a Perspective transformation from n Points (PnP) have multiple valid solutions for small numbers of correspondences. For example, the original RANSAC algorithm uses 3 points, which has up to 4 possible solutions. The results can usually be constrained, for example, by ensuring the calculated pose places the object in front of the camera rather than behind. However, this problem cannot be entirely eliminated in the case where ambiguous data

causes a model to fit in multiple valid ways. One example for humans is the popular Necker Cube optical illusion, where with certain line or corner arrangements a simple cube appears inverted. In vision systems this can lead to flipping between two poses.

Both these factors lead to jitter, which can either be smoothed using a motion model, minimised either using an approach such as keyframes discussed in section 2.6.6, or by employing stronger constraints. One solution using stronger constraints is registration with planar structures such as polygons or circles in the scene. This approach can be useful as many planar surfaces exist in everyday environments. For example, Ferrari et al. track a planar surface under affine transformations and overlay virtual textures [Ferrari, Tuytelaars et al. 2001].

2.6.6 Hybrid Detection

To trade-off the advantages and disadvantages of both the marker-based and markerless approaches hybrid detection approaches have been proposed. Genc et al. first proposed a learning-based approach, where markers are initially used to train a detection system while corners are extracted from the scene [Genc, Riedel et al. 2002]. A bundle adjustment is then performed off-line to reconstruct object geometry. This model is then used for tracking with a robust corner matching algorithm.

Similarly, Bougeois et al. use pose information from initial marker detection to remove the requirements for hand initialisation of a 3D model-based tracker for successive frames [Bougeois, Martinsson et al. 2005].

Recently, as computer processing power has increased a real-time tracking by detection approach has become feasible in markerless tracking approaches [Lepetit and Fua 2005]. However, as illustrated by Lepetit and Fua, detection in each frame independently has three main problems: reduced accuracy, increased jitter in the recovered pose and increased processing requirements over narrow-baseline matching. Consequently, imposing temporal continuity constraints across frames can help increase the robustness and quality of the results. For example, Vacchetti et al. propose a hybrid algorithm, combining a small number of key-frame models (generated from training images of the object in known pose) together with a real-time bundle adjustment algorithm [Vacchetti and Lepetit 2004]. This formulates the tracking as both wide-baseline matching to the key-frames and narrow-baseline matching to previous frames. The hybrid detection-tracking approach copes with large viewpoint changes due to the key-frames, while reducing pose jitter and drift.

Hybrid methods using vision detection with physical sensors have also been proposed. These are discussed further in section 2.7.

2.7 Vision-Based Detection with Physical Sensing

There are many failure modes for vision-based detection: giving false positives where no object exists, incorrect classification, or failing to detect an object. Common reasons for failure can be classified as: environment-related failures (such as significant changes in the illumination), motion-related failures (such as fast movement of the object or camera causing blurring of the image), distraction related failures (for example, where multiple objects with identical appearance are present), or occlusion-related failures where objects or environment partly or fully occlude object we want to detect.

To address these problems, researchers have used sensing in combination with vision-based detection. Most related work falls into two categories:

1. Sensing for Pose and Object Motion Prediction
2. Structured Light Sensing for Location and Pose

2.7.1 Sensing for Pose and Object Motion Prediction

Vision-based detection and pose calculation faces the problem that any rapid motion of the camera or object during the camera exposure generally causes blur. This motion blur changes an object's appearance; hence, can easily cause a loss of tracking.

Hybrid vision-inertial detection systems are often used to overcome this problem. For example, Klein and Drummond [Klein and Drummond 2003], You et al. [You, Neumann et al. 1999] and Chandraker et al. [Chandraker, Stock et al. 2003] all use inertial sensing to predict object motion when an object is not detected visually. Here, the vision and sensing approaches attempt to compensate for the limitations of the other technology, with vision to correct drift in the inertial system, and the inertial system to compensate for blur or occlusion in the camera image. Aron et al. use a similar approach [Aron, Simon et al. 2004], but here inertial sensing is used directly for guided local feature matching. In contrast, Kotake et al. present a different approach, using only an inclinometer sensor in the camera to constrain the detection and pose calculation process [Kotake, Satoh et al. 2005; Kotake, Satoh et al. 2007].

Combined vision-based and optical sensing methods have also been used. Klein et al. track a tablet using the edges in its 3D model and IR LED markers [Klein and Drummond 2004]. The LED are detected with a separate high-speed fixed camera, allowing detection and pose calculation even under large or fast movements which cause the edge features to blur and vision-based tracking to fail. However, in common with many hybrid systems using outside-in trackers, their system requires an additional offline calibration procedure to calculate the transformation between the optical sensing and vision-based camera coordinate systems.

Everyday objects face many problems when integrating sensing as part of their detection process. For example, inertial sensors are expensive; hence they cannot be routinely integrated with all objects. Separate outside-in trackers (such as optical or magnetic trackers) either require additional hardware in the object or environment and explicit calibration before use.

2.7.2 Structured Light Sensing for Location and Pose

Lee et al. [Lee, Dietz et al. 2004; Lee, Hudson et al. 2005] propose using light sensors in the objects to help address some of the problems associated with vision-based detection approaches, such as the problems of figure-ground separation (identifying what is the object and what is the background), variable lighting conditions, material reflectance properties (as reflective or transparent objects are challenging to track) and non-planar or non-continuous surfaces.

Lee et al. embedded light sensors in the corners of an augmented portable display screen and projected a series of structured light gray codes towards the object. The sensors detect and transmit observed light values back to the projector and the projector directly locates the display screen in its frame of reference based on the detected changes in brightness over time. Unfortunately due to the temporal coding used this approach suffers from two problems, firstly that the user sees a distracting set of

flickering patterns on their object, and secondly the gray code localisation takes up to a second, so cannot be used to augment mobile objects in real-time.

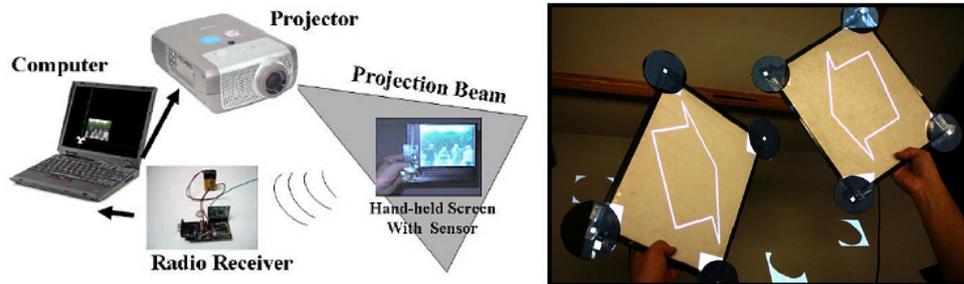


Figure 2.44 (left) Projection on mobile planar surfaces with single light sensor [Summet and Sukthankar 2005], (right) Projection onto surfaces with sensors at each corner for rotation information [Lee, Hudson et al. 2005]

Summet and Sukthankar extended this to real-time interaction on mobile screens [Summet and Sukthankar 2005], by restricting the size and location of projected patterns to the immediate area around the screen corners (as shown in Figure 2.44). This implementation reduced the flickering effect and allowed the remainder of the projection area to be used for display, however the update rate still limited movement speeds to slow hand motions before tracking was lost.

Raskar et al. demonstrated similar objects which sense projected structured light [Raskar, Beardsley et al. 2004], however, their implementation reverses the display paradigm by assuming a dynamic handheld projector used like a virtual flashlight and static objects. In this case, smart tags attached to objects use active RFID instead of a wireless RF link to return observed light values to the handheld projector. A gyroscope embedded in the projector also allows the object's relative 3D location to be calculated directly based on movement of the handheld projector coupled with the changes in the object's gray code value.

More recent work by Raskar et al. on their Prakash system [Raskar, Nii et al. 2007] presented a high-speed system that detects 3D location and orientation of photo-sensing tags using multiple Infra-Red (IR) projectors. Cheap Light Emitting Diodes (LED) were used as the light source, replacing the expensive video projectors used previously. The demonstration system achieved over a 500Hz update rate, based on fast switching of the LEDs and the use of sequentially illuminated dedicated binary code masks in the projector.

Structured light techniques require a minimum of one un-occluded light sensor in view of the projector to enable detection, or three light sensors for calculation of 3D location and orientation. The Prakash system [Raskar, Nii et al. 2007] reverses the paradigm, in that it requires only a single sensor, but a minimum of three LED projectors to calculate 3D location and pose. However, both approaches require many more sensors to guarantee correct pose calculation when used with 3D or self-occluding objects. For example, cubical objects would require either 18 light sensors (3 per face), or 6 for the Prakash system to order to detect all poses.

In contrast to the approaches that use magnetic, inertial or light sensors directly, in our work any movement sensor information available in a smart object is used to constrain the detection task. This allows indirect and opportunistic use of sensing to

increase detection robustness, without requiring expensive sensors, obtrusive sensors, or a minimum number of sensors to operate.

2.8 Summary

As we have seen in section 2.2.4, there is potential for output capability in smart objects to benefit the user by redressing the input-output imbalance in physical tangible interfaces. We reviewed projector-based augmented reality approaches as a suitable means to create non-invasive displays on objects and projector-camera system hardware used for the display itself. Smart objects can then achieve a display on their surfaces by cooperating with projector-camera systems. Of the approaches surveyed we integrate geometric, photometric and colorimetric correction in our work to enable displays that are undistorted and visible for an observer. Of the hardware surveyed we construct a steerable projector-camera system to enable vision-based detection, tracking and projection onto objects. This system is discussed further in Appendix A.

We reviewed computer vision approaches suitable for achieving vision-based detection and tracking of smart objects. Of the approaches surveyed in our work we integrate the markerless natural appearance detection cues of colour, texture, shape and features of objects described in section 2.6. We use multiple natural appearance cues as a single cue is not robust enough to detect all objects in every situation; however, we need to understand how to combine the cues to achieve the best detection performance. A first step towards this understanding is to investigate how the natural appearance detection methods perform with the object in different detection conditions, such as with scaling and rotation.

To increase robustness to some of the failure modes associated with pure vision-based detection approaches we integrate a hybrid vision and physical sensing detection method in our work. We propose using the sensing capabilities of a smart object in cooperation with vision-based detection in the projector-camera system, but we need to understand how sensing helps detection and what sensors are best suited for use.

This cooperative approach between smart object and projector-camera system is formalised in the Cooperative Augmentation Framework presented in Chapter 3.

Chapter 3 Cooperative Augmentation Conceptual Framework

In this chapter we present Cooperative Augmentation, a framework enabling smart objects and projector-camera systems to cooperate to achieve non-invasive projected displays on the smart object's surfaces.

There are eight defining characteristics of the Cooperative Augmentation concept:

1. **Generic, ubiquitous projector-camera systems offering a display service.**
All knowledge required to detect, track and project on objects was traditionally held by the projector-camera system in smart environments. In contrast, this knowledge is distributed among the smart objects in our approach, so that each smart object now contains the knowledge required to achieve a display. This reduces the projector-camera systems to providing a generic display service, allowing us to assume they are ubiquitous in the environment.
2. **Spontaneous cooperation between smart objects and projector-camera systems.**
Projector-camera systems in the environment are able to support spontaneous interaction with any type of smart object. An object simply registers for use of the generic display service to obtain an output capability on its surfaces.
3. **Smart objects embodying self-description knowledge.**
We assume smart objects are both real world objects with an inherent use and autonomous computational nodes. Objects cooperate with the projector-camera systems to achieve a display by describing knowledge they carry which is vital to the visual-detection and projection process, such as knowledge of their appearance. We call this information the "Object Model", as it is a representation of the object's appearance, form and capabilities.
4. **Dynamic tailoring of projector-camera system services to smart objects.**
The projector-camera system uses the Object Model to dynamically tailor its services to the object. The Cooperative Augmentation approach is flexible, as a dynamic configuration process caters for varying amounts of knowledge in the object. All configuration occurs automatically in response to the knowledge embodied by the Object Model.
5. **Using smart object capabilities to constrain detection and tracking.**
When sensor information is available from the object, this can be integrated in the detection and tracking process, allowing us to dynamically constrain the process and increase visual detection performance.
6. **Smart objects control interaction with projector-camera systems.**
After detection the smart object controls the interaction with projector-camera

systems. The smart object issues projection requests to the projector-camera system, controlling how the projected output on its surfaces changes and allowing direct visual feedback to interaction.

7. **Displays on the objects themselves, without modifying object appearance or function.**

Unlike physical embedded displays, the projected display is a temporary display which does not permanently modify the object's appearance or function.

8. **Projector-camera systems dynamically update knowledge held by the object.**

Over time, the camera system extracts additional knowledge about the object's appearance, and re-embeds this within the object, enriching the original Object Model and enabling increased detection performance.

This chapter expands the Cooperative Augmentation concept by explaining the four key areas of: the Cooperative Augmentation environment, the Object Model representation of the Smart Objects, the projector-camera system model and the actual Cooperative Augmentation process itself.

3.1 Cooperative Augmentation Environment

We assume all smart objects, projectors and cameras exist in a shared three-dimensional space, which we call the "environment". This allows us to locate each object in a shared frame of reference and easily model the relationships between devices. We term the shared frame of reference the world coordinate system, which is modelled as a three-dimensional Cartesian system. This can have an arbitrary origin in the physical world.

3.2 The Object Model

The "Object Model" is a description of a smart object and its capabilities, allowing the projection system to dynamically configure its detection and projection services for each object at runtime. We assume the Object Model knowledge is initially embedded within the object during manufacture. However, the knowledge can also be extended and added to by projector-camera systems.

The model consists of five components:

1. **Unique Object Identifier**

This allows an object to be uniquely identified on the network as a source and recipient of event messages and data streams. For example, by the IP address of the object's hardware.

2. **Appearance Knowledge**

This knowledge describes the visual appearance of the smart object. The description is specific information extracted by computational methods from camera images of the object. For example, knowledge about the object's colour, or locations and descriptions of features detected on the object.

3. **3D Model**

A 3D model of the object is required to both allow a projector-camera system to

compute the object's pose and enable the framework to refer to individual surfaces.

4. **Sensor Knowledge**

The sensor model is a description of the data delivered by the object's sensors. The data type is classified into three groups with regard to the originating sensor: movement sensor data, light sensor data and others. The data is further classified into streaming or event-based, depending on the way sensor data is output from the smart object. The model contains associated sensor resolutions, and sensor range information to allow the framework to interpret sensor values.

5. **Location and Orientation of the Object**

When an object enters an environment, it does not know its location and orientation in the world coordinate system. A projector-camera system provides this information on detection of the object, to complete the Object Model.

3.3 The Projector-Camera System Model

A projector-camera system consists of a projector, camera and their controlling systems. While many projector-camera systems are typically co-located devices (such as steerable projector-camera systems), we model the physical projector and camera as independent objects. However, each requires knowledge of its current location and orientation in the world coordinate system. This knowledge can be obtained by 5 methods:

1. Direct measurement for static devices.
2. Self-calibration and location methods for static devices (i.e. combining techniques discussed in sections 2.3.3, 2.3.6 and 6.4, or multi-camera calibration techniques [Sinha and Pollefeys 2006]).
3. Vision-based Simultaneous Location And Mapping (SLAM) methods (e.g.[Davison and Murray 2002; Davison 2003; Chekhlov, Gee et al. 2007; Klein and Murray 2007]) for mobile devices.
4. By calculation, where steerable hardware is installed in a known orientation [Spasova 2004].
5. From 3D location and orientation sensing systems attached to the device.

This approach allows virtualisation of a projector-camera system pairing across multiple projectors and cameras. The framework can now use any projector or camera hardware distributed in the environment in addition to static, steerable, mobile and handheld projector-camera systems. For example, in an environment with many distributed fixed cameras and a handheld projector, the camera used as part of a projector-camera system pair could vary depending on the location of the projection. In this case, as each device has knowledge of its location and orientation we can calculate the closest camera or the camera with the best view of the projection.

We assume projector-camera pairs only exist when the respective viewing and projection frustums overlap, allowing objects detected by the camera to be projected on by the projector.

A projector-camera system has five capabilities in our framework:

1. To provide a service allowing smart objects to register for detection and projection.
2. To detect smart objects in the camera images and calculate their location and orientation based on the knowledge and sensing embedded in the object, as explained in section 3.4.
3. To project an image onto an object in an area specified by the smart object, or choose the area most visible to the projector.
4. To perform geometry correction to a projected image so that the image appears to be attached to the object's surface and is undistorted.
5. To perform photometric correction to a projected image, compensating for variation in an object's surface colour and texture so the image appears more visible.

In addition, the framework allows explicit modelling of steerable projector-camera system pairs. In this case we assume the projector and camera are co-located and their respective viewing and projection frustums constantly overlap.

A steerable projector-camera system has two additional capabilities:

1. To search an environment for smart objects by automatically rotating the pan and tilt hardware.
2. To track detected objects by automatically rotating the pan and tilt hardware to centre the detected object in the camera and projector frustums.

3.4 Cooperative Augmentation Process

We decompose the cooperative augmentation of an object into five steps:

1. **Registration**

As an object enters the environment it detects the presence of a location and projection service through a service discovery mechanism. The object sends a message to the projector-camera system requesting registration for the projection service to display messages. On receipt of the registration request, the projector-camera system requests the Object Model from the smart object.

2. **Detection**

Following registration, the object begins streaming sensor data to the projector-camera system, as shown in Figure 3.1 (A). This data is used in combination with the Object Model to constrain the visual detection process and generate location and orientation hypotheses (B). When an object is located with sufficient accuracy, a location and orientation hypothesis is returned to the smart object (C) to update the Object Model. This process is explained in more detail in section 3.4.1.

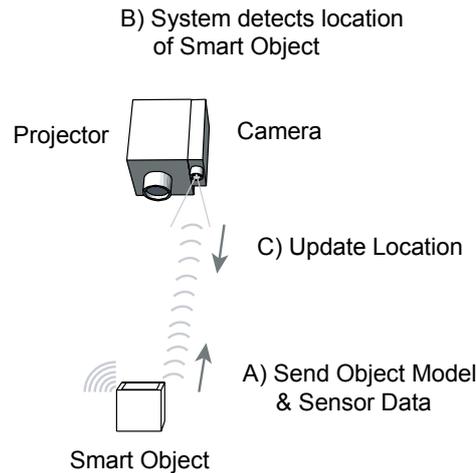


Figure 3.1 Detection Sequence Diagram

3. Projection

When an object has knowledge of its location and orientation it can request a projection onto its surfaces. For example, as the projector sends location information to the object, if an object is placed in the wrong area of the environment it could request a warning message is projected until moved to the correct location. This projection request message contains both the content to project and location description of where on the object to project the content, as shown in Figure 3.2 (A). Any geometric distortion that would appear when projecting on an object non-orthogonal to the projector is automatically corrected. The projection image is additionally corrected for the surface colour of the object to make it more visible to the user (B). The projector system starts displaying the corrected content on the objects surfaces immediately on receipt of the request, if the object is in view and the projector system is idle (C). This process is explained in more detail in section 3.4.2

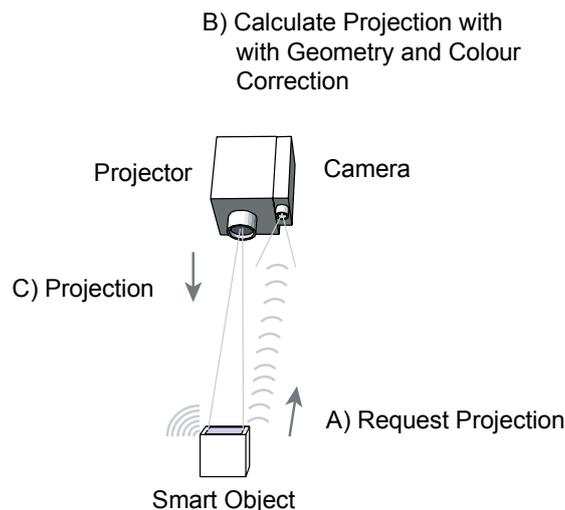


Figure 3.2 Projection Sequence Diagram

4. Interaction with smart object

A requested projection is active as long as the object is detected, including during movement or manipulation of the object. Consequently, smart objects can give

direct feedback to the user in response to the manipulation or movement of the object by changing the projection. Interactive user-interface components can also be projected, exploiting a user's experience with traditional desktop interfaces while allowing direct visual feedback on the object itself. The possibilities for interaction with smart objects are discussed in more detail in section 6.2.4.

5. Update Appearance

Additional information about the appearance of an object's surfaces can be extracted once the object has been detected and its pose calculated. As part of the cooperative process this new knowledge can be re-embedded into the Object Model for faster and more robust detection on next entry to an augmented environment. Even if an object is already detected reliably with one detection method, extracting more knowledge is beneficial as the environment can also change. For example, when distracting objects are introduced with similar appearances, the wall is painted a different colour, or the object is simply taken to another room. Similarly, the appearance updating capability allows deployment of new detection algorithms to the projector-camera system and automatic update of the object's appearance description to include the new algorithms following first detection with another method.

3.4.1 Detection

The Object Model transmitted to the projector-camera system contains an appearance description which allows visual detection of the object. The projector-camera system dynamically configures its visual object detection processing based on the type of appearance knowledge in the Object Model, and the sensors the object possesses. As seen in Figure 3.3, this processing involves computation of one or more vision detection methods on images from a camera system, location and orientation (pose) computation for one or more object location and orientation hypotheses and return of the best hypothesis to the Object Model.

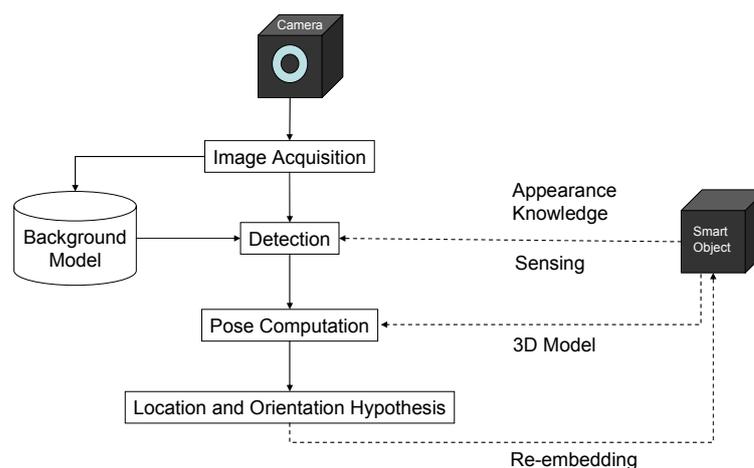


Figure 3.3 Knowledge flow in the detection process

The cooperative augmentation framework serendipitously uses sensors the object possesses to constrain the detection process. One typical class of sensor that can be used is movement sensors. Common sensor hardware that can be used for movement detection are accelerometers, ball-switches, tilt-switches and force sensors which detect pick-up and put-down events. If an object is moving, we use visual differences generated between the camera image and a background model to provide a basic figure-ground segmentation in the detection process, increasing the probability of correct detection.

Maintaining a background model also allows us to take the object's context into account in the detection process. For example, if we know the object's colour is similar to the background colour we would not use a colour detection method, as the probability of detection is low.

3.4.2 Projection

Following projection requests there are cases where projection cannot begin immediately, such as where the projection system is busy, the object is occluded or the object is out of the field of view of the projector. Here the display requests are cached at the projector system and the projection commences when the object is in view and the projector is available. Projection requests are displayed sequentially and simultaneous projection onto multiple objects is possible when all objects are within the field of view of a projector-camera system.

If multiple projectors exist in the environment, the display request applies simultaneously to all projectors. This allows the object to roam freely in an environment and achieve a display whenever it is in the field of view of a projector. To prevent multiple projectors overlaying displays onto the same surface a display rights token system is introduced. Here, each projector determines the visual quality of its projection based on the distance and relative orientation of the object's surface to the projector using the metric introduced by Ehnes and Hirose [Ehnes and Hirose 2006]. Closer, more orthogonal projectors score higher, allowing a ranking to be performed and the best projector assigned display rights either for the whole object or on a per-surface basis.

A rectangular image projected on to a non-perpendicular or non-planar surface exhibits geometric distortion. We compensate for this distortion by warping our projected image, as we know both the surface geometry of the object and the pose of the surface with respect to the projector. We obtain the surface shape from the geometric 3D model embedded within the Object Model and the pose of the object from the detection process. As we have seen in related work section 2.3.6 the geometric correction methods required depend on the surface geometry, with curved surfaces requiring a different approach to planar surfaces. Hence, we use the surface shape to directly configure the type of geometric correction applied in projection.

As one goal of the framework is to avoid the physical modification of the appearance of smart objects, their surfaces can present a challenge to projection. Smooth, diffuse and light coloured object surfaces are ideal for projection; however, few everyday objects exhibit these characteristics. Certain combinations of projected content and object surface colour can make the projection almost invisible to the human eye, for example, when projecting yellow text on a deep red background. Conversely, with a smooth, diffuse, light coloured object, the projection illumination on the object can significantly alter its appearance, causing the visual detection process to fail.

To compensate for this problem we use photometric and colorimetric correction techniques discussed in section 2.3.7 in the projection process.

3.5 Conclusion

This chapter expanded and formalised the Cooperative Augmentation concept introduced in Chapter 1, to provide detailed information about the four key areas of the framework (the Cooperative Augmentation Environment, the Object Model, the projector-camera system model and the cooperative augmentation process). We illustrated how knowledge from the Object Model and sensing can be used with a projector-camera system to cooperatively detect an object and project onto its surfaces.

An example implementation of this framework is discussed in Chapter 6 and Chapter 7. The next two chapters look in more detail at the detection process. Firstly, Chapter 4 investigates natural-appearance vision-based detection algorithms, while Chapter 5 explores how embedded sensing in the smart objects can be used in cooperation with visual detection to improve detection performance.

Chapter 4 Vision-Based Object Detection

A central problem for achieving displays on smart objects is their detection and tracking. As discussed in section 2.4, common approaches to object tracking involve embedding dedicated hardware location systems. However, many systems have limitations such as a small working volume, which precludes their use with mobile smart objects in unconstrained environments. In contrast, vision-based detection is commonly used in experimental prototypes by placing planar fiducial markers on objects. This enables detection of mobile objects anywhere in the camera's field of view. However, it requires modifying the appearance of an object with visually intrusive markers to enable detection, and for our work it suffers from several key limitations, as discussed in section 2.5.

Consequently, with a view to ubiquitous augmentation of objects, it is more realistic to base detection on the natural appearance of objects. While this vision-based detection non-intrusive, it is a significant challenge in real-world environments, as objects naturally vary in their appearance. Hence, there is an open question as to how best to use natural appearance detection and how different methods perform in different detection conditions, such as when an object appears with scaling and rotation as it moves around an unconstrained environment.

4.1 Natural Appearance Detection

In this chapter we investigate four detection methods, representing different natural appearance cues of objects (colour, texture, shape and surface features). The rationale for studying approaches that rely on different cues is that objects in real world naturally vary in their appearance – hence we assume that multiple methods should be provided as alternatives for detection. No single cue is both general enough and robust enough to cope with all combinations of object appearance and environment.

We perform an experimental study targeted at understanding the impact of object scale and rotation on different detection methods. This is important for detection in realistic scenarios, as objects will appear at varying distances and orientations with respect to the camera. We also look in-depth at the features cue, investigating the impact of invariance to scale and rotation in different feature detection algorithms.

As a result of the study, we gain insight into training requirements of different detection approaches to enable vision-based detection. This is important for embedding appearance knowledge into smart objects to achieve initial detection and for the

cooperative augmentation process to know when the appearance knowledge updating step is best performed (see section 3.4).

4.2 Object Detection Methods

Four detection methods were chosen as being well-known and typical for their respective appearance cues. These four methods form the basis for the studies presented in section 4.4 and Chapter 5:

For **Colour** we use Swain and Ballard’s colour histograms [Swain and Ballard 1991]. We choose to use CIE 1976 L*a*b* colour histograms, as this model was empirically found to detect light and dark objects better than Hue-Saturation-Lightness or RGB colour models. ‘L’ defines lightness; ‘a’ is the red/green value and ‘b’ the yellow/blue value. Unlike the RGB model, Lab colour is designed to approximate human vision, aiming for perceptual uniformity. We calculate a 3-dimensional histogram of the image, with each dimension divided into 16 bins, each 16 values wide.

For **Texture** we use Schiele’s multidimensional histograms of receptive fields [Schiele and Crowley 2000]. We choose to use 2-dimensional Gradient Magnitude and Laplacian histograms due to their invariance to rotation in the camera plane. Each dimension has 32 bins, each 8 values wide. Scale invariance is achieved by training with images of the object at multiple scales then creating a scale space when detecting the object by Gaussian smoothing the image with increasing standard deviation (σ). Here we train with images smoothed with $\sigma=2.0$ then use 3 scales in detection, equal to 0.5σ , σ , 2σ , to allow objects at different scales to be detected.

For **Shape** we use Belongie’s shape context descriptors [Belongie, Malik et al. 2001]. The contours on an object are matched with 100 points, using 5 radial bin and 12 angle bin histograms. The descriptors are made scale invariant by resizing the diameter of the radial bins equal to the mean distance between all point pairs, and rotation invariant by averaging the angle of all point pairs and calculating all angles relative to the mean.

For **Features** we first perform both an in-depth study of the Local Features method, where we evaluate 9 detection algorithms: Harris, Harris-Laplace, Harris-Affine, Hessian, Hessian-Laplace, Hessian-Affine, Difference-of-Gaussians (DoG), Laplacian-of-Gaussians (LoG) and Maximally Stable Extremal Regions (MSER) [Mikolajczyk, Tuytelaars et al. 2005]. All these algorithms are evaluated in combination with Lowe’s Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004].

For the remainder of the studies in this work we use only the complete SIFT algorithm presented by Lowe (comprising DoG detector and SIFT descriptor) as this is one of the most popular and widely used local feature algorithms, giving a good compromise between discriminative power and robustness [Mikolajczyk and Schmid 2005]. SIFT is invariant to scale, rotation in the plane of the camera image and partially invariant (robust) to changing viewpoint and illumination. We use the standard scale space image pyramid presented by Lowe, with 3 scales per octave and $\sigma = 1.6$.

In terms of complexity, the methods range from low (colour histograms) to high (SIFT local features), with texture and shape falling somewhere in between. More information on each of these methods can be found in related work section 2.6.2.

4.3 Evaluation Dataset

We use ten objects as the dataset for experiments, reflecting everyday objects of varying shape, size and appearance (see Figure 4.1). They are a football, a chemical container barrel, a book, a product box, a smart cube, a chair, a cup, a notepad, a cereal box and a toaster. The largest object was the chair (90x42x41cm); the smallest was the cube (9x9x9cm).

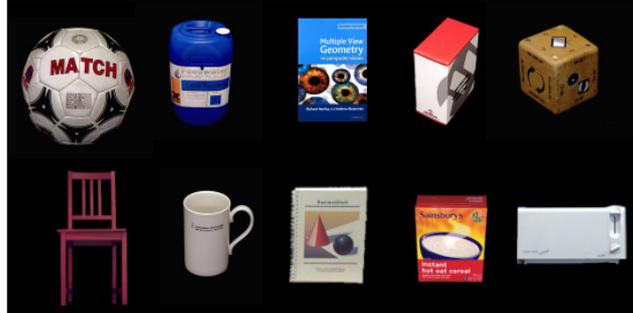


Figure 4.1 Experiment Objects (left to right, top to bottom): a football, a chemical container barrel, a book, a product box, a smart cube, a chair, a cup, a notepad, a cereal box and a toaster.

4.3.1 Object Appearance Library

We created an object appearance library to train algorithms for the experiments presented in section 4.4 and Chapter 5. The library consists of images of the objects with varying scale and rotation against plain backgrounds. Colour images of each object were acquired with even illumination using a Pixelink A742 machine vision camera, with 1280x1024 pixels and a 12mm lens (with a $40.27 \times 30.75^\circ$ field of view). For the varying scale, images of the objects were captured in 5cm intervals between 1 and 6m from a fixed camera ($10 \times 100 = 1000$ images). 6m approximated the size of a large room used in our scenario. The camera was horizontal, perpendicular to the front surface of the objects and at the vertical centre of the object. For rotation, images of the objects were captured at distances of 2m, 3m, 4m and 5m from a fixed camera. At each distance the objects were rotated in 10° intervals for a full 360° around the object's vertical world axis using a turntable ($4 \times 36 \times 10 = 1440$ images total). The camera was fixed 1.5m above the height of the turntable with a 40° declination angle, providing a view of both the top surface and sides of the objects on the turntable between 2 and 5m from the camera. All images were manually annotated with an object bounding box for a ground truth object location.



Figure 4.2 (left) Box Object Scale images at 1m, 3m, 6m from camera, (right) Notepad object rotation images at -40° , 0° , $+40^\circ$

4.4 Scale and Rotation Experiments

One of the challenges for visual detection algorithms is to reliably detect objects. Real-world objects are composed of different structures at different scales, which means in practice they appear different depending on the scale of observation. This effect can be seen in Figure 4.3, where different numbers of corners are detected and in different locations when the cereal box object appears at different distances to the camera.

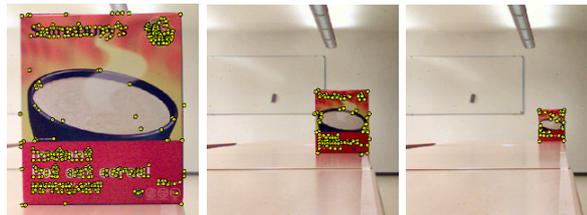


Figure 4.3 Different numbers of corner features (yellow dots) are detected and in different locations on the Cereal box object at 1m, 3m and 6m distance with the single-scale Harris algorithm [Harris and Stephens 1988]

When detecting an object in an unknown scene, there is no way to know at which scale the object will appear, as the distance to the object is unknown. In theory, by representing the object or camera image at multiple scales in a scale-space, detection algorithms can be made scale-invariant. A scale-space is built by successively smoothing an image using a Gaussian kernel with an increasing standard deviation (σ), to remove more and more fine detail [Lindeberg 1990]. We can now try to match the object at different scales or choose the most appropriate scale.

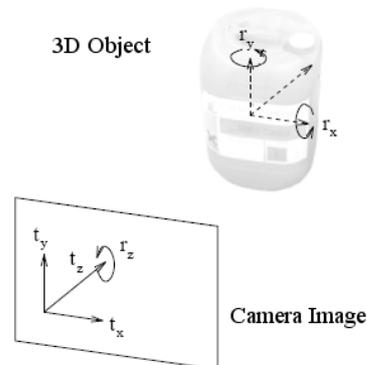


Figure 4.4 Camera and Object Detection Coordinate System Transformations

Rotation of an object can be decomposed into 2D rotation in the camera plane r_z , equivalent to rolling the camera and general 3D rotation (r_x, r_y) equivalent to changing the camera viewpoint (see Figure 4.4).

When an object is rotated in the 2D plane of the camera image (r_z), the method we use to detect an object may cause it to appear different. For example, if our detection algorithm relies on first gaussian derivatives (dx, dy) for gradient calculation, the results will change depending on the orientation of the object. Detection algorithms can be made invariant to this 2D rotation, however, general 3D rotation of an object presents another problem. In this case there are two separate aspects to the problem:

1. The appearance of an object surface becomes distorted when it is rotated from being perpendicular to the camera vector t_z .

2. As the object rotates, surfaces disappear from view and new surfaces appear.

We perform experiments to answer the four following research questions:

- R1)** What is the impact of using local feature algorithms invariant to scale, rotation or affine transformation, rather than non-invariant algorithms?
- R2)** Is scale and 3D rotation an issue for the final 4 detection methods we choose?
- R3)** At what distance do we need to train our 4 methods?
- R4)** Are some of the 4 methods more robust to scale and rotation than others?

4.4.1 Design

4.4.2.1 Research Question R1

To address question R1 we perform four series of experiments using 9 local feature detection algorithms {Harris, Harris-Laplace, Harris-Affine, Hessian, Hessian-Laplace, Hessian-Affine, Maximally-Stable-Extremal-Regions (MSER), Difference-of-Gaussians (DoG), Laplacian of Gaussians (LoG) } and the SIFT descriptor algorithm:

1. The first experiment investigates the quantity of features detected with scale.
2. The second experiment series investigates detection repeatability over the whole scale range when the algorithms are trained at different distances. For six training distances scale variant algorithms were compared against scale and affine invariant algorithms.
3. The third experiment series investigates 2D rotation of an object in the camera plane (r_z) and rotation invariance in feature descriptors by comparing the descriptor matching performance of rotation variant and rotation invariant algorithms under 2D rotation.
4. The fourth set of experiments addresses the general case of local feature detection performance with 3D object rotation (r_x, r_y) by comparing the detector matching repeatability and descriptor matching performance of rotation variant and rotation invariant algorithms under 3D rotation.

4.4.2.2 Research Questions R2 to R4

R2 to R4 look at the performance of final four detection algorithms: Lab colour histograms, Mag-Lap multi-dimensional histograms for texture, shape context and SIFT for features on objects. We address R2 and R4 by performing another two series of experiments:

1. The first series evaluates the average detection performance over all objects of each algorithm, when the objects are scaled.
2. The second series evaluates the average detection repeatability for SIFT local features over all objects, and the average detection performance of the other three algorithms over all objects, when the objects are rotated.

We investigate if scale and rotation is an issue, and compare the results between algorithms to identify whether some detection methods are more robust than others. From this we also gain insights into the training knowledge required to detect objects, addressing R3.

4.4.2 Procedure

For the local feature experiments, we define a procedure for calculating detection repeatability and descriptor matching performance as explained below.

To compare the relative performance of interest point detectors we use the repeatability criterion described by Mikolajczyk and Schmid [Mikolajczyk and Schmid 2004]. For this we compute the percentage ratio between the number of point or region correspondences (found between the current image from the scale set and the features found on the object in the training image) and the minimum number of points detected in both images. The correspondences are established by projecting points from each image into the other using a manually-annotated ground truth homography. We establish a correspondence when the following two criteria are met:

1. If the point locations are less than 1.5 pixels apart when projected.
2. Additionally, for scale and affine-invariant points, when the image region we project has an overlap intersection error $\epsilon_s < 0.4$. This error corresponds to 40% overlap error and is chosen as according to Mikolajczyk et al. as regions with 50% overlap error can still be matched successfully with a robust descriptor [Mikolajczyk, Tuytelaars et al. 2005]. The intersection error of the regions (ϵ_s) is defined as the intersection and union of the regions:

$$\epsilon_s = 1 - \frac{\mu_a \cap (A^T \mu_b A)}{(\mu_a \cup A^T \mu_b A)} \quad (4.1)$$

where μ_a and μ_b are the regions (represented as elliptical regions for affine-invariant features or circles for scale-invariant features, and defined by $x^T \mu x = 1$), A is the locally linearised ground truth homography relating the images and A^T its transpose [Mikolajczyk, Tuytelaars et al. 2005]. The error can be computed numerically, or in our case by counting the number of pixels in the union and the intersection of regions when one region is projected into the second using the homography.

Relative matching performance of the descriptors is evaluated by establishing correspondences between descriptors in the test image and descriptors in the training image using nearest-neighbour Euclidean distance. Correct matches are determined by the overlap error (as for detector repeatability), but here we assume a match is correct if the overlap error is $< 50\%$ of the region ($\epsilon_s < 0.5$). The final matching percentage score (known as Recall) is calculated as the number of correct matches with respect to the total correspondences:

$$\text{Recall} = \frac{\# \text{ Correct matches}}{\# \text{ Correspondances}} \quad (4.2)$$

4.4.2.1 Research Question R1

1. For the first series of experiments the 9 local feature algorithms were run on each image from the object appearance library scale set (see section 4.3.1) and the number of features detected inside the ground truth object location bounding box was recorded.
2. The second series is divided into 6 sub-experiments, corresponding to 6 algorithm training distances. Each local feature algorithm was trained at meter intervals between 1 and 6m (inclusive) using images from the object appearance library scale set. For training, feature detection was constrained to only detect features on the object by using the manually annotated ground-truth bounding box. Features were then detected in the remaining images in the scale set and the detection repeatability between the training and test images calculated using the method described above.
3. For each of the images of objects at 3m distance in the object appearance library scale set (see section 4.3.1) a 2D object rotation was simulated by rotating images in 10° increments between 0° and 350° using an affine transform. Simulation was used for this experiment as it was the easiest way to ensure accurate and repeatable rotations across all the objects without the need for specialized hardware to physically rotate the object. The SIFT descriptor [Lowe 2004] is computed for each interest point detected by the local feature detection algorithms in all images. The detection algorithms were trained with images of all objects at 0° rotation and matched against images of the objects at the other rotation angles to compare non-rotation invariant descriptors against rotation invariant descriptors.
4. The experiment is divided into 4 sub-experiments. The 4 sub-experiments are for 4 algorithm training distances, at meter intervals between 3 and 6m. In each sub-experiment the 9 algorithms were trained with the 0° rotation images of each object at the respective distance from the object appearance library rotation set (see section 4.3.1). These training images were matched to the -70° to $+70^\circ$ rotation images, comparing both scale invariant against non-scale invariant detector repeatability and non-rotation invariant descriptors against rotation invariant descriptor matching performance.

4.4.2.2 Research Questions R2 to R4

For the final 4 detection algorithms we define a procedure for calculating detection performance as detailed below.

For colour histogram detection we use a variation of Bradski's CAMSHIFT approach [Bradski 1998]. The object histogram is first back-projected into the camera image and the bounding box of the largest blob is detected, representing the most likely object location. The CAMSHIFT algorithm is then used to refine the bounding box size and location based on the original object histogram.

For texture detection with multi-dimensional histograms we use an exhaustive search method, dividing each scale image into a grid of scale-adapted 2D windows of uniform size, with 25% partial overlap between each window. Each area's histogram is calculated and matched against the object's histogram. For object location use a mean-shift [Comaniciu and Meer 2002] clustering approach similar to Liebe and Schiele's scale-adaptive method [Leibe and Schiele 2004] when match result maxima are greater

than a pre-determined threshold. Hence, the 3D mean-shift clustering function (X, Y, Scale) acts as a Parzen window probability density estimation for the position of the object, allowing a more accurate location hypothesis. We increase the clustering window size as scale increases above 1.0 as the histogram matching results are spread over a larger area.

For shape detection we use a similar approach to texture, but restricted to single scale, due to the scale-invariance of our shape context algorithm.

For colour histogram, shape context and multi-dimensional histograms algorithms, correct detection was assumed when the detection bounding box had <50% overlap error with the ground truth bounding box from the test image library. For SIFT local features correct detection was assumed when a minimum of 8 features were matched to the training image using nearest-neighbour Euclidean distance matching and >50% of feature correspondences were correct.

1. The first experiment set investigates scale-invariance in the 4 detection algorithms and aims to quantify in what scale range we can repeatably detect an object. We performed 6 sub-experiments, each with a different training distance. All of the algorithms were trained with an image at meter intervals between 1 and 6m (inclusive), resulting in the 6 sub-experiments. The remaining images of the object appearance image library (99 images) were used for testing. The results for all four cues are averaged over all objects to give the percentage of all detected objects over the scale range (every 5cm).
2. The second set of experiments investigates 3D rotation. For each object we trained the algorithms using a single 0° image from the object appearance library. The remainder of the images between -80° (anti-clockwise rotation of object from 0°) and $+80^\circ$ (clockwise rotation) were used to evaluate the percentage of objects detected with each algorithm. We both trained and tested the algorithms with images of the objects at 3m distance as this is the centre of our working range.

4.4.3 Apparatus

A 3.4GHz dual core Pentium-4 computer running Windows XP SP2 was used for all experiments in this thesis. The detection algorithms were implemented in C++ using Intel OpenCV API for image processing

4.4.4 Results

We first present results for the first four experiments to investigate local feature algorithm performance with scaling and rotation, followed by results of the detection performance for the final four selected detection algorithms for colour, texture, shape and features.

4.4.4.1 Research Question R1: Local Features with Scaling and Rotation

Although we performed the experiments for all objects in the object appearance library, we only present the results for the Mediacube object below as the general trends for all objects are the same, hence, these results are representative of the other objects. Additionally, we only present results for the Harris-based family of detectors as the result trends are also representative of the relative Hessian-based detector performances.

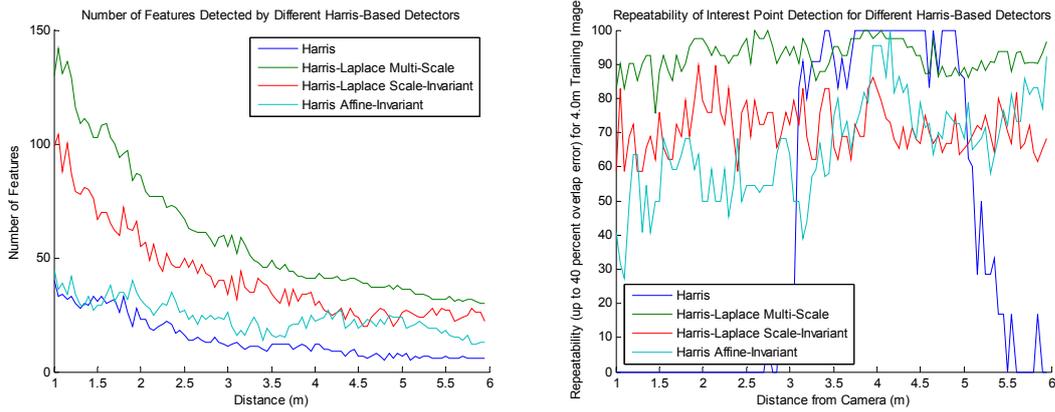


Figure 4.5 (left) Number of features detected on Mediacube object by Harris-based algorithms, (right) Harris-based algorithm detector repeatability when trained at 4m

As can be seen in Figure 4.5 (left), the number of points detected by all Harris based detectors decreases with distance, as the object appearance gets smaller in the image. The single-scale Harris detector always detects the least interest points as only the original image scale is used. In contrast, the Harris multi-scale detector detects all points where the Harris operator reaches a maximum in scale-space, leading to between 3 and 5 times more interest points throughout the whole object distance range. The number of points detected by the Harris-Laplace scale-invariant detector falls between the two, as it only detects points where the Laplacian also reaches a maximum. The affine-invariant Harris detector returns fewer points than the scale-invariant detector as the algorithm discards points which are not also invariant to affine transformations.

The difference in repeatability of the Harris-based detectors when trained with a single image of the object at 4m distance can easily be seen in Figure 4.5 (right). Here the single-scale Harris only has a small 2m range (3m to 5m) where points are repeatably detected, centred on the training distance. In contrast, multi-scale Harris-Laplace and scale-invariant Harris-Laplace have an almost constant performance across the scale range. Harris-Affine displays a general improving trend from 1m towards 6m, with a peak around the 4m training distance. For a 4m training image the scale factor ranges from 1:4 at 1m to 1.43:1 at 6m, with 4m being 1:1.

Figure 4.6 (left) shows that for the single scale Harris algorithm to achieve equivalent detection repeatability approaching the multi-scale and scale-invariant Harris-Laplace algorithms (over 60% throughout the test range), we must train and match the algorithm at 4 separate distances. The training images distances of 1.1m, 1.85m, 3m and 5m were determined empirically to have both minimum scale overlap between detectors and to maximise the repeatability across the whole scale range. This is equivalent to running the Harris detector algorithm 4 times independently on the image, with the consequent quadrupling of runtime.

The mean repeatability of the Harris-based algorithms over the whole distance range while varying the training data distance is shown in Figure 4.6 (right). Here we can see that both the multi-scale and scale-invariant Harris-Laplace algorithms have a stable mean repeatability around 91.22% and 75.12% respectively when the training image distance is varied. However, the training image scale makes a large difference both for Harris and Harris-Affine. The Harris algorithm has the highest repeatability (41.10%) when trained at 5m, and Harris-Affine reaches a peak of 74.98% when trained with images of the object at 6m.

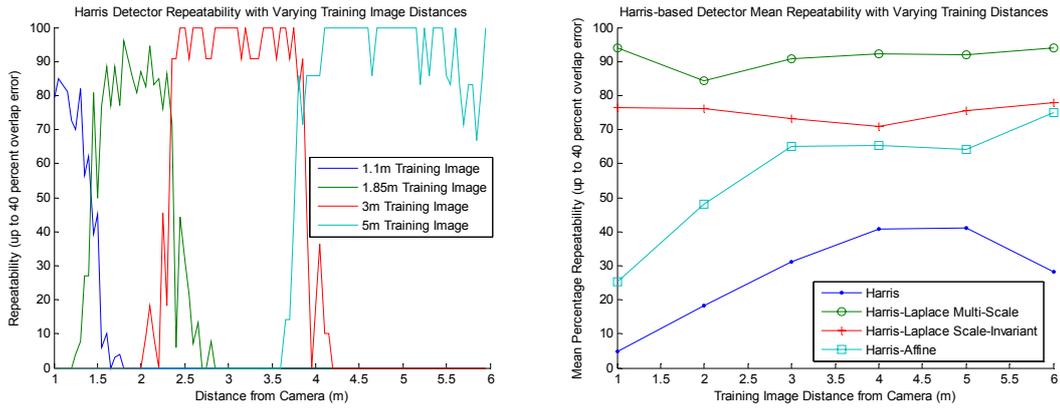


Figure 4.6(left) Harris detector with multiple training images 1.1m,1.85m,3m,5m, (right) Mean repeatability with varying training distance for Harris-based detector algorithms

Figure 4.7 (left) compares rotation-invariant descriptors with rotation-variant descriptors, both with and without multiple training images. As shown by the red dashed line, the rotation-variant Harris-Laplace-SIFT matching result with only a single training image is only ever above 50% within 50° of the training image 0°/360° angle. In contrast, the rotation-invariant descriptor performance was more constant, with a mean average around 60% for the whole 360° rotation. Rotation-variant matching performance was also evaluated relative to multiple training images. In this case 60° rotation increments were determined empirically to produce performance similar to rotation-invariant descriptors, while using the least training images.

For 3D objects such as the Mediacube the number of features detected did not change significantly with 3D rotation, as seen in Figure 4.7 (right), due to textured surfaces on multiple faces being visible at each rotation angle. However, for 2D objects such as the notepad or objects where some surfaces lacked strong texture the number of points detected depended greatly on viewpoint. This led to large changes in the detector repeatability and descriptor matching percentages with 3D object rotation.

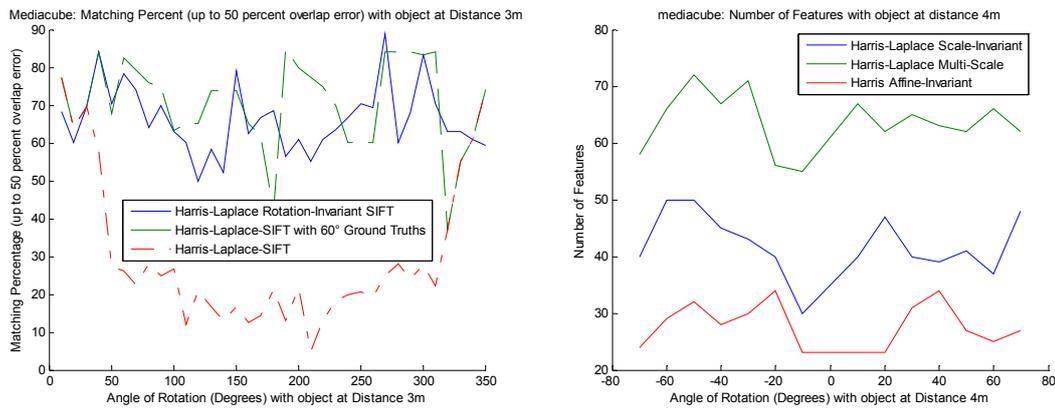


Figure 4.7 (left) SIFT Descriptor matching percentages with 2D rotation, (right) Number of features detected with 3D rotation

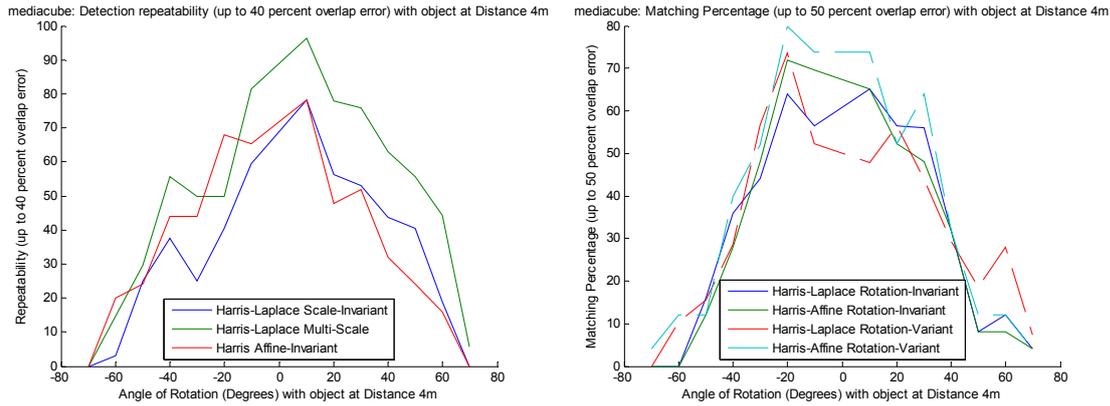


Figure 4.8 (left) Detector repeatability for 3D rotation, (right) Descriptor matching percentages for 3D rotation

The repeatability of interest point detection with viewpoint transformation is shown in Figure 4.8 (left). As can be seen, for all algorithms the repeatability is generally at a maximum near the training image orientation (0°) and decreases with increasing rotation in either direction. In this graph the multi-scale Harris-Laplace algorithm performs best. Affine-invariance only leads to a small improvement in mean repeatability over scale-invariance of 37% against 34% respectively, over the whole rotation range. On average, between 20° and 40° from the training image orientation the repeatability has decreased to 50% or under.

The matching performance of SIFT descriptors with 3D object rotation is shown in Figure 4.8 (right). As can be seen, all algorithms have almost identical performance irrespective of whether the descriptor is rotation-variant or invariant.

4.4.4.2 Research Questions R2 to R4: Detection with Scaling and Rotation

To address questions R2 to R4 with the final 4 detection algorithms we first present the detection results for scale over all objects, followed by the results for 3D rotation:

Figure 4.9 (left) shows detection performance over all objects for SIFT local features. It is clearly visible that the percentage of objects detected falls below 50% after 4.5m distance when trained at 1m. For the chosen scale range, the average percentage of objects detected is highest when trained at 2m ($M=82.31$, $SD=14.34$). However, the detection percentage varies least across the scale range when trained at 6m ($M=74.44$, $SD=9.32$).

Figure 4.9 (right) shows detection performance over all objects for the Shape Context algorithm. Detection percentage is lowest when we train at 6m, never increasing above 30% across the scale range. The performance when training at both 1m and 3m both show a downward trend with similar detection performances, however, the highest percentage of objects were detected at when trained 1m ($M=56.10$, $SD=19.71$).

Figure 4.10 (left) and (right) show detection performance over all objects for Texture and Colour respectively. The results for both experiments show an overall downward tendency for all training distances, with little difference in performance between the training distances. For Texture, the highest percentage of objects were detected with a training distance of 2m ($M=72.74$, $SD=26.59$). Here, 50% or more of the objects are detected between 1m and 6m, with the exception of 4.5m to 5.5m where, unusually, the performance drops. In contrast, for Colour the highest percentage of objects are detected when we train at 6m ($M=38.05$, $SD=46.81$). The standard deviation for all colour

training distances is high, indicating a high variability of detection performance between objects. In fact, 50% of the objects were never detected by colour.

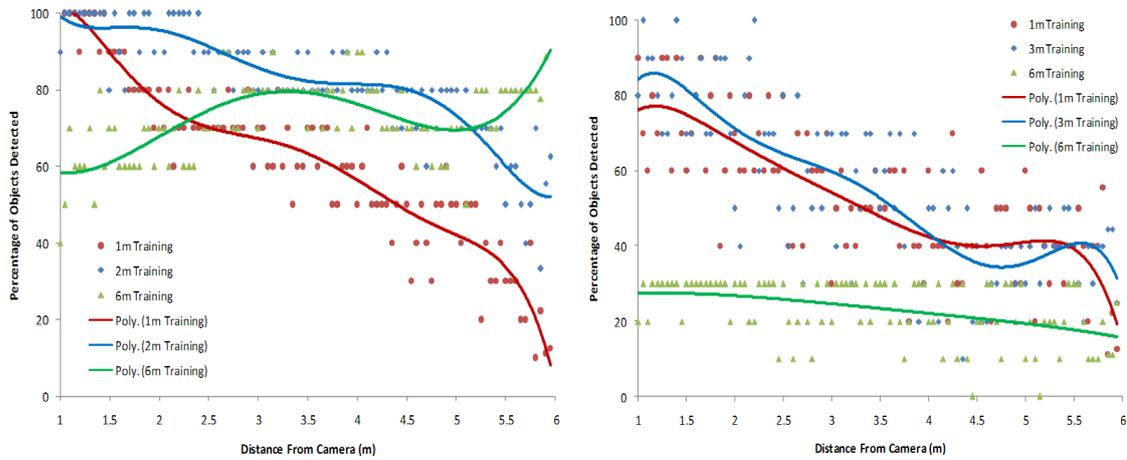


Figure 4.9 (left) Detection performance over all objects for SIFT local features at 1m, 2m and 6m training distances, (right) Detection performance over all objects for Shape Context at 1m, 3m and 6m training distances.

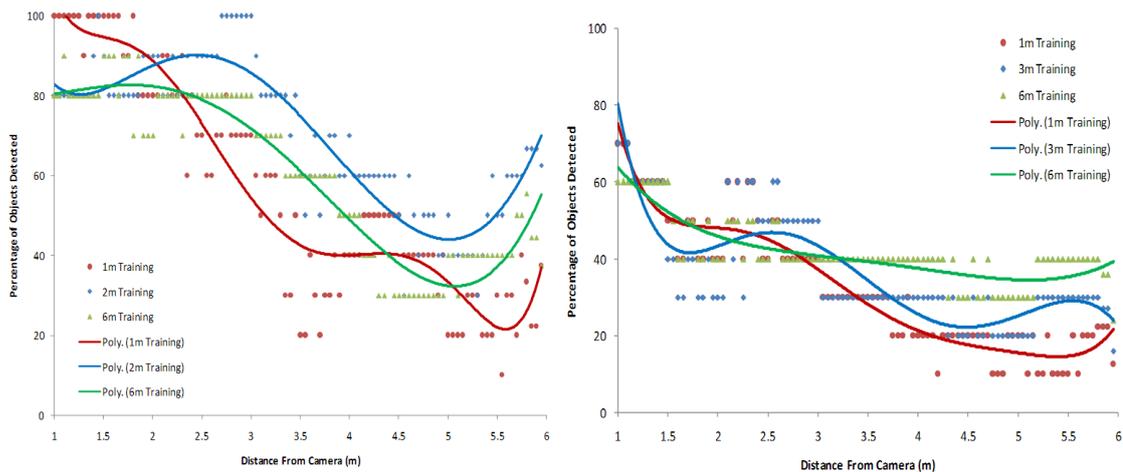


Figure 4.10 (left) Detection performance over all objects for Texture (Mag-Lap) Multidimensional Histograms at 1m, 2m and 6m training distances, (right) Detection performance over all objects for LAB Colour Histograms at 1m, 3m and 6m training distances.

Figure 4.11 (left) shows detection repeatability over all objects at 3m, for SIFT local features, when we rotate in the horizontal plane. The curve is bell-shaped and shows a sharp fall off in repeatability as we rotate objects away from 0° . Around 20° rotation the repeatability falls to around 40% on average, for all objects. This means only 40% of the original training image features are still being detected. This performance is object dependant. For example, the book object has repeatability greater than 40% between -40° and 40° , peaking at 10° (65.25%). In contrast, the barrel's repeatability is only ever above 40% at 10° , where it reaches its peak (42.11%).

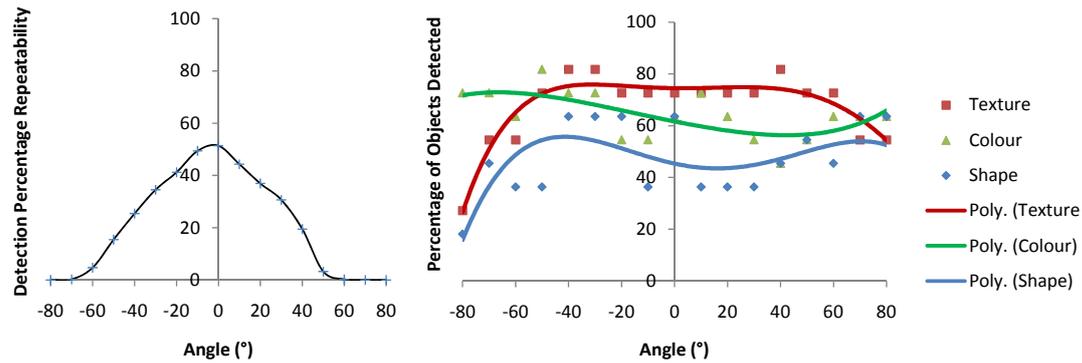


Figure 4.11 (left) Detection percentage repeatability of SIFT (DoG) over all objects when varying 3D rotation angle, (right) Detection performance over all objects when varying 3D rotation angle, for Texture, Colour and Shape, at 3m training and test distances.

Figure 4.11 (right) shows the percentage of objects detected at 3m, for the remaining three algorithms. For both Texture and Shape the curves have a bell-shaped trend, similar to SIFT. The colour algorithm varies between 80% and 50% of objects detected, but does not show a decreasing tendency with rotation.

4.5 Discussion

This section discusses issues arising from the Scale and Rotation experiments in section 4.4.

4.5.1 Research Question R1

For local feature algorithms, the drawback of a single-scale algorithm becomes apparent, as features are only repeatably detected and matched around the training image scale. The scale where we extract our smart objects appearance makes a big difference and to enable detection throughout the whole scale range for an object such as the Mediacube we need to match features detected from training images at approximately each meter (replicating a multi-scale algorithm).

The most repeatable interest point detection occurred with the multi-scale algorithm; however, as can be seen in Figure 4.5 (left) the Harris-Laplace multi-scale algorithm detects many more features than other Harris-based algorithms, so consequently has a higher processing cost. The additional number of descriptors to match also causes a higher probability of mismatches with large scale errors.

Both the number of interest points returned by the detectors and the number of mismatches is important, as wide-baseline detection approaches based on feature correspondences require enough correct matches to enable pose calculation when using robust approaches such as RANSAC [Fischler and Bolles 1981]. As RANSAC easily fails with $>50\%$ mismatches, there is a trade-off between having enough interest points to ensure pose calculation, and number of mis-matches.

Our results agree with those presented by Mikolajczyk et al. [Mikolajczyk, Tuytelaars et al. 2005] which showed that scale-invariant Harris-Laplace performs better than Harris-Affine with large scale changes. This is illustrated with the reduced Harris-Affine performance near 1m in Figure 4.5 (right).

Figure 4.7 (left) shows it is possible to make use of matching to multiple training images to recover object rotation. However, we need training images with at least 60° increments for rotation-variant SIFT descriptors to perform equal to rotation-invariant descriptors. This increases the number of training images we must match by at least 6 for each 3D rotation of the object.

As can be seen when comparing Figure 4.7 (left) and Figure 4.8 (right), the performance of algorithms under affine transformations (such as 3D rotation) is very different to 2D rotation in the camera plane. The use of rotation-invariant descriptors only provides invariance to 2D rotation in the plane of the camera. For 3D rotation multiple object viewpoints must be stored and matched, with a consequent increase in runtime.

Unfortunately, when detecting objects moving around a large environment in the real world, objects are likely to move in multiple axes simultaneously. As features disappear and re-appear as an object is manipulated or moved, multiple viewpoints are required to detect the object in all poses. For example, in Figure 4.8 (right) the matching performance with the Mediacube object is only above 50% between the angles of -25° to $+35^\circ$, which is equal to a 60° useful viewing angle (centred roughly on the 0° training orientation). This result agrees well with those presented by Mikolajczyk and Schmid [Mikolajczyk and Schmid 2005] and Lowe [Lowe 2004], who recommend a maximum rotation of 60° between viewpoints for 3D object detection. This would require 14 viewpoints to describe a 3D object with a viewing sphere using equally spaced 60° polar intervals (26 for 60° latitude and longitude intervals). The use of non-scale invariant detector and rotation-variant descriptors multiplies this requirement by another 6, for a total of 84 training images to match for just single-scale detection. In contrast, when using scale (or affine) invariant detectors and rotation-invariant descriptors a full viewing sphere only requires training and matching the 14 viewpoints.

4.5.2 Research Question R2 to R4

The results from the experiments to address R2 to R4 show it is important to consider scale and rotation effects when designing a detection system with multiple cues. For example, for scale, we learned that different algorithms perform best when trained at different distances. Shape performs best when trained at 1m, Local Features and Texture are best at 2m and Colour performs best at 6m.

Similarly, the algorithms perform differently when we rotate objects. Here, colour does not exhibit a strong decreasing tendency with rotation, suggesting that one viewpoint of a uniformly colourful object may be enough to detect the object in any pose, and only a small number of viewpoints are required for 3D objects with non-uniform surfaces (Swain and Ballard recommend 6 viewpoints [Swain and Ballard 1991]). In contrast, the more bell-shaped curves for the other methods indicate that to detect an object in any pose we need to extract information from many more viewpoints.

These results support both our starting assumption that different detection methods behave differently and hence the argument for using multiple detection methods.

The fact that detection performance results when scaling and rotating objects were often not close to 100% around the distance or angle where we train our detection system also suggest that some of the objects were either not detected by a particular method or there is a large inter-object detection performance variability with the algorithms. This finding is investigated further in Chapter 5.

4.6 Conclusion

In this chapter we conducted an experimental study to understand impact of object scale and rotation on different detection methods. We first studied in-depth the impact of scale and rotation on local feature algorithms, finding that scale and rotation invariant algorithms provide performance benefits over single scale and rotation-variant algorithms. This enables us to use invariant algorithms to detect objects without loss of discrimination, hence, we chose to use the popular SIFT algorithm in our work to detect objects with the features cue.

From the second set of experiments we found that scale and rotation has a large impact on detection performance. An object's appearance changes as it is rotated in 3D, but the change can be much greater than with scaling, as whole surfaces can appear or disappear. This has important implications for our approach, and we found that with the exception of the colour method we need to train our algorithms with multiple object viewpoints to detect an object in any pose. Similarly, we learned that detection performance varies when the algorithm is trained at different distances and found the best training distances for each algorithm based on our dataset. This knowledge is significant as it enables our framework to extract additional appearance knowledge from the camera image when an object is at the best training distance.

Chapter 5 Cooperative Detection

The experiments presented in Chapter 4 demonstrated that it is possible for a camera to detect objects at a range of scales and orientations by using the natural appearance cues of colour, texture, shape and features on objects. These experiments were performed using training and test images with plain backgrounds. However, cluttered real world environments with distractions are a large problem for traditional computer vision approaches and can negatively impact detection performance. For example, if two identical objects appear in the camera image, there is no way for the camera system to determine which is the correct object and which is the distraction on its own.

In this chapter we look at how cooperation between a smart object and the vision-based detection system allows the sensing capability of objects to be used in the detection process. To achieve this we perform a study exploring cooperative detection between the camera system and smart object. Specifically, we analyse the increase in detection performance achieved when using movement sensing in the smart object.

5.1 Video Test Library

Synchronised sensor data and video of each object moving in a cluttered lab environment was captured for the testing library. The objects and equipment used was identical to that described in section 4.3. Video of each object was captured at 10fps for 20 seconds from a fixed camera location at 2.15m height from the ground, with a 25° declination angle. All video was captured with even illumination. The objects were handheld and moved at a constant walking pace (approximately 0.75m/s) in a 15m path shaped like a ‘P’ from first entry through a door 5m from the camera. The object moved around the loop of the ‘P’ towards the camera (the tip is 2m from the camera), then returning to the door (see Figure 5.1 for example images).

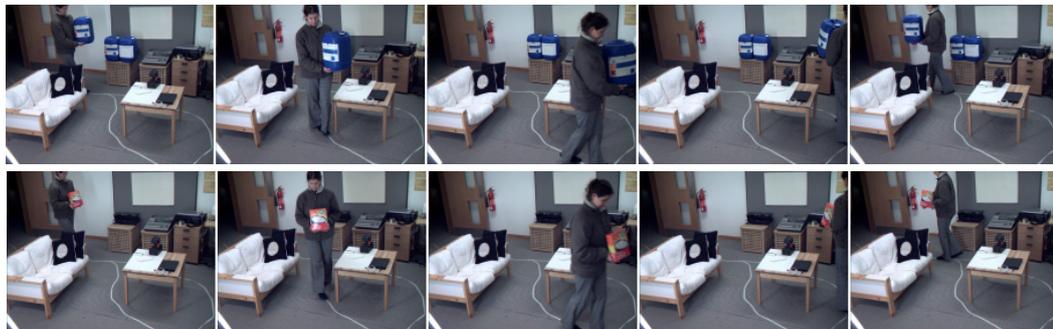


Figure 5.1 Video Test library images of chemical container (top) and cereal box (bottom).

The test videos include challenging detection conditions such as scaling, rotation around the object’s vertical axis and motion blur. Distractions were also present (from other objects or areas of the scene with similar appearances) and the limited camera field of view caused partial occlusion in some frames, hence the videos reflect realistic detection scenarios. Video frames with up to 50% of the object occluded were included in the 200 frame analysed. Each of these 200 frames was manually annotated with a 2D bounding box for a ground truth object location.

A Particle Smart-Its device in the object sent data from an IEE FSR152 force sensor wirelessly at 13ms intervals. Data was abstracted to simple object moving and object not moving events using thresholds on the results of operations performed on the raw sensor values, which allows us to abstract away from the performance of individual sensor types. Here we calculate the mean value over a window of 20 samples (260ms). The threshold was set empirically so the object generated continuous “non-moving” events when placed on a surface and “moving” events when mobile.

5.2 Cooperative Detection Experiments

We hypothesise that there are several measurable benefits from using movement sensing in combination with vision detection:

- H1)** Sensing increases detection performance (i.e. sensing reduces the number of misdetections and outliers).
- H2)** Sensing increases the detection performance of non-discriminative and simple detection methods to the level of the most discriminative and most complex.
- H3)** Sensing increases detection algorithm speed.
- H4)** Sensing reduces the time to first object detection from entry to an environment.

To evaluate our approach using multiple detection methods, we also propose a research question:

- R1)** How do algorithm detection performances change with the individual objects?

5.2.1 Design

To address hypothesis H1 we designed four between-subjects experiments, one for each algorithm. Movement sensing was the independent variable for this experiment set, with two levels: 1) with sensing and 2) without sensing. The dependant variable is the detection performance, which is the percentage of frames where the object is correctly detected.

The second set of experiments addresses hypothesis H2. Here three algorithms each combined with movement sensing are compared to the SIFT local feature algorithm without sensing. The local feature algorithm is used as a benchmark.

To address hypothesis (H3) we compare the runtime of all four algorithms with sensing to the runtime without sensing. Movement was again the independent variable, with two levels: 1) with sensing and 2) without sensing. The dependant variable is the algorithm runtime.

Finally we conduct a set of experiments to address the last hypothesis (H4). Here the time to first detect the object of all four algorithms with sensing is compared to the time to first detect the object without sensing. Movement sensing was again the independent

variable, with two levels: 1) with sensing and 2) without sensing. The dependant variable is the time taken to first detect the object from entry into the environment.

To address the research question R1, we break down the detection results from the first set of experiments, firstly looking at the detection performances of each object, and secondly the detection performance from combinations of cues.

Apparatus was the same as in the section 4.4 experiments.

5.2.2 Training

In order to perform the experiments, we first trained all four algorithms. For each object, then for each detection algorithm, an appearance description was trained using the rotation images in the object appearance library (see section 4.3). As the video test library includes rotation around the object's vertical axis we use multiple viewpoints for detection in the experiments. We assume the bottom surface of objects is not visible, consequently the description was trained with 6 viewpoints from the upper viewing hemisphere with the object at 3m distance from the camera (the centre of our working range) in rotation intervals of 60° around the object's vertical axis. The object ground truth bounding boxes were used to mask the training images when creating the appearance descriptions of the objects.

5.2.3 Procedure

For all experiments the 4 algorithms were run on the 200 frames of each object video in the video test library using the respective object appearance description for detection. As there were multiple viewpoints, the algorithm is run multiple times in each frame with the individual viewpoints. We assume a correct detection from any viewpoint is a detection of the object.

We use the same detection procedure as discussed for the 4 algorithms in section 4.4.3.2.

The detection result from the algorithms was a bounding box for colour histogram, shape context and multi-dimensional histograms, or a set of feature correspondences for the local feature algorithm. For the first three algorithms, correct detection was assumed when the detection bounding box had <50% overlap error with the ground truth bounding box. For local features correct detection was assumed when a minimum of 8 features were matched to the training image using nearest-neighbour Euclidean distance matching and >50% of feature correspondences were correct. Correct correspondences were established based on a manually annotated ground truth homography transformation between the test image and the training image of the object.

Detection algorithm processing time was measured over all 200 frames in the test video using timers with millisecond resolution and the mean time per frame calculated.

The time taken to first detect the object was measured by counting the number of frames before the first detection occurs when the object is visible on entry into the environment. These results were converted to time based on the fixed 10fps frame rate we used for the video by dividing the number of frames by 10. The times were mean averaged for each detection method for the two cases: with sensing and without sensing. Objects with no detections in either or both of the two cases are excluded from the analysis of the respective detection method.

For the multi-cue detection performance, a successful detection was recorded whenever any one of the combination of cues detected the object successfully. This is equivalent to a simple logical OR cue combination.

5.2.4 Results

5.2.4.1 Hypotheses H1 and H2

Figure 5.2 shows the detection performance of four algorithms: SIFT local features, Mag-Lap Texture, Lab Colour and Shape Context. The results presented are mean averaged first for each object, then for each algorithm. It is clearly visible that for all algorithms the use of movement sensing increases detection performance over no sensing. These results support hypothesis H1.

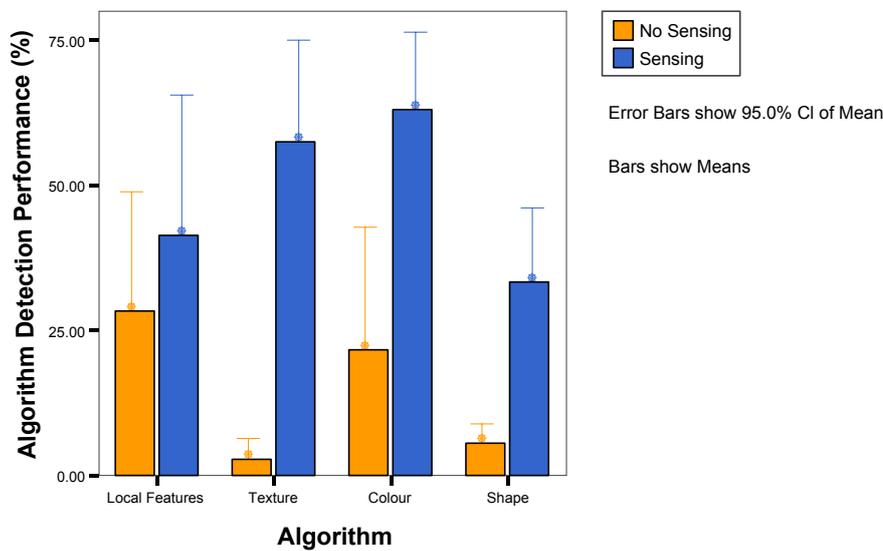


Figure 5.2 Graph of detection algorithm results without sensing (orange) and with movement sensing (blue), averaged over all objects. Error bars show 95% Confidence level (CI) of mean.

Without sensing, the Local Feature algorithm has the highest performance (orange bar on the left) of all algorithms, when averaged over all objects in the test video library. On average, the use of sensing ($M=41.07$, $SE=10.82$) gives a statistically significant improvement in detection results over no-sensing ($M=28.42$, $SE=9.13$) for all objects, with the Local Feature (SIFT) detection algorithm, $t(9)=-3.75$, $p<.05$, $r=.78$.

The Texture algorithm shows the greatest increase in detection performance when using movement sensing. The use of sensing ($M=57.63$, $SE=7.71$) gives a statistically significant improvement in detection results over no-sensing ($M=2.98$, $SE=1.50$) for all objects, with the Texture detection algorithm, $t(9)=-7.24$, $p<.001$, $r=.92$.

The colour algorithm has the highest detection performance of all algorithms when used with sensing. The use of sensing ($M=63.22$, $SE=5.92$) gives a statistically significant improvement in detection results over no-sensing ($M=21.82$, $SE=9.30$) for all objects, with the Lab colour detection algorithm, $t(9)=-6.02$, $p<.001$, $r=.89$.

While the performance of the shape algorithm increases with movement sensing, this algorithm has the lowest mean performance when used with sensing. On average, the

use of sensing ($M=33.03$, $SE=5.69$) gives a statistically significant improvement in detection results over no-sensing ($M=5.52$, $SE=1.52$) for all objects, with the Shape Context detection algorithm, $t(9)=-5.93$, $p<.001$, $r=.89$.

When using movement sensing, all algorithm performances are higher than the performance of local features without sensing. This supports hypothesis H2.

5.2.4.2 Hypothesis H3

Table 5.1 Mean algorithm runtime per frame, averaged over 200 frames

Method	No Sensing (ms)	Movement Sensing (ms)
Local Features	641	605
Texture	648	447
Colour	307	307
Shape	852	288

Table 5.1 shows the mean time per detection for each of the four algorithms, with and without the use of movement sensing. It is clearly visible that for three algorithms (Local features, Texture and Shape) the use of sensing reduces the mean detection time per frame, supporting our hypothesis H3.

5.2.4.3 Hypothesis H4

Table 5.2 shows the mean time to object detection from first entry to the environment for each of the four algorithms, with and without the use of movement sensing. It is clearly visible that for all four algorithms the use of sensing reduces the mean detection time, supporting our hypothesis H4. For all algorithms and all objects, the mean average time before first detection when using sensing is 0.67s, compared to 3.97s without sensing. The use of sensing both reduced the inter-object and the inter-algorithm variability in detection time, as can be seen by the consistently lower scores in standard deviation with sensing.

Overall, the Colour algorithm showed the greatest change with a mean average of 6.78s over all objects before first detection without sensing, while only 0.15s with the use of movement sensing. The lowest change was for the Local Features algorithm, with an average of 1.35s reduction from 2.36s to 1.01s with sensing. The highest variation in detection times between objects was seen when detecting object with the Shape algorithm without sensing ($SD=4.39$), however, shape shows the largest decrease in standard deviation with sensing (to $SD=0.78$).

Table 5.2 Mean time to detection from first entry to the environment

Method	Mean Time, No Sensing (s)	Std Dev, No Sensing	Mean Time, Sensing (s)	Std Dev, Sensing
Local Features	2.36	2.33	1.01	1.73
Texture	2.30	1.48	0.93	0.10
Colour	6.78	1.45	0.15	0.10
Shape	4.44	4.39	0.58	0.78
Mean (s)	3.97		0.67	
Std Dev	2.12		0.39	

5.2.4.4 Research Question R1

The following Figure 5.3 and Figure 5.4 show the detection results with the four detection algorithms for each object in the test video library.

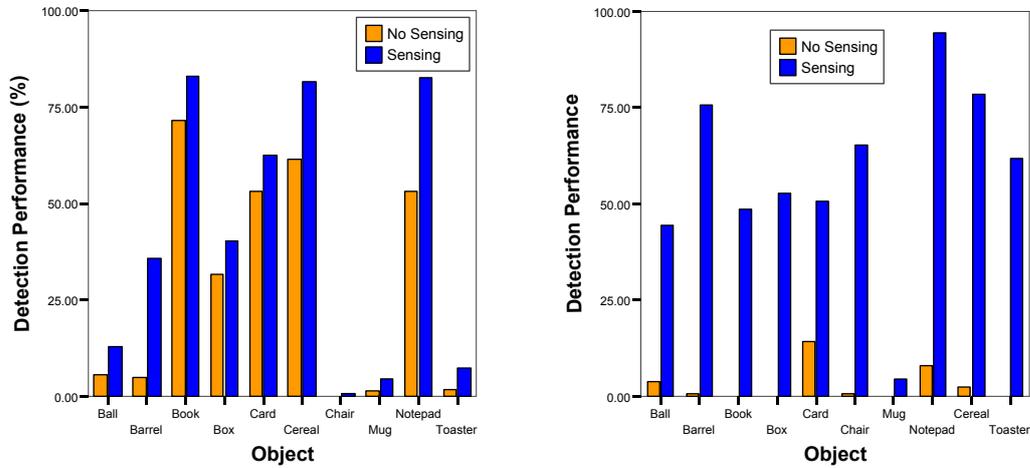


Figure 5.3 (left) SIFT Local Feature Detection Performance with and without sensing for each object in test library, (right) Mag-Lap Texture Detection Performance with and without sensing for each object in test library

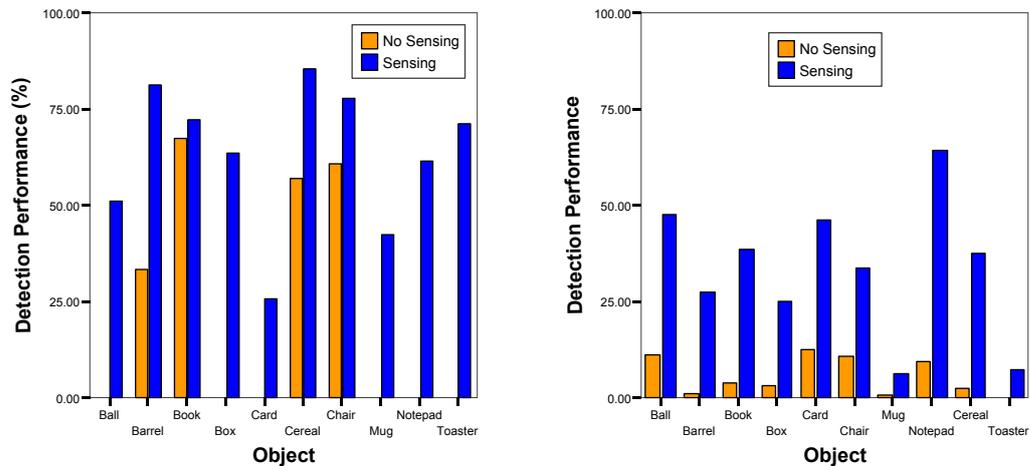


Figure 5.4 (left) Lab Colour Detection Performance with and without sensing for each object in test library, (right) Shape Context Detection Performance with and without sensing for each object in test library

As can be seen in the local features performance in Figure 5.3 (left), the book, notepad and cereal box objects are detected very well, with or without sensing. However, some objects such as the chair, mug or toaster are detected in few video frames.

Although the average texture detection performance shown in Figure 5.3 (right) is the lowest of all algorithms without sensing, all objects except the mug achieve performance around 50% or greater when using movement sensing. When detecting the notepad object, texture also produces the highest detection result for any object and any algorithm using movement sensing (detected in 94.44% of all video frames).

As seen in Figure 5.4 (left), the colour detection performance again varies greatly by object without sensing, similar to local features. Here, barrel, book, chair and cereal objects are detected well, without sensing. However, all objects achieve detection performance higher than 25% with sensing, and this is reflected by colour achieving the highest mean average detection result for all algorithms. The mug in particular has its highest detection performance here, when detecting with the use of movement sensing (42.42% of all video frames).

The shape detection performance shown in Figure 5.4 (right) without sensing is higher on average than texture, however, the average with sensing is the lowest of all algorithms. Only the notepad object achieves detection performance higher than 50% when using sensing.

5.2.4.5 Multiple-Cue Combination

The results from section 5.2.4.4 are used for Table 5.3 and Table 5.4 to rank each detection cue by its detection performance for each object, without and with movement sensing respectively. The Local Features algorithm proved to be the best single cue for 70% of the objects without sensing, while colour is the best single cue for 70% of the objects with movement sensing. However, more importantly, each of the cues we studied was the best algorithm for at least of one of the objects, indicating that there is potential for further improvement in overall detection performance by using multiple cues.

Using the cue ranking from Table 5.3 and Table 5.4, we illustrate the detection performance improvement attained over a single cue when using multiple cues to detect objects in cluttered scenes, such as the video test library (see section 5.1).

The overall detection performance achieved by multi-cue combination, together with the breakdown of cue contribution to this performance figure (shown as column colour) can be seen in Figure 5.5 and Figure 5.6. It is clearly visible that by using a combination of the highest performing cue (lowest colour of the column) together with other cues, the overall detection performance improves for all objects in the study, with or without movement sensing.

Over all objects without sensing the mean average performance increases from 37.89% (with just the highest performing single cue) to 43.47% with all 4 cues, an improvement of 4.96%. The largest performance increase is for the book, which improves from 71.52% to 86.71% (a 15.19% increase in the number of frames detected). The smallest increase is for the Card object (0.62%). The most frequent combination of two cues was Local Features and Shape, ranked as the best two for 50% of all objects.

For objects with movement sensing, the mean detection performance increase was larger (15.38%), with the ball object benefiting from the largest increase (of 26.77%) to 77.78% of frames detected. The smallest improvement was for the Card object (8.13%). The most frequent combination of two cues was Colour and Texture, ranked as the best two for 40% of all objects.

Table 5.3 Ranking of Cues by Detection Results for Each Object, No Movement Sensing

	Best Cue		2 nd Best Cue		3 rd Best Cue		Worst Cue	
Object	Method	Result (%)	Method	Result (%)	Method	Result (%)	Method	Result (%)
Ball	Shape	11.11	Local Features	5.55	Texture	3.70	Colour	0.00
Barrel	Colour	28.28	Local Features	4.76	Shape	1.19	Texture	0.59
Book	Local Features	71.52	Colour	67.17	Shape	3.80	Texture	0.00
Box	Local Features	31.74	Shape	2.99	Colour	0.00	Texture	0.00
Card	Local Features	53.13	Texture	14.38	Shape	12.50	Colour	0.00
Cereal	Local Features	61.35	Colour	57.06	Shape	2.45	Texture	2.45
Chair	Colour	60.60	Shape	10.76	Texture	0.63	Local Features	0.00
Mug	Local Features	1.27	Shape	0.63	Colour	0.00	Texture	0.00
Notepad	Local Features	53.09	Shape	9.26	Texture	8.02	Colour	0.00
Toaster	Local Features	1.79	Shape	0.60	Colour	0.00	Texture	0.00

Table 5.4 Ranking of Cues by Detection Results for Each Object, with Movement Sensing

	Best Cue		2 nd Best Cue		3 rd Best Cue		Worst Cue	
Object	Method	Result (%)	Method	Result (%)	Method	Result (%)	Method	Result (%)
Ball	Colour	51.01	Shape	47.53	Texture	44.44	Local Features	12.96
Barrel	Colour	81.31	Texture	75.60	Local Features	35.71	Shape	27.38
Book	Local Features	82.91	Colour	72.22	Texture	48.43	Shape	38.61
Box	Colour	63.63	Texture	52.69	Local Features	40.12	Shape	25.15
Card	Local Features	62.50	Texture	50.63	Shape	46.25	Colour	25.75
Cereal	Colour	85.27	Local Features	81.60	Texture	78.53	Shape	37.42
Chair	Colour	77.78	Texture	65.19	Shape	33.54	Local Features	0.63
Mug	Colour	42.42	Shape	6.33	Local Features	4.43	Texture	4.43
Notepad	Texture	94.44	Local Features	82.72	Shape	64.20	Colour	61.61
Toaster	Colour	71.21	Texture	61.90	Local Features	7.14	Shape	6.55

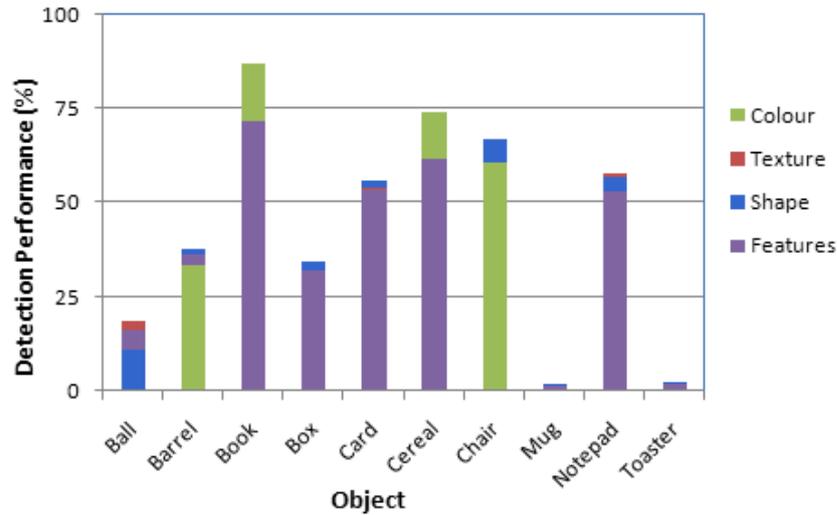


Figure 5.5 Detection Performance improvement using multiple cues, No Sensing

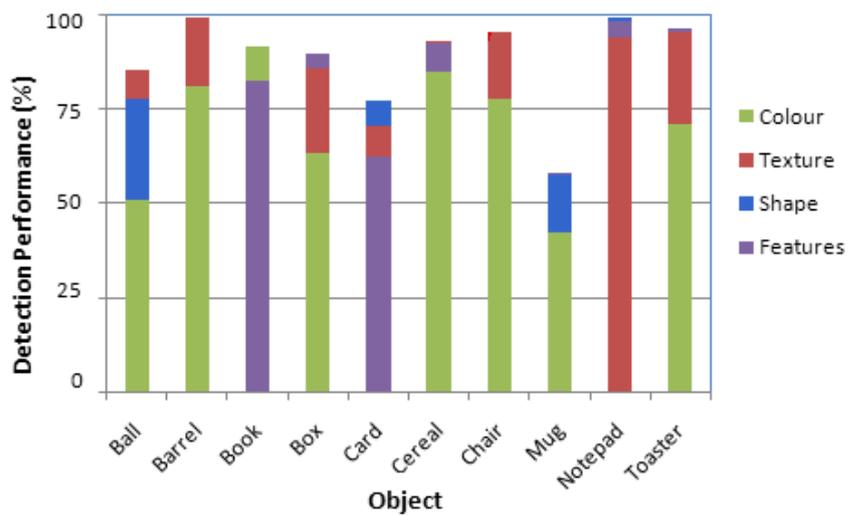


Figure 5.6 Detection Performance improvement using multiple cues and Movement Sensing

Using a combination of the three best cues produces much less additional benefit. For objects without sensing a third cue only produced a mean increase of 0.62% over the use of the two best cues, with the largest increase being an additional 2.47% of detection performance for the ball. Objects with movement sensing again benefit more, however the increase is again much less than the improvement gained over a single cue, when using the two best cues. The mean improvement is 2.10%, with the ball again benefiting the most with an 8.03% increase in performance. The Barrel, Book and Chair did not increase in performance by adding a third cue.

The addition of a fourth cue did not improve detection performance in any object, with or without movement sensing.

5.3 Discussion

This section discusses issues arising from the Cooperative Detection experiments in section 5.2.

5.3.1 Single Cue Detection

5.3.1.1 Hypothesis H1

For hypothesis H1 we compared the means of detection performances. The results indicate that performance significantly increases in all algorithms when detecting smart objects with basic movement sensing. Consequently, in our experiments the use of unobtrusively embedded sensing makes the detection process more robust. We believe this performance increase can be generalised to other detection algorithms, as it was observed for all four cues of the object's appearance. Consequently we suggest that smart objects should try and include at least one type of movement sensor. A range of basic movement sensors are available, such as accelerometers, gyroscopes, ball switches, tilt switches, and force sensors.

5.3.1.2 Hypothesis H2

To address hypothesis H2 detection performance of less complex algorithms (texture, colour and shape) with sensing was compared to local features without sensing. The results suggest that less complex detection algorithms used with sensing achieve better performance than local features. Consequently, smart objects with movement sensors can use a less complex method to be detected with no loss in performance. The runtime of less complex detection methods also tends to be faster, as seen in Table 5.1.

5.3.1.3 Hypothesis H3

To address hypothesis H3 we compared the runtime of the four detection algorithms with sensing against runtime without sensing. While the absolute runtimes are a function of the algorithm implementation, CPU speed and image size being processed, we showed that three algorithms (texture, shape, local features) increased measurably in speed with the use of context information from movement sensing. This movement mask constrains the detection by enabling areas without movement to be discarded from the processing, hence increasing overall processing speed. The Lab colour histogram implementation did not increase in speed as the current implementation of the method only uses the movement sensing events to mask the result of the colour detection step into moving and non-moving areas. This mask operation is performed in under 1ms. Re-implementation of the colour detection algorithm would allow the movement mask to be taken into account.

Algorithm speed could be increased further for all algorithms by using information from previous frames to predict an object's location in the current frame, or by using guided matching where location hypotheses from different detection method are fused (both methods constrain the search space with a region of interest).

5.3.1.4 Hypothesis H4

To address hypothesis H4 we compared the time to first detect an object following entry to the environment for each of the four algorithms with sensing, against time to detection without sensing. The results shown in Table 5.2 indicate that sensing provides additional benefit by enabling objects to be detected faster than without sensing. For example, the 0.15s time for Colour detection and low standard deviation of 0.10 when using sensing indicates that objects are typically detected within the first few frames of becoming visible to the camera (each frame is captured at 0.10s intervals).

The reduction in detection times is likely due to two reasons. The first is that the use of movement sensing constrains the detection process, allowing a faster overall detection. However, if we compare the results against the detection performance results in Figure 5.2, we infer that a secondary reason for the reduction in detection time seen in Table 5.2 is likely due to the absolute overall increase in detection performance with sensing. Here, the objects are detected in more frames overall and this effect occurs in all the videos; hence this will contribute to an average reduction in detection time.

5.3.1.5 Research question R1

We posed research question R1 to look at how our object dataset (shown in Figure 4.1) was detected with different algorithms. The test videos include challenging detection conditions (scaling, rotation, distractions, motion blur and partial occlusion), hence the results reflect expected real-world performance. Location prediction between frames with Kalman or Particle filters is expected to improve all results further.

In the local features performance, the chair, mug and toaster have few detections because of their appearance. The chair has plenty of corner features, however many look identical due to the geometrically repetitive structure. Similarly, objects with repeating patterns on their surface, such as the ball, also exhibit high numbers of mismatches. Local features use the area around each detected corner for calculating a descriptor. Consequently, the structure of the chair again causes poor matching results, as features detected at object boundaries include a lot of background. Plain objects such as the toaster and small objects, such as the mug only have a small number of features, so are difficult to detect reliably.

For texture and shape, all objects had a poor performance when used without sensing due to the cluttered environment in the video test library. Here other objects and surfaces with different appearances but similar amounts texture or similar shapes distract the algorithms. Sensing constrains the search area, causing less distraction and higher performance.

The colour algorithm performs best with uniform and brightly coloured object appearances. The low detection rate of the barrel without sensing (28.28% of frames) is due to other chemical containers in the background distracting the algorithm. Movement sensing masks the distraction and increases detection to 81.31% of the video frames.

Table 5.5 Guidance for which method to use, based on object appearance type

Object Appearance Type	Best Method	Worst Method
Small Objects	Colour	Texture / Features
Large Objects	Features / Texture / Shape	
Plain Objects or Objects with Repeating Pattern	Colour / Shape	Texture / Features
Non-Repeating Pattern or Textured Objects	Features / Texture	Colour / Shape
Saturated Colourful Objects	Colour	
Muted Colour or Black and White Objects	Features / Texture / Shape	Colour
Simple Geometry	Depends on appearance.	
Complex Geometry or Geometry with Holes	Colour / Texture / Shape	Features

Colour with sensing was the only algorithm to detect the white mug in more than 40% of the video frames. Movement sensing masked the static white wall, responsible for distraction. The small size and plain appearance of the mug made it a particularly challenging object, almost never detected with other algorithms.

These results underline the fact that a single algorithm cannot reliably detect all objects, and validates our approach using multiple cues and method selection.

In practice the results suggest when certain algorithms should be used to achieve the best detection performance. This knowledge can be incorporated into our method selection step, however, the suggestions do not take into account external factors such as the object's context (for example, its background).

5.3.2 Multi-Cue Combination

The performance benefits from multi-cue detection are shown clearly by Figure 5.5 and Figure 5.6. As the cue rankings changed with objects and performance benefits are seen when using multiple cue detection, we infer that the four cues we use in detection are complimentary.

When using all 4 cues without movement sensing the objects are on average only detected in 43.47% of all the video frames, which is a very low detection performance for a system which aims to be deployed in a real-world environment. These detection results are also significantly lower than the results reported for the algorithms in [Swain and Ballard 1991; Schiele and Crowley 2000; Belongie, Malik et al. 2002; Lowe 2004], however, the reported results were typically for objects on plain backgrounds or with dissimilar distracting objects. The use of movement sensing with multiple-cues on our approach makes the detection robust to the clutter and distraction seen in "realistic" environments. Over all objects the mean average detection performance is now increased to 88.72% of video frames, when using all 4 cues.

All objects receive an increase in detection performance by using the two best cues for detection, by a mean average of 15.38% when used with movement sensing. However, increasing the number of cues considered beyond two gives little additional benefit (2.10% for 3 cues) and the addition of a fourth cue did not improve detection performance in any object..

The multi-cue results presented are a best-case scenario, where appearance knowledge is available for all cues and the ranking of cues is known a-priori. We may not initially know the ranking of cues in new objects to achieve best detection performance, however, in practice the detection system can maintain detection metrics for each object and method combination. This valuable knowledge can be re-embedded into the smart object so it is not lost when the object leaves the environment.

Unless multiple cues can be implemented to run in parallel, multi-cue detection is always a trade-off. Each extra detection algorithm causes a corresponding increase in total run-time per camera frame. Consequently, the detection performance increase must be balanced against the processing required in the projector-camera system. Table 5.1 shows that algorithm runtime can be decreased by the use of movement sensing for 3 of the 4 algorithms, and comparing Figure 5.5 and Figure 5.6 we see a larger effect from multi-cue detection when used with sensing. This suggests that when a smart object contains a movement sensor and has appearance knowledge for a minimum of two cues, the detection system has the potential to run both algorithms and combine the results

with little penalty in terms of runtime, to achieve a further performance increase over sensing alone.

Sequential combination of cues also allows location hypotheses from the first cue to be used when processing the second. For example, if the colour detection method returns two possible locations when searching for a blue chemical container, we can choose to constrain our processing of the second cue to just these image areas, giving both a reduction in runtime and increased chance of detection with the second cue. However, detection using sequential cue combination is more liable to exhibit catastrophic failure, due to detection failures in early cue methods. Hence democratic integration (parallel combination) could be used for robustness and sequential combination for accuracy or speed.

5.3.3 Limitation of Movement Sensing

The results support all three of our hypotheses on the benefits of the use of movement sensing in combination with vision-based detection. However, there are some limitations to an approach only involving movement. To detect static objects, the only way we can use movement sensing is to exclude moving areas in the image from the detection process. While this confers some similarities to the scenario where there is a moving object and static background, the detection performance will vary between performance without sensing and with sensing, depending on the size of the moving area in the image.

In this case, greater knowledge of the object's context from other sensors may help us detect the object, for example, using an embedded electronic compass or any other available location system. If the object has external light sensors the structured light approaches taken by [Lee, Hudson et al. 2005; Summet and Sukthankar 2005; Raskar, Nii et al. 2007] would also be possible.

5.3.4 Use of Sensing in smart objects for Pose Calculation

Following the initial detection process, the location and orientation (the pose) of a smart object must be calculated before a projection can be established. The space that the Projector-Camera system and smart objects exist in is modelled in a 3D coordinate system with a single world origin. Local coordinate systems exist for all projectors, cameras and objects. For example, objects can be modelled with the centre of the object as their local origin and their front surface parallel to the X,Y plane, however, with known transformations we can convert between any arbitrary coordinate systems. Pose calculation aims to recover all 6 degrees of freedom of the object to camera coordinate system transformation (see Figure 4.4, section 4.4). This comprises the location of the object origin in X,Y,Z camera coordinate system (t_x, t_y, t_z) and rotation of the object local coordinate system relative to the camera X,Y,Z axes (r_x, r_y, r_z).

If a new object enters the environment it can be in an arbitrary location and orientation. For example, a user who enters with a cubic object held in the palm of their hand could have any of its 6 surfaces upwards. The location t_x, t_y, t_z can be estimated both from the centre of detection in camera coordinate system and the apparent size of the detection area combined with the known surface size from the 3D model stored in the smart object.

We model the orientation transformation as P, where $r_x, r_y, r_z \in [0, 2\pi)$. In order to determine P we have to search in \mathfrak{R}^3 , a search space of $[0, 2\pi) * [0, 2\pi) * [0, 2\pi)$

possibilities. However, if the object contains a 3D accelerometer sensor, we can make use of the sensing ability to help us recover the object pose.

3D Accelerometers can be easily calibrated to measure their orientation relative to gravity using methods such as that proposed by Krohn et al. [Krohn, Beigl et al. 2005]. The smart object can also use machine learning methods and be trained to detect its orientation using the method proposed by Van Laerhoeven et al. for smart cubes, with little cost (20 extra multiplication and 5 addition operations per 3D accelerometer reading) [VanLaerhoven, Villar et al. 2003].

We assume the camera orientation relative to gravity is known, either from a pan and tilt unit with known mounting orientation, or by using an inclinometer sensor, as proposed for handheld projectors by Raskar et al. [Raskar, Beardsley et al. 2006]. These methods therefore allow direct calculation of 2 of the 3 rotation parameters of the vector P directly from the accelerometers. The third rotation (rotation around the gravity vector) cannot be detected with accelerometers. Using this knowledge we reduce our search space from \mathfrak{R}^3 to \mathfrak{R} , as we only need to recover 1 unknown rotation in P , a search space of just $[0, 2\Pi)$.

If an object contains electronic compasses or gyroscopes, attempts could also be made to detect the third rotation axis. However, a minimum of 3 orthogonal compasses would be required as orientation errors increase significantly with any tilt.

5.4 Conclusion

In this chapter we conducted an experimental study, to explore cooperative detection between the visual detection system and smart object.

We found that detection performance increases when using basic movement sensing with all four detection algorithms. Movement sensing constrains the search space, reducing distractions from the cluttered real-world environment. The improvement was observed for all algorithms, suggesting this can be generalised to other detection algorithms. This is a significant result for our framework, as it means any objects with embedded movement sensors can achieve more likely visual detection by cooperating with the projector-camera systems and sharing their sensor information.

In addition, we found that with the use of movement sensing, simple algorithms achieve similar or better detection performance to complex detection algorithms and that the runtime of 3 of the 4 detection algorithms in the study was reduced. This has important implications for the projector-camera systems in our framework, as it means overall detection performance can be maintained or improved either while reducing the amount of processing power required for detection, or that detection speed can be increased. This allows more flexibility in object detection. For example, when objects share their sensing information we can increase the chances of detecting and tracking a moving object by using the fastest algorithms. Similarly, we can either increase robustness in detection or detect more than one object simultaneously by running multiple detection methods in parallel, with no loss in detection speed.

None of the algorithms had a high detection performance for all objects in our dataset. This is another significant result for our framework, as it underlines the fact that a single cue cannot reliably detect all objects. Consequently, this indicates it is worthwhile supporting multiple detection methods and validates the approach we propose in the Cooperative Augmentation framework.

In this study the most challenging objects to detect were small, plain objects, which were best detected with colour and movement sensing. The best objects were highly textured (a non-repeating pattern or unique features) or had uniform saturated colour.

Multi-cue detection was demonstrated to improve detection performance for all objects when using the best 2 cues; however, little improvement was seen when using more than 2. This effect is again down to the variable natural appearance of objects.

Improvements in detection performance were most pronounced when using movement sensing. The time saved in detection when using movement sensing can be traded off with processing of additional cues, enabling a projector-camera system in our framework to improve detection performance while maintaining the original single cue detection speed.

In conclusion, the results presented in this chapter indicate that the combination of different sensing modalities is important in detection. Smart objects that cooperate with projector-camera systems and share their embedded sensing information can benefit directly with faster or more robust detection and pose calculation.

These findings are used to inform the design of a system architecture for Cooperative Augmentation in Chapter 6.

Chapter 6 System Architecture

This chapter describes a system architecture which serves to validate the feasibility of the Cooperative Augmentation approach proposed in Chapter 3.

The design of the architecture integrates all components of the Cooperative Augmentation framework and is directly informed by the results from the visual detection experiments in Chapter 4 and Chapter 5. This enables the architecture to serve as a platform for the demonstration applications presented in Chapter 7.

We first discuss the design itself, in terms of the components which comprise the architecture, followed by the implementation details, and then discuss the issues which arise in practice when implementing the design with a set of smart objects and steerable projector-camera systems. Finally, three series of experiments are presented to characterise the performance of the architecture in terms of the detection method memory requirements in the smart object, the accuracy of pose calculation, the magnitude of jitter in pose calculation and a combined system evaluation of the overall projection accuracy on the smart objects.

6.1 Architecture Design Overview

The architecture is designed to enable detection, tracking and interactive projection on a smart object's surfaces. As discussed in section 3.3, the framework aims to enable serendipitous use of any projector and camera hardware distributed in the environment in the augmentation process. Hence, the architecture is designed to support multiple projector and camera systems by using a distributed client-server design.

To achieve this distributed design we assume our system has four aspects of knowledge about the physical hardware:

1. Location and orientation of all projectors and cameras, as discussed in section 3.3
2. Calibration information about the intrinsic optical characteristics of the cameras and projectors, as discussed in sections 6.2.1 and 6.2.2.
3. Calibration of any pan and tilt unit used in a steerable projector-camera system, as discussed in section 6.2.3.
4. Knowledge of the pairing of distributed projector and camera hardware in the environment to form projector-camera systems.

The cooperative augmentation conceptual framework maps to a physical structure of mobile smart objects and projector-camera systems in a ubiquitous computing environment, as seen in Figure 6.1. A discovery mechanism allows smart objects to

initially discover services offered by projector-camera systems, and register for the services. We assume the objects use an existing service discovery mechanism, so this component will not be discussed further in this thesis.

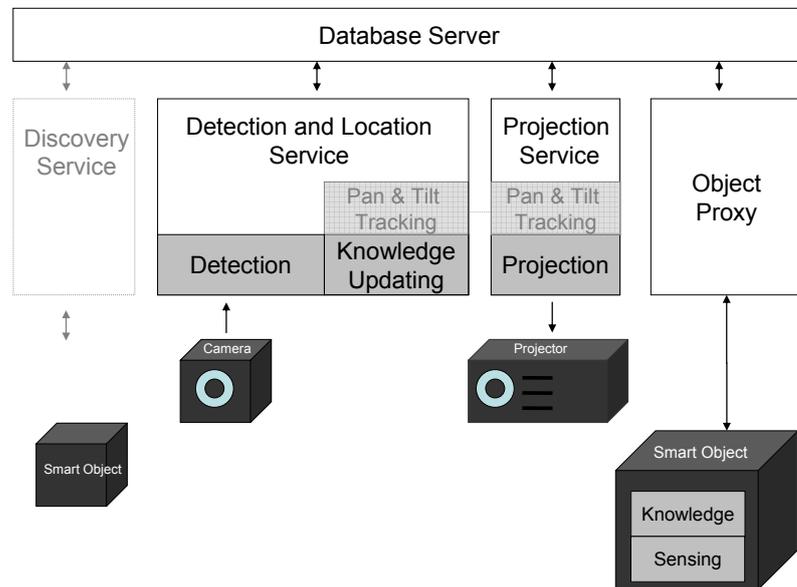


Figure 6.1 Distributed System Architecture Overview

A detection and tracking service exists for each camera in the environment. The service is composed of the two core components Detection and Knowledge updating. If a camera is attached to pan and tilt hardware, Pan and Tilt Tracking forms the third optional component of the service.

A projection service exists for each projector in the environment. The service is composed of the core projection component and a second optional Pan and Tilt Tracking component if the projector is attached to pan and tilt hardware.

The detection and projection services may share a Pan and Tilt Tracking component if they are part of the same physical steerable projector-camera system.

Physical devices such as smart objects are represented by proxy components in the system architecture. We assume proxies, services and applications in our architecture send and receive messages in a common message format, using a common protocol on the network. This design ensures that the system architecture can be distributed effectively while abstracting from the hardware and allowing proxy components to be easily exchanged.

The Database Server component is a single world model maintained on the network, supporting services and applications on top of its model. The world model contains knowledge of all projectors, cameras and smart objects in the “environment”. In the real-world this “environment” maps to an area of physical space visible to the projector-camera systems contained within it, which supports projected displays on smart objects.

The Object Proxy component is responsible for maintaining coordination of state between a physical smart object and the cached knowledge in the Database Server component.

6.1.1 Architecture Novelty

The architecture components themselves are based on several well-established concepts discussed in related work Chapter 2, such as the natural appearance detection and pose calculation algorithms used in the detection component, or the geometric, photometric and colorimetric correction algorithms in the projection component. Additionally, our architecture incorporates the six novel aspects described below.

The first is the embedding of the knowledge required to achieve a display in to the object. This removes the need to store information about all possible objects which can enter the environment, or manually update the system whenever a new object appears. This avoids creation of large databases of objects which must be searched in detection, reducing the possibility of misdetection. Instead, the initial smart object registration step makes the object detection process simpler, as the system knows exactly which objects are present in the environment and hence, what to search for.

The second is that abstraction enables flexibility in our architecture. By abstracting the detection and projection process to services in the environment we enable use by any type of smart object and any projector or camera hardware. Similarly, by abstracting object movement sensors to generic moving or non-moving events we enable use of any type of sensor able to detect movement. For example, accelerometers, gyros, ball switches, tilt switches or force sensors which detect pick-up and put-down events. Here we only care about detecting basic motion, allowing us to abstract away from the specific performance or calibration requirements of individual sensors.

The third is that our system is flexible and adaptable to the knowledge contained in the object due to a dynamic tailoring process. This process occurs in two situations: firstly, in dynamic multi-cue detection algorithm selection based on knowledge embodied in the smart object, its sensing capabilities, the object context and its current detection status. Secondly in dynamic projector geometry correction algorithm selection based on knowledge of object geometry embodied in the smart object.

The fourth is that objects monitor their own appearance and form via embedded sensing and update the projector camera system when these change. For example, an articulated object such as a book changes both appearance and form when opened, but with embedded sensing it can detect the opening and automatically updates the projector-camera system with a new appearance description and 3D model geometry; hence, tracking continues uninterrupted.

The fifth is a dynamic projector and camera pairing method to support multiple projectors and cameras distributed in the environment. This process allows serendipitous cooperation between projector and camera services in our architecture to achieve the best display possible on smart objects in the environment.

The final aspect is that over time camera systems automatically extract more appearance knowledge about objects and re-embed this into the smart object. Even if the object is already detected reliably with one detection method, extracting more knowledge is beneficial as the environment can also change. For example, distracting objects are introduced with similar appearances, the wall is painted a different colour, or the object is simply taken to another room. Similarly, new detection algorithms can be easily deployed in the projector-camera system, as the object appearance knowledge will be automatically updated with the new appearance information.

6.2 Architecture Components

The components of the architecture are described in more detail in the following sections.

6.2.1 Detection and Tracking

The design of the detection system architecture is informed directly from the results presented in Chapter 4 and Chapter 5. We design multi-cue detection system with the different detection methods (shown in Table 6.1), corresponding to different appearance cues of the object (colour, texture, shape and features of objects).

Table 6.1 Appearance knowledge levels and detection methods with associated processing cost

Appearance Knowledge	Detection Method	Discriminative Power	Cost in Time
Colour	Colour histogram comparison	Low	Low
Texture	Multidimensional Receptive Field Histograms	Medium	Medium
Shape	Contour detection and Shape Context	Medium	Medium
Local Features	Interest point detection and feature descriptor comparison	High	High

These methods form a flexible layered detection process that allows an object to enter the environment with different levels of appearance knowledge. As we descend the table, the power of the detection methods to discriminate between objects with similar appearances increases, however, at the cost of increased processing time (due to increasing algorithm complexity). We consider higher discriminative methods to hold more knowledge about the object.

6.2.1.1 Multi-Cue Detection Method Selection

The detection method selection step forms a novel part of the visual detection pipeline shown in Figure 6.2. Here, following each camera frame acquisition the method selection step is performed based directly on the appearance knowledge embedded in the object, its embedded sensing capability and its visual context obtained from the background model.

If an object holds only knowledge of a single cue, the respective natural appearance algorithm is automatically executed. Additional appearance knowledge can be subsequently extracted by the knowledge updating component once the object is detected and its pose calculated, as discussed in section 6.2.7.

Where an object holds appearance knowledge of more than one cue, the design allows either a single cue to be used, or multiple cues to be fused for improved detection performance (as demonstrated in section 5.2). However, detection method selection is always a trade-off, as processing multiple methods sequentially and the more discriminative individual methods (such as local features) both share the cost of increased processing requirements.

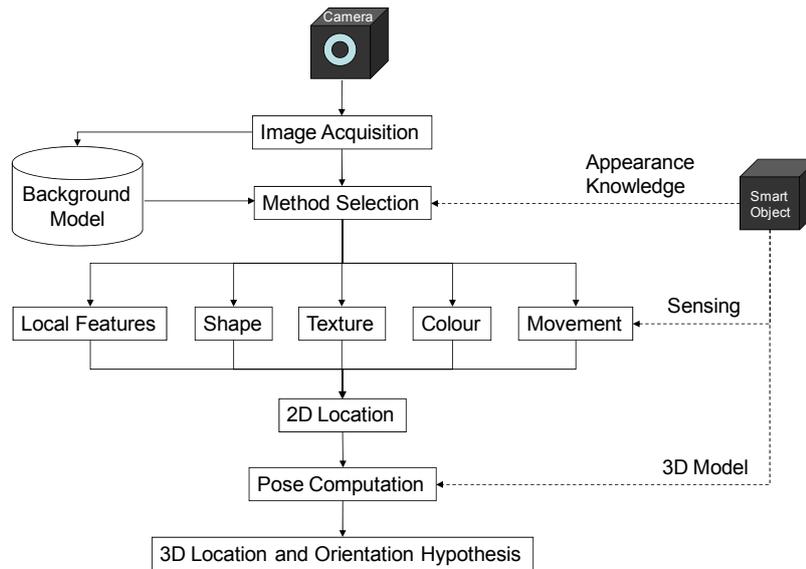


Figure 6.2 Detection method selection based on smart object knowledge

To inform the architecture of the cues most likely to detect the object we maintain detection metrics, which are updated each time a detection process is executed. We store total and mean average algorithm runtime and detection performance histories. The detection performance metrics are only used following first detection of an object and only updated when the object is potentially visible (i.e. its last location was inside the camera FOV). These cue performance metrics also re-embedded into each object to allow use of the accumulated knowledge in other environments.

When using only a single cue we can choose to prioritise one of three aspects of the detection process:

1. Detection speed, by using the fastest method when objects are moving
2. Accuracy, by using the detection with the highest detection rate when objects are static.
3. Robustness, by using the most discriminative (least abstract) information to increase the probability of detection when distractions are present.

When multiple cues are available we need to decide which cues should be combined. To perform this method selection we rank the detection methods based on suitability for detecting a particular object. The ranking is based on three aspects:

1. When an object is moving we rank algorithms with shorter average runtimes higher, whereas algorithms with higher detection performance are ranked higher for static objects.
2. We take the object's context into account, by looking at the background model and the object's movement sensing capabilities. For example, we can directly compare the object's colour histogram with the colour histogram of the area of the background model currently visible to the camera. If the histograms are similar we would not select the colour method unless the object had movement sensors and was mobile, as the probability of detection is low due to distraction (see the Cooperative Detection experiment in Chapter 5).

3. Objects can store optional knowledge of their detection performance with scale and rotation (as explained in section 4.4). This allows us to select a method which performs best at the current distance and orientation, following first detection of the object.

6.2.1.2 Pose Calculation

If the object is successfully detected a 2D location result is generated. This can take the form of correspondences between extracted image features and features in the Object Model, or a 2D image region in which the object has been detected.

The pose calculation step is subsequently performed to calculate the 3D location and orientation of the object with respect to the camera. The object pose can then be converted into a world coordinate system pose using the known camera location and orientation. This, in turn, allows the projection service to later convert the object world coordinate system pose to the pose of the object with respect to the projector using the known projector location and orientation, as shown in Figure 6.3.

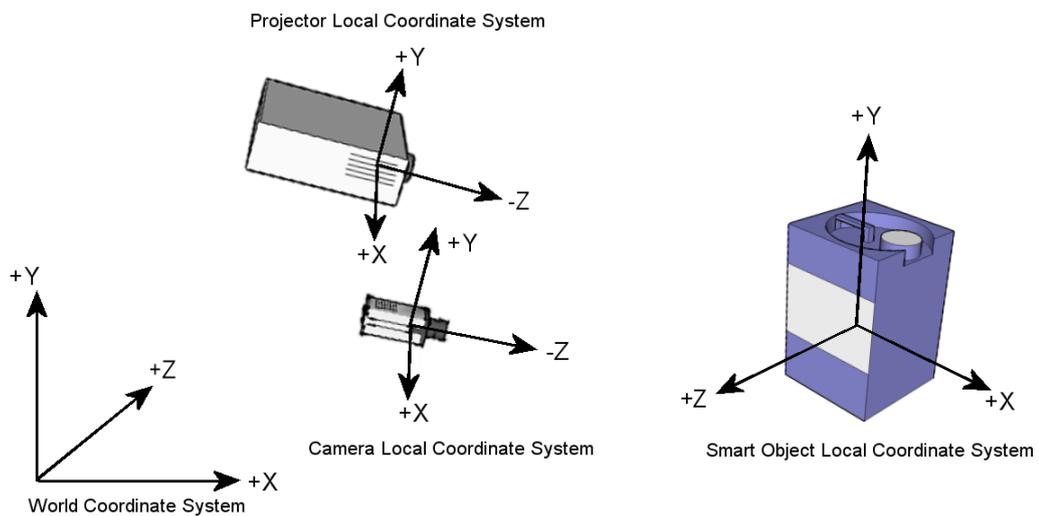


Figure 6.3 The 4 coordinate systems: Camera, Projector and smart object Local Coordinate Systems, and the arbitrary World Coordinate System

The object pose is calculated either directly from matched local feature correspondences, or by fitting the 3D model to edges detected in the 2D image region from the detection step. This step requires both the geometrical 3D model of an object and the camera intrinsic parameter matrix obtained from the camera calibration.

If the camera has a computer controllable powered zoom lens then the system has the ability to either detect many objects in a large area with the wide angle setting, or zoom in to increase the accuracy of detection for a single object or closely spaced group. The pose calculation algorithm requires the camera calibration to calculate the pose, however, this changes with the focal length when the lens is zoomed. Consequently, we can pre-calibrate the camera at multiple focal lengths, load these calibrations in the architecture, automatically fit a linear function and use this for interpolating the calibration matrix between the calibrated positions, based on the current zoom setting.

6.2.2 Projection

The design of the projection component enables undistorted images to be projected by correcting geometric distortion in the image resulting from projecting on a non-perpendicular or non-planar surface, as discussed in section 2.3.6. We compensate for this distortion by warping our projected image, as we know both the surface geometry of the object and the 3D pose of the surface with respect to the projector. We obtain the surface shape from the geometric 3D model embedded within the Object Model and the pose of the object from the detection process. The surface shape directly configures the geometric correction method used in projection [Bimber and Raskar 2005], as shown in Table 6.2.

Table 6.2 Geometric correction methods for projection based on object geometry [Bimber and Raskar 2005]

Object Geometries	Correction Method
Planar	Planar Homography
Rectilinear	
Cylindrical	Quadric Image Transfer
Spherical	
Irregular	Discretised Warping

The projection service manages projection requests from the smart object. Content is either downloaded from a location supplied by the object, or is loaded directly from the object for projection onto the object's surfaces.

Images of the object's surfaces from the detection service are used to correct the colour of the projection image to make it more visible on coloured and patterned object surfaces, as discussed in section 2.3.7.

The projection component requires the optical calibration of the projector to display projections accurately. This is obtained from the methods described in section 6.4. The projector calibration provides the system with three aspects of knowledge:

1. Horizontal and vertical Field Of View (FOV).
2. The optical Centre of Projection (COP).
3. Native projector resolution in pixels.

As discussed in section 2.3.8, in traditional (i.e. non-LASER) projectors an image appears acceptably in focus at the actual focal plane and for a limited distance in front and behind this distance. Outside this distance the projection appears blurred and difficult to read. Consequently, for environments with mobile objects (or objects with large depths) we must dynamically focus on objects to ensure a readable projection. If the projector hardware contains a computer controllable powered focus lens, the focus setting must be calibrated manually at multiple distances, by moving an orthogonal planar surface in steps from the minimum to maximum focus-distance and at each step focussing and recording the focus setting. The architecture can then load this calibration data and fit a non-linear function automatically, to be used for interpolating a dynamic focus based on the distance to a single object, or average of a group of objects.

Similarly, we would like maximum resolution projection (from the projector at maximum zoom) on each object, however this restricts both the size of the object and the number of objects that can be projected on, as the field of view of the projector is

limited. If the projector hardware has a computer controllable powered zoom lens then the system has the ability to trade-off the zoom level (hence projection resolution) with field of view dynamically, so as to encompass the most objects with the highest resolution. To use the zoom, the projector's optical parameters must be calibrated at multiple zoom settings and these calibrations loaded by the architecture, a linear function automatically fit and used for calculating the zoom setting based on the position of objects in the field of view of the projector requesting a projection.

To enable multiple projector-camera systems, we design our projection component using a similar display rights method to Ehnes et al. [Ehnes, Hirota et al. 2005] (see related work, section 2.3.8), which relies on the world model having an overview of all camera, projector and object locations and viewing volumes. Each projector determines the visual quality of its projection based on the distance and relative orientation of the object's surface to the projector using the metric introduced by Ehnes and Hirose [Ehnes and Hirose 2006]. Closer, more orthogonal projectors score higher, allowing a ranking to be performed and the best projector assigned display rights. This ranking is performed for each surface of an object (as defined in the 3D model) with an active projection, allowing simultaneous projection from multiple projectors onto multiple surfaces of the same 3D object. This method has the benefit of maximising the area of the object which can be projected on, while eliminating the possibility of mis-registration from image overlap causing blurred or unreadable projections.

6.2.3 Pan & Tilt Tracking Component

The pan and tilt component is solely responsible for controlling steerable hardware, allowing minimal code modification when changing or modifying hardware. We design the component to enable searching for objects using two search methods, and choose between them depending on our knowledge of the object.

6.2.3.1 Search and Tracking Methods

The first method is used to detect objects when the system has no prior knowledge of their location. In this case we use a creeping line search to rotate the pan and tilt hardware through its whole mechanical field of view. Figure 6.4 (left) shows this search method. The dotted line represents the pan and tilt mechanical Field Of View (FOV). The length and spacing of the movement track lines is based on both the pan and tilt FOV, and the angular FOV of the camera hardware being rotated, represented by the blue box. The track is designed so that some overlap of camera FOV occurs in the longest axis (here the horizontal pan axis), increasing the likelihood of detecting objects located midway between the track legs. However, the amount of overlap is a trade-off, as increasing overlap requires more tilt legs, hence, more time per search.

The second search method is used when the system has prior knowledge of an object's location in the world coordinate system. In this case the hardware is rotated to point at the object's last known location and an expanding-box search is used from this point, as shown in Figure 6.4 (right). In this case the movement track is solely based on the angular field of view of the hardware being rotated, represented by the blue box. The track is again designed so that some overlap in field of view occurs and for this pattern we overlap in both pan and tilt axes.

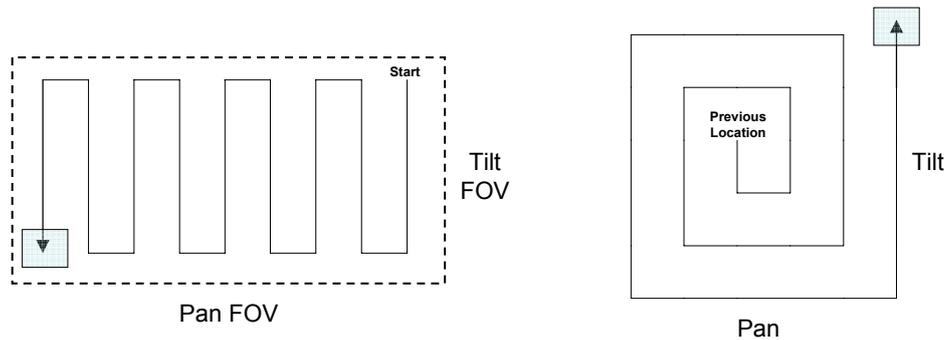


Figure 6.4 Search Methods: (left) Creeping Line Search over the whole Pan and Tilt Field Of View, (right) Expanding-Box search from previous object location

Once detected by the detection system, an object can be dynamically tracked. We assume knowledge of the location and orientation of the pan and tilt hardware in the world coordinate system and knowledge of the object location from the detection system. Consequently, object tracking is accomplished by calculating the relative angle of an object to the pan and tilt hardware origin in the pan and tilt axes, then rotating the hardware to minimise these angles.

6.2.3.2 A Concept of System Focus

If multiple smart objects exist in an environment it is possible to simultaneously detect and project on all objects inside a projector-camera system field of view. However, if only one projector-camera system exists and one or more objects are mobile this becomes a resource allocation problem. In this case the system must decide which objects to track.

To help resolve this problem we introduce a concept of system “focus”, which is similar to the concept of human attention. The system focus dynamically determines which objects to track and can be set either explicitly by the user, by an application aware of multiple objects, or automatically according to a set of rules.

An automatic system focus can be designed in many ways, depending on the amount of knowledge available about the user’s context. For example, we theorise that detection of object movement indicates deliberate interaction by a user. Consequently, we can easily design a system which always focuses on moving objects. However, such a naive function would encounter many difficult situations. For example, if a projection is already established on a group of static objects and another unrelated user moves through the camera field of view carrying another smart object, the projection would follow the unrelated user.

Another solution to the problem is to make use of multiple projector-camera systems in the environment, as discussed in section 6.2.2. The system architecture is designed to allow multiple projectors and cameras automatically detect and project on any objects within their field of view. This allows easy load-balancing across multiple projectors, so that an object can still receive a projection (even if it is not the best quality) if an object is inside the field of view of a projector and helps to overcome the problem of projection starvation, where an object never receives a projection as another mobile object is continually in focus.

6.2.3.3 Pan and Tilt Hardware Control

The pan and tilt tracking component abstracts from direct hardware control. This abstraction layer allows the pan and tilt hardware to be controlled in terms of angles in the world coordinate system, rather than in a hardware-dependant way. As we assume the pan and tilt unit has knowledge of its location and orientation in the world coordinate system, we can calculate the orientation of the pan and tilt hardware using a calibration between the hardware-dependant measurement units used to control the pan and tilt orientation to angles in the physical world.

We assume the pan and tilt hardware has an open-loop control system, so that there is no feedback of actual pan and tilt orientation. This requires a simulation of the motion of the hardware to determine the instantaneous location and orientation of the projector or camera attached to it. We simulate the motion of the hardware using a system of equations of motion, as proposed by Ehnes et al. [Ehnes, Hirota et al. 2004]. Two systems of control are possible, based on different hardware implementations:

1. Constant Speed Control, where the projector rotation speed is set (either manually, or by the system architecture) and constant for the whole of a movement, with exception of very short acceleration and deceleration periods.
2. Constant Time Control, where a projector always takes a constant time to perform a movement by automatically varying its rotation speed. This is subject to its maximum rotation speed, which limits the angle it can rotate in a constant time. Angles greater than this require increasing amounts of time.

For both control systems the maximum rotation speed will never be reached on very short movements, as the projector acceleration and deceleration phases overlap. However, in practice, the maximum pan or tilt movement speed we can actually use while searching or tracking with a camera is primarily determined by the camera exposure length. Fast movements with long exposures will cause blurring in the camera image unless the object is moving in the same direction at exactly the same angular rate. This blurring can cause loss of tracking and reduced detection performance.

Both control systems use the same equations of motion, however, we change the way the velocity variable is calculated for the different systems. To determine the current pan or tilt angle $S(t)$ we simulate movement individually for each axis using the following equation:

$$S(t) = S_{start} + V(t - t_{start}) \quad (6.1)$$

where $S(t)$ is the angle at time t , S_{start} is the initial angle and in the constant speed control system V is the signed rotation velocity (measured in degrees per second).

In the constant time control system the velocity is proportional to the distance remaining to the destination angle, limited by the maximum velocity. Consequently, V is calculated with the following equation:

$$V = (S_{start} - S_{end}) * \left(\frac{1}{T_{const}} \right) \quad (6.2)$$

where V is the signed rotation velocity, S_{start} is the initial angle, S_{end} is the destination angle, and T_{const} is the empirically measured constant time for the movement.

Ehnes et al. [Ehnes, Hirota et al. 2004] extended this simulation to include initial acceleration by adding an acceleration constant A , which must be empirically measured. This allows us to incorporate the time required to accelerate from the previous velocity to a new velocity.

$$S(t) = S_{start} + V_{start}(t - t_{start}) + \frac{A}{2}(t - t_{start})^2 \quad (6.3)$$

where V_{start} is the initial velocity set for the constant speed control, or calculated for the constant time control system.

While using an acceleration term may provide greater angle accuracy, it in turn relies on an accurate empirical measurement of the acceleration. This can be challenging without using actual sensors, as many pan and tilt units accelerate very rapidly.

In the constant speed control system we can calculate T_{motion} , the time required for the motion (and hence the time the projector will arrive at the destination by adding the current time) using the simple equation:

$$T_{motion} = \frac{(S_{start} - S_{end})}{V} \quad (6.4)$$

For the constant time simulation we simply assume the projector has arrived at its destination when t is greater than the constant time of motion.

6.2.4 Smart Objects

Smart objects describe themselves and their capabilities through knowledge embedded in the Object Model, contained in the object itself (as discussed in section 3.2). We assume that the smart objects automatically detect availability of a detection and projection service (simulating the availability of a discovery service such as UPnP [UPnP(TM)Forum 2003]), allowing them to cooperate with projector-camera systems and communicate this knowledge. Many everyday smart objects possess on-board sensing such as light, temperature and movement sensors associated with other purposes. The smart object cooperates with projector-camera systems to communicate this information, enabling serendipitous constraining of the detection process. Finally, the component assumes smart objects do not have access to external location systems precise enough to allow projector-camera systems to register a projected image with the object's surfaces.

In the Cooperative Augmentation Framework it is the smart object which controls the interaction with projector-camera systems. The smart object issues projection requests to control how an output (the projection) changes. Projection directly onto the object is the visual feedback to interaction.

The smart object is modelled as a state machine which responds to user interaction based on variations in input values, such as sensor inputs. This modelling is analogous to programming the smart object.

To “program” the smart object state machine we define a set of states which are constantly evaluated against the sensors. Each state defines one or more sensors together with operations on the raw sensor values (such as calculating the variance, or the mean), minimum and maximum boundary values required to enter the state, and the method of combining results from individual sensor operations, such as boolean AND or OR. Only when the operations performed on the raw sensor values evaluate to within the boundary values set and the required number of sensors (all for AND, or any one sensor for OR) are in range does the state change. A hierarchical tree of sensors, with each branch having its own method of sensor combination can be used to describe complex states.

Each Object Model contains three separate classes of states, for three associated object characteristics:

1. States to determine when object is moving or static (and so generate moving events)
2. States to determine when object changes appearance or geometrical shape (for example, for articulated objects to update the architecture with a new appearance and 3D model)
3. States to determine when projections are requested. One state must be defined per projection.

6.2.4.1 Input Modalities for State Change

Seven separate input modalities have been identified for use in modelling interaction with smart objects to enable the three separate classes of states to be programmed. We cluster the modalities into direct interaction (i.e. the user is physically touching the object) and indirect interaction:

Direct interaction that can be sensed by the object:

- 1) Manipulation of object location and orientation, as sensed by the camera.
- 2) Manipulation of object geometry (for example, opening a book).
- 3) Manipulation of physical interaction components on and sensed by object (for example, direct interaction with buttons or dials on its surfaces).
- 4) Other manipulation of object, sensed by object (for example, shaking detected by an embedded accelerometer sensor).
- 5) Interactive Projected User Interfaces (sensed via a camera).

Indirect interaction that can be sensed or used by the object:

- 6) Manipulation of physical environment remote to object (for example, switching the light on in the room).
- 7) Interaction with other smart objects in the environment.

All these input modalities are treated as sensors in the system architecture. Consequently, the sensor knowledge contained in the Object Model varies depending not just on the sensors physically embedded within the smart object, but also on what modalities we want to use for input in our smart object program.

Physical and projected user interface components, such as buttons, dials and sliders, are also modelled as sensors. Buttons are modelled with a range of 0-1, with 2 possible values of 0 (not pressed) and 1 (pressed). Dials and sliders are modelled as a continuous

value range between 0-100. Multiple axis controls are modelled as separate single axis controls. For example, a 2D joystick would be modelled as 2 sliders with separate physical axes.

Additionally, the current projection can be used as a sensor. As smart objects are modelled with a state machine, all possible projections must be known a-priori. Consequently, each individual projection can be numbered and used as an input sensor value when programming an object's states. For example, if projection image number 2 contains a button, we only want to detect whether the button is pressed when this projection is being displayed, as the other projections may not include buttons. While this requires us to model what is projecting in any individual state, it does not restrict what can be projected. For example, a video can be modelled as one projection in a single state, rather than a collection of individual numbered projection frames, each with its own state.

6.2.4.2 State Machine Processing

A generic pseudo code example of the processing required to determine when a state change occurs, and hence, when an event message is generated is shown below:

```

foreach defined state s
    foreach sensor n in state s
        Run Operation defined for sensor n on raw sensor data

        if operation result > min limit and result < max limit
            sensor n result = 1
        else
            sensor n result = 0
        endif
    endfor

    if result combination method == AND
        if all n results == 1
            Change to state s
            End
        endif

    else if result combination method == OR
        if any of n results == 1
            Change to state s
            End
        endif
    endif
endfor

```

The state processing example above does not take into account hierarchical trees of sensors; however, this is supported simply by recursion of the processing algorithm. In this case the nested processing return the result value of the processing rather than

change states directly, and the higher levels in turn use these results as sensor values in their processing.

In addition to the demonstration applications presented in Chapter 7, three simple examples of programming a smart object using the state-machine approach can be found in Appendix B.

6.2.5 Object Proxy

The object proxy translates messages between the native format of the smart object hardware and the architecture message format. By using a standard message format in the architecture different to the smart object format, this allows us to substitute different object hardware and only have to re-write the proxy.

Unknown messages received from the smart object by the Object Proxy are converted to the architecture message format and broadcast on the network. Similarly, unknown messages addressed to the smart object received from the architecture by the Proxy are converted to smart object format and forwarded to the smart object. This automatic conversion process allows the object to implement functionality outside of our defined system architecture, with the only requirement being that other communicating applications use our architecture message format for communication.

Similarly, it allows the smart object to decide where the abstraction of raw data values to events is performed, as events generated directly by the smart object are treated as unknown messages.

A smart object is modelled as a state machine which emits events and projection requests in response to changes in sensor data, location, orientation or stored knowledge. The object proxy converts any sensor data streamed from the smart object directly to event messages if the smart object is unable to perform this operation due to lack of processing power.

The object proxy application itself is generic, with conversion processing configured dynamically for each smart object from the Object Model. As discussed in section 6.2.4, the Object Model stores the description of sensors in the object and the conversion method to events. This conversion method is described in terms of operations (such as calculate the variance, or the mean), minimum and maximum threshold values and the method of combining results from the operations, such as boolean AND or OR. This conversion of data to events can be applied to any type of sensor data. The processing required to determine when a state change occurs, and hence, when an event message is generated is identical that described in 6.2.4.

6.2.6 Database Server

The smart objects initially register with the Database Server and provide a Unique ID (UID). This creates the virtual object cache in the server which mirrors state changes of the physical object. Changes to the database state are either broadcast on the network to all applications or sent to the smart object, depending on the source and contents of the message. Other applications can now query the object state directly from the server.

Maintaining an Object Model database has four additional benefits:

1. It synchronises access to smart object knowledge by different services and applications

2. It coordinates distributed projector and camera hardware, allowing them to each offer separate projection and detection services, while maintaining a unified world model of the environment
3. It allows services to be stopped and subsequently re-initialised with current world state by querying the database.
4. It minimises network traffic to and from the physical smart object

Object proxies are responsible for keeping the Object Model updated when the 3D model or appearance of the object changes. The database server additionally caches any display request messages or requests for interactive projected interfaces sent by the Object. This allows architecture applications to re-initialise and regain full knowledge of the object state by querying the database server.

6.2.6.1 Projector and Camera Database

Similarly, projectors and cameras register with the Database Server, so it contains knowledge of all hardware in the “environment”. The concept of an environment is introduced here as many projector-camera systems may be deployed throughout a building. The concept itself is similar to the concept of sub-nets in IP address based networking, where small areas of a larger network are effectively segregated. Similarly, our concept has a single database server responsible for a small physical real-world area, or “environment” and a small group of projectors and cameras. This enables our approach to easily scale, allowing simultaneous use of many objects in a whole building. Similarly, if mobile, handheld or wearable projector-camera systems enter the environment this approach also maintains the benefits of hardware flexibility and performance.

The server maintains a database of all projectors and cameras present in the environment. When a projection is requested by an object, this database is used to identify the physical detection and projection hardware currently detecting and projecting onto each object surface. We form loosely-coupled projector-camera system pairs from all the available hardware in the environment for each projection requested.

We aim to dynamically modify the pairings so that the best projector and camera is always used to increase robustness. Detection and projection can continue as long as one camera and one projector can see the object, even if other hardware is occluded.

For example, we can imagine an environment with two projectors and two cameras which form two separate physical projector-camera systems, as each camera is attached to the top of its respective projector. Both projector-camera systems are oriented the same way, but with a small horizontal separation between them. Consequently, there is a large horizontal overlap of the viewing frustums between the projector-camera systems. Inside the overlapped region we can use any (or all) of the cameras and projectors for the detection and projection tasks. If a person walks in front of the object, the actual hardware used will vary depending on which projector and camera is occluded as the person walks past.

6.2.6.2 Dynamic Pairing

All cameras detecting a smart object return a location and orientation update to the database server. If more than one camera detects the object the location and orientation hypotheses are integrated to a single predicted location via a Particle filter [Pupilli and Calway 2006]. The predicted location is finally returned to the smart object in a location and orientation update message.

The Database Server stores an array of the UID of cameras currently tracking the object in the respective Object Model. We assume that the object is not being tracked if its location and orientation has not been updated by a detection service for a user defined time (e.g. 1 second).

When an object requests a projection the request is sent to all projectors. Each projector returns a visibility score based on the relative location and orientation of the object's projection location with respect to the projector. The database server manages the projectors to ensure only one projector is active for each requested projection.

The returned values are ranked, and the highest scoring projector has the object set in-focus (the concept of system focus is discussed in section 6.2.3). Lower scoring projectors have the object set to out-of-focus. The database server re-evaluates all visibility scores at a user defined interval (e.g. 1 second) and changes the projector's focus as required. Two strategies are possible to prevent rapid flicking between different projectors with identical scores:

1. A user-configurable delay before change.
2. A re-focus threshold, so the projector chosen remains active until its score drops below a pre-defined minimum, irrespective of the ranking.

6.2.7 Knowledge Updating

The design of the Cooperative Augmentation architecture provides a mechanism for re-embedding knowledge into smart objects. The knowledge is extracted by the knowledge updating component and re-embedded into a smart object via the Database Server and Object Proxy. This design has three main benefits. Firstly, it allows the objects to enter the environment with flexible amounts of appearance knowledge. Secondly, it allows projector-camera systems to use multiple-cues in detection to increase detection performance (as shown in section 5.2). Thirdly, new detection methods can be implemented in the projector-camera system for improved detection performance, without explicit modification to the Smart Object. Instead, the knowledge updating component automates appearance extraction and re-embedding in the object.

To achieve this automation, the component uses images of the object's surfaces extracted from the camera image by using the known 3D model and pose of the object when detected. The new detection method can be used in the knowledge update process to extract new appearance knowledge from each surface whenever the surface is at the optimum distance and orientation relative to the camera for training the detection methods (see section 4.4.2). This knowledge is re-embedded in the Object Model for more flexible and more robust detection.

Newly extracted knowledge is merged with existing knowledge in the Object Model. If the knowledge is not contained in the object we simply add the knowledge to the Object Model. However, if the object already contains knowledge extracted with the same detection method we have three options:

1. Keep the existing knowledge
2. Replace the existing Knowledge with the new knowledge
3. Merge the existing Knowledge with the new knowledge

To determine which option to choose we can run the detection method twice per camera frame (once for the old knowledge, once for the new knowledge). At the end of a pre-determined period we decide which knowledge performs best and either keep the old knowledge or replace it.

Knowledge can also be merged, however this has dangers as the resultant knowledge may not accurately reflect the object's appearance if there was a misdetection, or the old and new knowledge were extracted under different conditions. For example, a blue chemical container would appear to have one hue under incandescent illumination (3200K temperature), and a different hue in daylight (6500K temperature) unless the camera is accurately colour calibrated in each case.

Extracted appearance knowledge is stored in the Object Model on a per-surface basis, relating to surfaces defined in the 3D model.

6.3 Implementation

This section describes a demonstration implementation of the Cooperative Augmentation conceptual framework using the system architecture designed above.

As the system will be deployed in a lab environment, we aim for a level of performance to allow the implementation to function as an interactive demonstrator:

1. Based dimensions of the lab environment shown in Figure 6.5, the implementation should aim to detect, track and project onto objects up to a minimum of 6m from the projector-camera system.
2. The implementation should aim to achieve a maximum 20mm median projection location error and a maximum 1° median orientation error when projecting on a planar object at 3m distance from the camera.

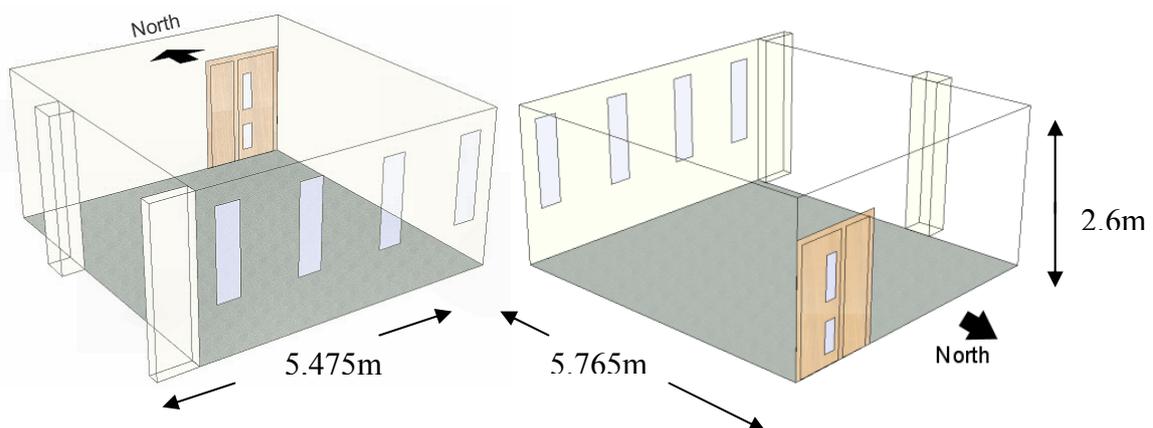


Figure 6.5 Two views of a Lab Environment: (left) South-West Elevation, (right) North-East Elevation

We implement smart objects using Smart-Its particle hardware [Decker, Krohn et al. 2005], with attached “ssimp” sensor board part (see section 2.2.1).

A steerable projector-camera system (shown in Figure 6.6) similar to the ones described by Ehnes et al. [Ehnes, Hirota et al. 2004] and Butz et al. [Butz, Schneider et al. 2004] is implemented from a moving-head display light pan and tilt unit, a lightweight DLP projector and a firewire camera. As our scenarios incorporate mobile

objects, there is great benefit in using a moving head steerable projector system, as it allows a single system to be mounted in a location with the best view of the whole environment (the centre of the ceiling) and maximise the volume in which objects can be tracked.

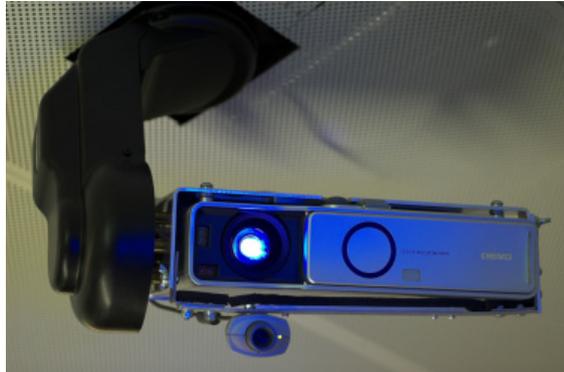


Figure 6.6 The moving-head steerable projector-camera system

The pan and tilt unit is controlled by the computer using the DMX512 serial protocol and can move through an addressable 540° pan angle and 265° in tilt at up to 180° per second. The unit can rotate the projector-camera system hardware in two real-world dimensions – horizontal (pan) and vertical (tilt), about the approximate centre of projection (COP) of the projector.

The projector is a 2800 Lumen Casio DLP projector with large 2x zoom capability, allowing us to maintain relatively high resolution projection on objects at large distances. The firewire camera is a Pixelink 1280x1024 resolution colour CMOS camera capable of 27 frames per second (fps) at full resolution and up to 104fps with a 640x480 region of interest enabled. The camera is used with a 12mm C-Mount lens with $40 \times 30^\circ$ field of view (FOV).

More information on the steerable projector-camera system is found in Appendix A.

The implementation of the architecture components are described in detail in sections 6.3.4 to 6.3.3. Each component was implemented in C++ as a stand-alone application on the network and use a common messaging format for communication, as described in section 6.3.6 below.

6.3.1 Detection and Tracking

The detection and tracking service is started with an XML configuration file providing four further aspects of information:

1. Name or unique identity (UID) of camera hardware (to address it on the network).
2. Location in world coordinate system (WCS) if static.
3. Orientation in WCS if static.
4. Optical parameters or camera calibration file.

If the camera system is a steerable system, the Pan and Tilt tracking component will automatically provide the system with the current hardware location and orientation.

The detection system attempts to detect all smart objects in the environment. As this would have a significant processing cost with large numbers of objects, the detection system first checks to ensure an object is visible. If the object has not been detected yet, or the Database Server indicates the object is not currently tracked, then the system automatically attempts to detect it.

If the Database Server stores a location and orientation for the object and indicates it is currently tracked, the detection system calculates its viewing frustum in the world coordinate system based on its current location and orientation. The location of the object is checked against the viewing frustum to see if the object is inside. Only when the object is potentially visible does the system attempt to detect it.

One detection algorithm per cue (colour, texture, shape, features) is implemented in the detection system, as described in section 4.2. Algorithms are implemented using OpenCV [IntelOpenCV 2007] to benefit from its built-in algorithm optimisations for Intel x86 CPUs. Additionally, as many image-processing tasks are inherently parallelisable, we also implement components of the detection system on the graphics card (GPU). The user can switch between CPU-only and combined CPU-GPU implementations by re-compiling the detection system application with different pre-processor parameters.

Common tasks such as gaussian blur, absolute difference, background subtraction and histogram creation are all implemented on the GPU using NVIDIA's "Compute Unified Device Architecture" (CUDA) language. This allows algorithms to be written in a C based language, yet benefit from a typical 10-15% speedup when executed on the GPU. For example, a CPU optimised version of the SIFT local feature algorithm takes approximately 333ms to detect a single object in a 640x480 pixel image [Lowe 2004], whereas a GPU version only takes 100ms [Sinha, Frahm et al. 2006]. As CPU-GPU data transfers are typically a bottleneck, each camera image is uploaded to the GPU following capture and as much processing as possible is performed on the GPU before returning any result data to the CPU.

6.3.1.1 Figure-Ground Segmentation for Object Motion Detection

The detection method selection is performed for each object based on what appearance knowledge the object holds, the object context (which we can obtain from a background model) and whether the object contains a movement sensor.

To generate a motion mask and basic figure-ground segmentation of the target object the current system architecture implementation uses a simple absolute difference operation between the previous camera frame and the current frame whenever the object's sensors detect motion. This method is fast to compute and works well when the camera is static or only moving slowly, increasing the probability of correct detection (as demonstrated in section 5.2).

Another method for mobile or fast moving cameras that are constrained to move in a known path (e.g. a steerable projector) is to maintain a background model. To create the background model for steerable projectors we implemented a separate application to rotate the steerable projector to cover their whole Field Of View (FOV) while capturing camera images. For moving head steerable projectors the FOV is typically a full hemisphere. The application also requires knowledge of the camera Field Of View (FOV) to calculate how many photos are required to achieve full coverage. Each photo overlaps with its neighbours by 1/3, allowing automatic stitching and blending of the images into a single rectangular texture. An example of a 360x90° FOV background

model stitched from 16x4 overlapping images then re-projected into a hemisphere can be seen in Figure 6.7. As the capture and stitching process typically takes several minutes, we assume it is performed off-line before object detection (for example, the environment could be scanned automatically overnight).



Figure 6.7 View down into a background model re-projected into a hemisphere, captured by rotating a moving head steerable projector through a 360x90° FOV.

During on-line detection the known pan and tilt unit orientation and camera FOV allows us to calculate the area of the background model currently viewed by the steerable projector for visual differencing, for use as context information or for update. To reduce both memory requirement and the chance of any small inaccuracies in the pan and tilt unit orientation affecting the visual differencing we can reduce the resolution of the stored background model to 50% or 25% of the original size.

To increase robustness we could also use visual differences generated between the camera image and a Gaussian-mixture model of the background [Stauffer and Grimson 1999] to provide a basic figure-ground segmentation to the detection algorithms. The Gaussian-mixture model is calculated independently for each pixel in the stitched rectangular texture background model and selectively updated based on the area of the background model currently viewed by the steerable projector.

For mobile, handheld or wearable cameras a background model can be built using structure-from-motion SLAM techniques [Davison and Murray 2002; Davison 2003; Chekhlov, Gee et al. 2007; Klein and Murray 2007].

6.3.1.2 Detection Method Selection

If the object only has knowledge of a single appearance cue we automatically choose the respective detection method. However, as explained in section 6.2.1, when an object has appearance knowledge for multiple cues we can choose whether to perform a single detection method which prioritises an aspect of detection (speed, accuracy or robustness), or perform multiple methods and combine the results. The multi-cue detection results in Chapter 5 indicated only the best two cues contributed significantly to successful object detection, hence when detection metrics are available for multiple methods we can choose the top two.

We select multiple algorithms in two cases:

1. For first detection of an object, to maximise the chance of detection. Here we process the top 2 algorithms sequentially, in order of ranking, and use the detection results of the first algorithm as a figure-ground segmentation for the second algorithm.
2. When the average runtimes of the top two ranked algorithms are less than 10% different. In this case we execute the algorithms in parallel, either in multiple threads on the CPU (taking advantage of multi-core CPUs) or with one on the CPU and one on the GPU. This approach retains the benefit of a multi-cue system with minimal impact on performance. The cue results are combined at the end of the processing step using a binary OR operation on the detection areas in the camera image for colour, texture and shape, or by masking the detected local features before the pose calculation step.

The detection process aims to maintain between 5 and 60 frames per second for interactive system performance. As we process every camera frame, the exact frames per second achieved will vary, depending on the user-configurable camera frame rate, the execution time of the detection algorithms and the number of objects to be detected.

To reduce processing time for multiple objects we separate detection algorithms into operations which are common for multiple objects (such as the initial difference-of-Gaussians pyramid creation for the SIFT local feature algorithm) and object specific operations (such as matching the object appearance to the processed camera frame). While requiring sufficient memory to store all intermediate results, this allows the majority of image processing to be performed only once per camera frame, irrespective of the number of objects using the detection method.

6.3.1.3 Pose Calculation

We perform a two-step pose calculation. Firstly, the object pose is calculated either directly from matched local feature correspondences or by fitting the 3D model to edges detected in the 2D image region from the detection step. The local feature pose calculation algorithm calculates a homography matrix for a planar surface and for non-planar objects the Direct Linear Transform (DLT) algorithm is used. The calculated matrix is decomposed to extract the location and orientation of the object with respect to the camera. The 3D model fitting initially detects edges in the camera image around the location of detection using the Canny algorithm [Canny 1986]. A RAPiD-like algorithm [Harris 1993; Armstrong and Zisserman 1995; Gee and Mayol-Cuevas 2006] is then used to project lines from the model into the camera image at multiple orientations and scales, centred around the detection location and test to see which pose has the best fit to the detected edges.

If the smart object contains 3D accelerometer sensors these can additionally be used in the pose computation step when performing the 3D model fitting. In this case the sensed gravity vector is directly used to constrain the orientation, and hence the number of 3D model poses that must be tested to match the edges detected in the image, as explained in section 5.3.4.

These detection and pose calculation steps requires the camera calibration and use the RANSAC algorithm [Fischler and Bolles 1981] for both robust model parameter estimation and for eliminating incorrectly matched correspondences. This initial calculation is followed by the second step where we perform an iterative pose refinement to increase accuracy, using the Gauss-Newton algorithm. An example of the

registration of an object with its 3D model using the result of the pose calculation is seen for the book object in Figure 6.8 (left).

6.3.1.4 Tracking

There are three main sources of latency in the implementation – camera frame acquisition, image processing for object detection and projection. For a camera running at 30Hz the frame acquisition takes up to 33.3ms, while for a 60Hz projector a frame is projected every 16.7ms. Hence, maximum latency before image processing is 50ms (assuming unsynchronised cameras and projectors). It was reported by Brooks [Brooks 1999] that users of projector based interactive systems routinely accept total system latencies of 150ms, so we should aim to perform the detection step below 100ms. In contrast, the CPU runtime of the natural appearance algorithms we use in the architecture is between 288ms and 852ms per 1280x1024 pixel frame (as shown in section 5.2.4).

For the demonstration applications presented in Chapter 7, these natural appearance detection algorithms do not perform in real-time, hence, our implementation uses a tracking system to speed up the detection process after first detection. A recursive tracking approach constrains the area of each camera frame used to detect the object based on its previous location, increasing the frame rate to interactive levels (>5fps). An example demonstrating tracking of multiple detected objects is seen in Figure 6.8.

To increase tracking robustness we change the tracking method used to use one of two methods, depending on the appearance description in the object. The initial detection is performed by the natural appearance cues, then either the 3D model is used for RAPID-like tracking [Harris 1993; Armstrong and Zisserman 1995; Gee and Mayol-Cuevas 2006] and pose calculation, or corner features are extracted on the object's surfaces using FASTcorners [Rosten and Drummond 2005; Rosten and Drummond 2006] and a CONDENSATION particle filter [Isard and Blake 1998; Pupilli and Calway 2006] used for recursive tracking, with Normalised Cross Correlation (NCC) of image patches around detected corners on the object. NCC is not scale or rotation-invariant, hence this tracker fails and must be re-initialised whenever the object is scaled or rotated (both in the camera plane and in 3D) significantly from the original pose where features were extracted.

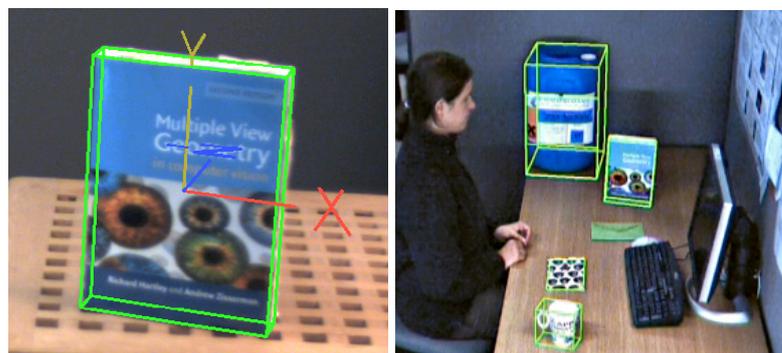


Figure 6.8 (left) Image of detected of Book object with overlaid 3D model and object coordinate system axes, (right) Four detected objects tracked simultaneously - the Chemical Container, the Book, the Card and the Cup (green lines denote the maximum extent of the 3D model).

Objects with surface texture (texture or local features appearance descriptions) use NCC tracking, while the RAPID-like tracker is used for objects with well defined edges (for the colour or shape appearance descriptions). If appearance descriptions allowing use of both the NCC tracker and RAPID tracker are present, NCC takes precedence.

A complete detection step is performed again whenever tracking is lost. Loss of tracking is determined by a tracking quality figure, dependant either on the number of line-segments matched for the RAPID-like tracker, or the number of NCC image patch features matched (hence, on the similarity of the extracted features to features seen in the current image) in the particle filter.

6.3.1.5 Vision-Based Interaction Detection

Messages sent from the smart object requesting interaction components (such as buttons and sliders) contain the location for virtual interactive areas on the object's 3D model and values to return when activated. For each camera frame, the system calculates whether any of the active areas on the set of visible objects are themselves visible. If an interactive area is visible it is extracted from the camera image by first projecting the 3D corner locations into the image using the known object location, orientation and camera calibration. The quadrilateral area in the camera image is unwarped to a rectangular image of user-specified size, typically 100x100 pixels.

A finger tracking process is run on the rectangular image which performs a normalised cross-correlation of a semi-circular fingertip template image with the image area. The maxima locations are found in the result image and if greater than a user-defined threshold (typically 0.6 or 0.7) the template is considered a match.

This interactive area is monitored whenever the object is visible and a motion history silhouette image created using accumulation of background subtracted images of recent frames. This silhouette image enables calculation of the gradient orientation, allowing us to check for the "lightening strike" touch of button activation whenever a fingertip template is matched inside the interactive area, as explained in section 2.3.5.

For an interactive button, the first fingertip match inside the area with the correct motion profile is considered as button activation and the process returns the value requested. We assume the finger is still inside the area while the cross-correlation results return a match, so prevent repeat activation of the button if the user's finger is hovering. An interactive slider returns a value depending on the location of the maxima along the largest axis of the slider area. This axis is interpolated linearly to return a value between the minimum and maximum values specified by the smart object.

6.3.2 Projection

In a similar way to the detection service, the projection service is started with an XML configuration file providing four further aspects of information:

1. Name or unique identity (UID) of projector hardware (to address it on the network).
2. Location in world coordinate system (WCS) if static.
3. Orientation in WCS if static.
4. Optical parameters of the projector.

If the projector system is a steerable system, the Pan and Tilt tracking component will automatically provide the system with the current hardware location and orientation.

For the implementation of the projection service we use OpenSG API scenegraph [OpenSG 2007], providing an object-oriented approach to building a world model. The FreeGLUT and OpenGL APIs are used for windowing and graphics rendering respectively. The scenegraph models the real world, with a virtual camera in the scenegraph placed at the location of the real-world projector and modelled with the same optical characteristics of the real-world projector from the configuration file. Any object registering for a projection has their 3D model added directly to the scenegraph hierarchy. The object's surfaces are set to black (invisible) and the model placed at the location of the real-world object in the 3D scene when its pose is calculated.

When the smart object requests a projection its message includes both the content to project (which can be images, text or video or a URL where content can be found) and the location to project it. We use the OpenCV [IntelOpenCV 2007] API to load the content to project, supporting the following image file formats: BMP, DIB, JPEG, JPG, JPE, PNG, PBM, PGM, PPM, SR, RAS, TIFF, TIF and AVI video files.

The system can project onto any object area visible to the projector. The location description refers to the projection location abstractly or specifically. Abstract locations refer to faces of the object's 3D model by their name. For example, a projection can be requested on the top or front face if these are declared in the 3D model. A smaller or more specific area can also be specified as coordinates in the 3D model coordinate system, allowing exact placement and sizing of the projection on an object. In this case, for the duration of the projection a new planar polygon is created with the specified coordinates and added to the 3D model.

To create a projection we texture map the content of the projection either directly onto one of the faces of the object in the 3D scene, or onto the newly created polygon. This texture image is automatically updated in the case of video content to achieve the correct frames-per-second playback speed.

6.3.2.1 Geometric, Photometric and Colorimetric Correction

Geometric correction is automatic for planar surfaces due to the use of projective texture mapping, where the virtual scene arrangement mirrors the real-world object and projector relationships. Similarly, we use the calibrated real-world projector intrinsic parameters for the optical characteristics of the virtual camera. With curved geometries we can use vertex and pixel shaders in NVIDIA's Cg language for Quadric Image Transfer, as described by Bimber and Raskar in [Bimber and Raskar 2005].

We use a colour correction algorithm to change the texture image, correcting for non-uniform and non-white surface colours. An image of the object without projection can also be calculated as part of this process and used for object detection.

The real-time algorithm by Fujii et al. [Fujii, Grossberg et al. 2005] is used for this step, however, the correction has a one camera frame delay cost, to allow the latest camera image to be used in the algorithm. Additionally, the algorithm requires an initial one-time projection of four colour calibration image frames (red, green, blue and grey) to recover the reflectivity response of the surface. The adaptation algorithm allows colour correction for each subsequent frame to be projected without projecting the calibration frames again. However, the algorithm cannot completely correct very saturated surfaces, as the dynamic range of typical projectors is not sufficient to invert

the natural surface colour. This effect can be seen in the projection on the red top surface of the box object in Figure 6.9 (left).



Figure 6.9 (left) 3 surface projection on box object, (right) Sensed temperature projected on the non-planar smart mug surface - the blue wire at the right is the antenna of the Smart-Its device

6.3.2.2 Projector Powered Focus Control

When a projector-camera system includes a projector with a powered focus the architecture can control this to dynamically focus the projector on the objects currently “in focus” in the architecture (“focus” as described in 6.2.3). The architecture requires calibrated focus data (calibrated as described in section 6.2.2) to be able to calculate the correct focus setting for an object distance. The focus setting-distance calibration data is loaded from a configuration file and a non-linear function fit to the data.

Whenever the object “in focus” changes, the architecture outputs focus-near or focus-far commands, depending on whether the object is closer or more distant than the current focus distance.

If the projector includes a serial port for computer control, the Focus Control can output projector-specific control strings. These control codes are loaded from a configuration file and consist of a series of bytes for the focus-near and focus-far functions. These codes are output on COM2 serial port when required.

If the projector does not have a serial port, LIRC [Bartelmus 2007] can be used to control the projector functions by generating Infrared remote control codes to a connected IR transmitter. Here again, the codes are loaded from configuration file and consist of a series of bytes for the focus-near and focus-far control functions. The codes are sent to a network port where the LIRC API listens and translates the codes to a series of pulses to be sent to an attached IR transmitter which converts them to IR light pulses. IR transmitters can either be purchased or easily be constructed.

As there is no focus setting feedback from the IR remote control method the architecture outputs a continuous stream of focus-near commands for several seconds when the software is first started, to guarantee the focus has moved to the closest focus setting. The calibration data can then be used to change focus from this known initial focus setting. In this case, focus commands are continuously sent with a pause of 0.5s in-between to give the hardware time to react, until the correct focus distance is reached.

6.3.2.3 Projector Powered Zoom Control

In a similar way to the dynamic projector focus control, if a projector-camera system includes a projector with a powered zoom we can dynamically zoom-in the lens to achieve a high resolution projection on distant objects or zoom-out to project on large close objects. With dynamic zoom we can trade-off projection resolution for Field Of View (FOV).

However, in addition to changing the projector intrinsic parameters, changing the zoom changes the projector-camera transformation (extrinsic) matrix. Hence, projector-camera calibration must be performed for every zoom step (as described in 6.4) and all the intrinsic and extrinsic matrices loaded to be used at the respective zoom setting. We additionally require calibrated zoom data (as described in section 6.2.2) which describes the projector image FOV and corresponding zoom settings.

An identical control method to that used for Projector Powered Focus Control is used to change the zoom – either by output projector-specific control strings to the serial port, or by sending Infrared remote control codes to the LIRC network port.

6.3.3 Pan and Tilt Control

The pan and tilt component is used whenever either (or both) projector or camera hardware is attached to a pan and tilt unit. In a similar way to the detection and projection services, the pan and tilt component is started with an XML configuration file providing four further aspects of information:

1. Location of the Centre of Rotation (COR) of the Pan and Tilt platform in the world coordinate system.
2. Unique ID (UID) of Projection hardware and Offset transformation of the Projector from COR (if applicable).
3. Unique ID (UID) of Camera hardware and Offset transformation of the Camera from COR (if applicable).
4. Calibration of the Pan and Tilt unit, to allow control in angles rather than hardware units.

The pan and tilt component automatically searches for and tracks mobile smart objects in the environment. When only a single object exists in the environment and its location is unknown, the pan and tilt component will automatically search for it using the creeping line algorithm. When multiple unknown objects exist the component will keep searching until it either finds all the objects, or an object requesting a projection (in which case it will track this object). If an object being tracked stops requesting a projection, the component will return to searching for objects if there are objects with unknown locations in the environment and no other objects requesting projection. If another object with previously known location requests a projection it will return to this location, perform an expanding-box search to locate the object and track this object.

If multiple objects request projection the tracking is performed using a 3D clustering with the K-means algorithm on location information of the objects. This is followed by selection of the target objects to be tracked based on recent object interaction history (we assume interaction takes the form of change in location, orientation or manipulation of the object). We track whichever object in the cluster was last interacted with. This aims to track the largest group of objects where interaction has recently occurred.

6.3.3.1 Pan and Tilt Hardware Control

Pan and Tilt hardware control is achieved by outputting value strings via the serial port to a serial-to-DMX512 converter. The strings output channel number (relating to control channels in the pan and tilt hardware), followed by 8-bit channel value (0-255). The Pan and Tilt unit calibration loaded from the configuration file specifies how these channels and values map to pan and tilt angles in the real world, allowing the architecture to control the unit without knowledge of the hardware.

The pan and tilt component automatically updates the detection and projection services with the location of the camera or projector respectively, based on knowing which hardware is attached to the unit and the current location and orientation of the pan and tilt unit derived from the motion simulation discussed in section 6.2.3.

If the projector or camera hardware does not have a static pan and tilt unit (such as mobile or handheld projector-camera system) another application is required to calculate and update the detection and projection services with the hardware location and orientation.

6.3.4 Object Proxy

Messages from the Smart-Its device are broadcast on an RF channel to Smart-Its bridge hardware, which re-transmits the packets on the local IP subnet as UDP messages containing the data in the Smart-Its AwareCon protocol [Decker, Krohn et al. 2005].

To replicate the existence of a discovery service a listener application was implemented to automatically start and stop smart object Proxy applications. The listener application listens for network traffic generated by Smart-Its devices. If a new Unique ID (UID) of a Smart-Its device appears on the network an Object Proxy is spawned to automatically handle communications with the object.

The listener application maintains an internal database of devices active on the network. If a device has not been active for a pre-determined period of time it sends a query message to the device. If no response is received following a short time-out the listener sends a stop message to the respective Object Proxy.

The Object Proxy is implemented as an application which converts messages between the AwareCon protocol and the EiToolkit [Holleis 2005] protocol used for inter-process communication in our architecture. There is no direct routing of messages from the proxy to services due to the absence of a discovery service; instead all messages are simply broadcast as UDP messages on the network.

6.3.4.1 State Machine and Sensor Data Abstraction to Events

The proxy implementation is generic. Each proxy spawned by the listener application is configured by an XML Object Model file either loaded directly from the Smart-Its device, or from an accessible network resource. The XML file contains specifications of sensor configurations for each state. A sensor configuration is composed of one or more sensors and associated sensor ranges which the sensor must be between to enter the state. Three separate classes of states are defined in the XML file, for the three associated object characteristics of movement, geometry or appearance change, and for projected content modification, as discussed in 6.2.2.

Abstraction of sensor data to events is performed either in the smart object or in the proxy itself, depending on the performance of the smart object hardware. Conversion of streamed sensor data in the proxy is implemented using operations performed by the Common Sense Toolkit (CSTK) [VanLaerhoven 2006]. CSTK provides type independent operations with the support for a variety of mathematical methods, such as min, max, mean, median, variance, running variance, standard deviation. The XML Object model file specifies the buffer length of raw data samples on which the operation should be performed, together the name of the sensor, operation and its associated numerical minimum and maximum in ascii text.

Conversion of sensor data and state evaluation is performed by the proxy at between 30 and 100 iterations per second (this is user configurable). When the proxy is first started, the initial states will be broadcast following filling of the CSTK buffers with data to avoid incorrect results from initial operations. The proxy then broadcasts an event message for any subsequent change in state.

On receipt of messages addressed to the smart object, the object proxy caches the messages if raw sensor data is being streamed to the Object Proxy. As all important operations occur in the proxy, periodic updating of the smart object is sufficient to allow state synchronisation while maintaining bandwidth for sensor data streaming.

6.3.5 Database Server

When smart objects register a cache of their Object Model is stored in a vector by the database server. A registering smart object must provide the following minimum information from the Object Model:

1. A name or unique identity (UID). This is usually the IP or MAC address.
2. A 3D model of the current object shape in VRML format.
3. An appearance model containing a minimum of one appearance description.

The information is either included in the message as XML, or optionally the messages contains the URL of an XML file containing this information on an accessible network resource. Similarly, when a detection or projection service registers, this information is again stored in a vector by the database server for use in dynamic projector and camera pairing. The following configuration information must be provided:

1. A name or unique identity (UID) of the hardware (to address it on the network).
2. The hardware type – either projector or camera.

6.3.6 Networking Protocol Implementation

The EiToolkit [Holleis 2005] (developed in conjunction with HCILab, Ludwig-Maximilians-University Munich) was used to provide a common networking component for all distributed applications.

EiToolkit provides support for event-based programming, with mechanisms for receiving messages, message decoding and function callback execution. Interface querying is also supported, allowing a proxy or application to interrogate the

capabilities of others. EiToolkit also supports automatic translation between Smart-Its [Decker, Krohn et al. 2005] sensor node AwareCon protocol and the EiToolkit native format via a dedicated proxy.

The message format of EiToolkit is plain text, in the form: <senderID>:<msgType>:<message>:<destinationID>.

The senderID and destinationID are unique identifiers, and can be the IP of the hardware running the proxy or application, or a text-based user-defined unique identifier (e.g. "Projector1"). Message broadcast is possible by specifying the destinationID as '*'. The msgType defines the command or interface callback to execute on the receiving system. The <message> is the payload or data to be used in the executed callback.

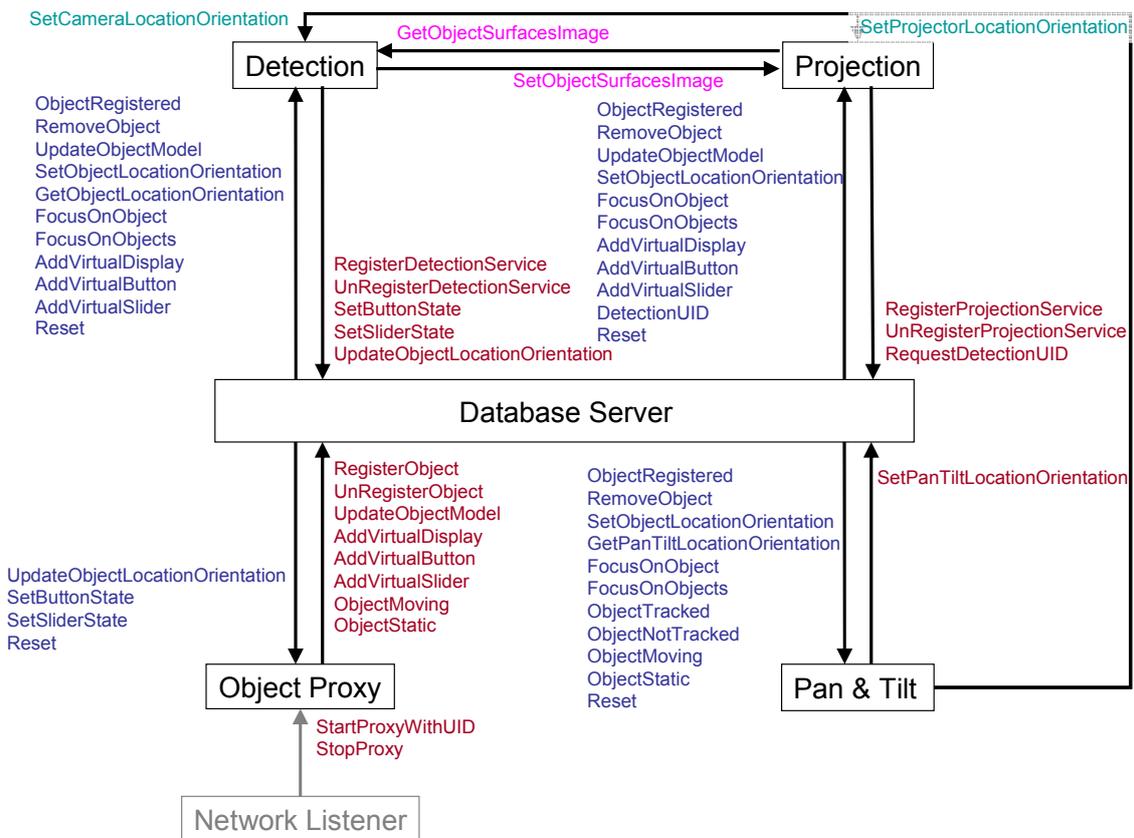


Figure 6.10 Architecture Message Protocol and Routing.

We designed a communication protocol based on the conceptual framework in Chapter 3. Figure 6.10 illustrates the architecture message protocol and typical routing. Red messages are from other components to the Database Server, Blue messages are from the Database Server to other components, green messages are Pan and Tilt position updates and pink messages are for colour correction. These messages used in the system architecture are broadly separated into four groups:

1. Messages between the smart object Proxy and Database Server.
2. Messages from Database Server, common to multiple applications.
3. Messages for between Database Server, Detection Application and Projection Application for colour correction.

4. Messages from Pan and Tilt to either or both Detection and Projection Applications.

These four groups are explained in more detail below.

1. Messages between the smart object Proxy and Database Server are listed below, together with the message contents.

RegisterObject	Registration request when proxy starts. Message contains object ID.
UnRegisterObject	UnRegister request sent when proxy stops. Message contains object ID
UpdateObjectModel	Message contains object ID and updates to either, or both, model geometry and object appearance.
AddVirtualDisplay	Projection request message. Message contains object ID, content to project (or URL) and location to project on object.
AddVirtualButton	Interactive user interface request message. Message contains object ID and location of button on object and sensor values to return when clicked.
AddVirtualSlider	Interactive user interface request message. Message contains object ID, location of slider on object and sensor values to return when used.
ObjectMoving	Event message when smart object is moving. Message contains object ID.
ObjectStatic	Event message when smart object is static. Message contains object ID.

2. Architecture Messages Common to Multiple Applications

RegisterDetectionService	Registration request when detection service starts. Message contains uniqueID of hardware.
UnRegisterDetectionService	UnRegister request sent when service stops. Message contains uniqueID of hardware.
RegisterProjectionService	Registration request when projection service starts. Message contains uniqueID of hardware.
UnRegisterProjectionService	UnRegister request sent when service stops. Message contains uniqueID of hardware.
ObjectRegistered	Sent by Server when object registers. Message contains object ID.
RemoveObject	Sent by Database Server when object unregisters. Message contains object ID.
UpdateObjectLocationOrientation	Location and Orientation Hypothesis sent by detection application when object is detected. Message contains object ID, location as 3 numerical coordinates in world coordinate system and orientation as 4 numerical values of a quaternion.
SetButtonState	Button state (pressed or not pressed) sent by detection application. Message contains object ID and sensor values.
SetSliderState	Slider state (in use or not in use) sent by detection application. Message contains object ID and sensor values.
SetObjectLocationOrientation	Location and Orientation update for smart object sent by Database Server. Message contains object ID, location as 3 numerical coordinates in world coordinate system and orientation as 4 numerical values of a quaternion.

<code>GetObjectLocationOrientation</code>	Request for location and orientation of single smart object to detection application. Message contains object ID.
<code>FocusOnObject</code>	Sent by Database Server when it detects interaction with object. Message contains object ID.
<code>FocusOnObjects</code>	Sent by Database Server when it detects interaction with multiple objects. Message contains multiple object IDs.
<code>Object Tracked</code>	Event message when smart object is detected visually. Message contains object ID.
<code>Object Not Tracked</code>	Event message when smart object is not detected visually. Message contains object ID.
<code>Reset</code>	Sent by Database Server to reset all applications to their initial state.

3. Messages between Database Server, Detection Application and Projection Application for colour correction

<code>RequestDetectionUID</code>	Sent by the Projection Service to the Database Server to determine which detection service to contact when requesting images for colour correction. Message contains object ID.
<code>DetectionUID</code>	Reply by Database Server. Message contains unique ID(UUID) of detection service detecting object.
<code>GetObjectSurfacesImage</code>	Request from Projection Service to Detection Service for smart object surface image. Message contains object ID and surface name in 3Dmodel or 3D surface location.
<code>SetObjectSurfacesImage</code>	Reply message from Detection Service. Message contains object ID, surface name or surface location and encoded image.

4. Messages from Pan and Tilt Application to either, or both Detection and Projection Applications (depending on whether the hardware is co-located or separate)

<code>SetCameraLocationOrientation</code>	Sent by PanTilt application to detection application when orientation of steerable camera changes.
<code>SetProjectorLocationOrientation</code>	Sent by PanTilt application to projection application when orientation of steerable projector changes.
<code>SetPanTiltLocationOrientation</code>	Sent by PanTilt application to Database Server when orientation of a steerable platform changes.

6.4 Projector-Camera System Calibration

The architecture incorporates two methods to calibrate steerable projector-camera systems. Calibration can be performed at any time by running a calibration application, which calculates both the projector intrinsic parameter matrix, and the projector-camera transformation matrix (extrinsic parameter matrix). These calculations are identical to those used to calibrate the camera in section 2.6.4; however, here we assume the camera calibration is known a-priori. Both methods use correspondences between known points in the 2D projector image and 3D locations detected in the camera coordinate system (i.e. real-world) for calibration. One important characteristic of these methods is that

they implicitly calibrate any lens-shift on the projector by calculating an off-axis projection frustum in the projector intrinsic parameter matrix.

The first method uses ARToolkit markers as a 3D location system to provide 3D point locations during calibration [Kato and Billinghurst 1999]. The method displays a series of 10 crosses sequentially on the projector. These crosses are arranged as two identical sets of 5 crosses, as shown in Figure 6.11 (left). To perform the calibration we align the centre of the ARToolkit 80mm calibration marker with the 5 projected crosses at a far distance to the projector (e.g. 3m), then repeat the alignment again at a near distance to the projector (e.g. 1m). For each cross location a correspondence between the 2D cross location and the detected 3D marker location in the camera coordinate system is established. The 10 correspondences and camera intrinsic parameter matrix are used to solve the linear system of equations for the 2D-3D projection using SVD. The result matrix can then be decomposed into the projector intrinsic parameter matrix, and the projector-camera transformation matrix. This calibration method takes about 10 minutes to perform and requires the user to manually align the marker with the projected crosses.

The second method is the automatic calibration method proposed by Park et al. using projected structured light [Park, Lee et al. 2006]. This is a modified version of the method presented by Zhang for camera calibration [Zhang 2000] using co-planar 3D points and their corresponding 2D points. In Zhang's method the 3D points detected by the camera are corners on a printed paper chess-board pattern of known 2D dimensions, whereas Park et al. project the chess-board pattern using the projector. A white planar surface is required to be placed where the projector and camera viewing frustums overlap for this calibration, and a camera-to-surface homography established.

To calculate this homography, point correspondences between the camera image and surface can be established either by using vision based detection of the corners of the planar surface, by detecting fiducial markers on the surface, or by simply by identifying the corners pixels manually in the camera image. The homography is calculated using the methods described in section 2.3.6.

As seen in Figure 6.11 (right), in the projector pattern image the corners $m(x,y)$ are the 2-D points and the projected points $M(X,Y,0)$ correspond to the 3D points. The corners in the projected pattern image are detected using the standard computer vision algorithms used to detect the equivalent paper pattern for camera calibration [IntelOpenCV 2007]. The actual 3D location of the projected points are then calculated with the following formula using the camera-to-surface homography (H_{c-o}) and the homogeneous corner coordinates of the camera image c .

$$(X \ Y \ 1)^T = H_{c-o}c \quad (6.5)$$

The relationship between 3-D points (M) and 2-D points (m) is then represented in homogeneous coordinates as:

$$m = PM = H_{o-p}(X \ Y \ 1)^T \quad (6.6)$$

where H_{o-p} is the surface-to-projector homography. Zhang's method is then used with the calculated 2D and 3D correspondences as normal, to recover the projector-camera transformation matrix (P).

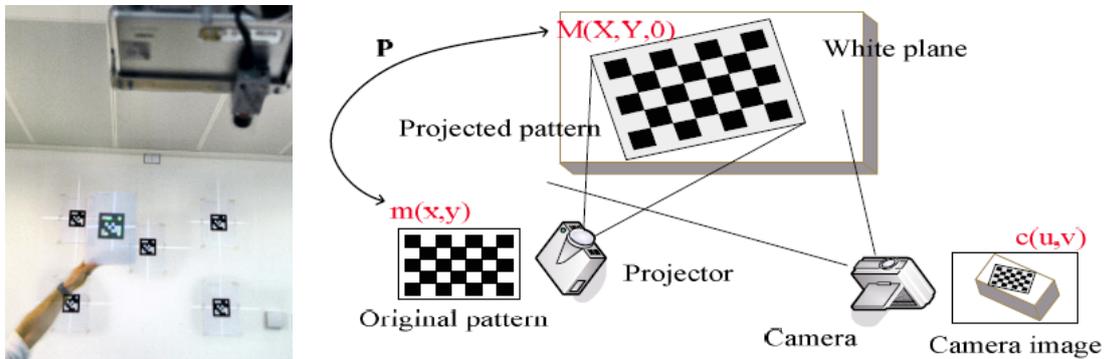


Figure 6.11 (left) Composite image of ARToolkit marker aligned with 5 far calibration locations and handheld for one of the near calibration locations, (right) Projector Calibration using projected pattern on planar surface [Park, Lee et al. 2006]

For steerable projectors, if the projector Centre of Projection (COP) is at the pan and tilt unit Centre Of Rotation (COR), then following calibration we can invert the extrinsic parameter matrix to calculate the camera to pan-tilt coordinate system and hence (with known mounting location and rotation values) calculate the world coordinate system location of objects in the camera field of view.

Following calibration, the projector-camera transformation will only remain valid in systems where the projector-camera relationship is fixed (for example, a static projector and camera, or when the camera is attached to the projector in a moving head steerable projector system). For moving mirror steerable systems we can use an alternate calibration method proposed by Ashdown and Sato [Ashdown and Sato 2005]. Similarly, for projector-camera systems where either the camera or projector (or both) is mobile, but where both intrinsic parameter matrices are known, we could use the imperceptible structured light techniques discussed in section 2.3.6 to continuously project a calibration pattern into a scene and hence attempt to recover the projector-camera transformation.

6.5 Discussion

In terms of the efficiency of the implementation, our approach currently does not detect objects in real time. Our architecture aims for a tracking-by-detection approach, but for the implementation we chose to integrate a particle filter [Pupilli and Calway 2006] to increase tracking frame rates following the first detection by the natural appearance algorithms. The benefit of using a particle filter over a Kalman filter in this case is that it allows us to model multiple alternative hypotheses, so it can automatically integrate detection results from multiple distributed cameras. It also better suits the non-linear movement typically seen in handheld objects [VanRhijn and Mulder 2005].

Use of a tracking algorithm has the added benefit that we can de-couple the projection from the detection step by predicting future object motion from the pose history we maintain in the particle filter. The ability to predict object pose can be exploited by the projector, with its faster frame rate being used to “fill-in” projection on

moving objects between camera frames. This is an issue as the lag of projection is very obvious for users during fast movement. For example, with a combined camera acquisition (33.3ms) and processing step (100ms) of 133.3ms and a typical human walking speed of 5kph the projection location would lag the handheld object by maximum of 18cm. However, reliance on prediction too far in the future can lead to other artefacts, such as lag in fast movement or swimming of the projected image.

In the implementation we identified two limitations due to the Smart-Its sensor nodes used in smart objects. The first is that the wireless network bandwidth only allows a maximum of 2 smart objects to stream sensor data simultaneously while remaining synchronised with a 30Hz (33.3ms) camera refresh rate. This limitation is due to the fixed 13ms timeslots used for each node with the Smart-Its AwareCon networking protocol [Decker, Krohn et al. 2005]. However, this limitation only occurs if the smart objects are not powerful enough to process the state model and abstract sensor data to events so must stream raw sensor data to the Object Proxy.

The second limitation is the maximum available 512KB of flash memory in the Smart-Its device. This limits the size of the Object Model; hence the amount of knowledge and content to project that can be stored in the object itself. We saw in the detection method memory experiments in section 6.6, that the amount of storage required to embed appearance knowledge already varies between 14.32 and 3223.50KB for just a single detection method. Consequently, our solution for larger appearance models or objects with large amounts of visual content to project is to assume availability of a network. We then either just embed a URL link to the whole Object Model in the smart object, or embed just the state definitions part of the Object Model in smart object together with URL links pointing to appearance knowledge and content to project on the network.

6.6 Detection Method Memory Experiments

The Object Model stores trained appearance data for the detection methods, descriptive knowledge about the object such as its 3D Model and its sensors. Additionally, it contains collections of sensor ranges which represent object states. There are 3 sets of states evaluated by the Object Proxy application – at least one appearance configuration (which changes the 3D model and appearance knowledge for the detection system), an optional movement configuration (which represents the moving and non-moving states when an object has movement sensors) and the content configuration (effectively the program logic) which is a set of states together with a link to URL of content to project and the location to project, for each state (see section 6.2.4, Chapter 7 and Appendix B for examples).

In the framework architecture implementation the body of the Object Model is encapsulated in XML. However, the Object Model may also consist of additional files, such as the 3D model and appearance representations for the detection system in standard file formats (such as Mikolajczyk’s affine-invariant feature file format from [Mikolajczyk and Schmid 2002]). Additionally, the Object Model XML file can be split into multiple sub-files to allow sharing of commonly used knowledge and configuration between objects, such as the sensor information (which is identical for all Smart-Its devices), appearance knowledge for identical objects or display model information for two objects that project identical content (see the discussion in section 7.1.8).

When a smart object enters the environment, the type and amount of appearance knowledge stored in the Object Model varies depending on the detection method used to train the appearance representation.

The minimum appearance representation can be created from a single view of the object. A more comprehensive representation requires incorporation of multiple-views to allow detection of the object in more poses when an object is rotated (as shown in section 4.4 experiments). A full appearance model will incorporate views of every surface of the object, for example, from the whole viewing hemisphere. The best angle between views is dependant on the detection method. Each view adds more information to the appearance model, so the angle between views is a trade-off between object coverage and memory requirements. We aim to obtain the best coverage with the least memory requirement.

For example, Lowe recommends using 60° increments between training images for the SIFT local feature algorithm [Lowe 2004]. Using this increment an object requires a total of 10 equally spaced viewpoints for just the upper viewing hemisphere, or 14 for a whole sphere. In contrast, as shown in the section 4.4 experiments, the repeatability of the colour detection algorithm does not reduce significantly with rotation, so for 3D objects, one viewpoint per object surface would suffice.

This experiment aims to evaluate the average memory requirements to enable successful detection of an object with the cooperative framework architecture.

6.6.1 Design

The detection method memory requirements were evaluated by averaging the size of the whole Object Models (including 3D model and any additional files) used in the detection experiments, demonstration applications and over all objects in the training library, for each method. The appearance knowledge results were split into minimum requirements, where we only hold training data for a single view of the object (such as the front surface), and maximum requirement which we define as either a full viewing sphere (with 15 viewpoints at 60° increments) or a representation with 6-viewpoints (imagine the object inside a cube, where the cube's 6 surfaces form the viewpoints) for the colour detection method. 6 views were chosen, as this would give a single view of each surface for 6 of the objects (the cubical objects) in the object appearance library (see section 4.3). However, in practice, uniform coloured objects will require less viewpoints, as each histogram will be similar, allowing viewpoint combination.

6.6.2 Results

The mean average size of 26 VRML format 3D Models is 3.06KB.

The mean average size of 26 Object Model files (including the 3D Models and all display configurations, but excluding appearance knowledge) is 22.65KB.

The mean average size of 101 JPEG format images (with 95% quality) for projection at native projector resolution (1024x768) is 215KB per image.

As can be seen from Table 6.3, the Local Feature appearance description required the most storage, followed by the Shape appearance. The Texture appearance required the least storage. This ranking is identical for both a single viewpoint and a full viewing sphere.

Table 6.3 Memory requirements for Appearance Knowledge and the total Object Model with single viewpoint (minimum) and full viewing sphere (maximum)

Method	Average Object Model Appearance Knowledge Memory Requirement (KB)		Total Object Model Memory Requirement (KB)	
	Minimum	Maximum	Minimum	Maximum
Local Features (SIFT)	214.90	3223.50	237.55	3246.15
Texture (MagLap 32x32 Histogram)	14.32	210.00	36.97	214.80
Colour (LAB 16x16x16 Histogram)	41.00	246.00 (6- viewpoints)	63.65	268.65
Shape (100 points)	59.98	899.66	82.63	922.31

6.6.3 Discussion

The results indicate that the largest use of memory for all objects is generally the appearance knowledge (between 14.32 and 3323.50KB, compared to an average of just 22.65KB for the rest of the Object Model). However, the total memory requirement for the smart object will vary depending also on the number of projections (hence number of content configuration states in the Object Model) and whether the content to project is stored in the object with the Object Model.

For example, the scenario described in section Appendix B.1, where a Chemical Container detects rough handling contains only 4 content configuration states and two images for projection requests. The Object Model with a single local feature appearance viewpoint of the front of the container is 202KB. With two images (430KB) the total size is 632KB. This can easily be reduced below 512KB to fit in a Smart-Its device, either by storing the images on the network and embedding only a URL link to them in the Object Model (as discussed in section 6.5), by storing the images with higher compression ratios, or reducing image resolution.

In contrast, the smart photograph album described in section 7.2 (below) contains many content configuration states for different projections, which depend both on the state of the light sensor and the projected buttons. The total Object Model size with two local feature appearance viewpoints and 3D models (front cover and inside) is already 898.3KB. The 18 photographs used in the demonstration add 2325KB, for a total size of 3223.3KB (3.14MB). This size of Object Model cannot be stored in the 512KB memory of a Smart-Its device. Consequently, we store only a URL to the Object Model and its associated files in the smart object and assume the files are available on the network.

While this limitation may be overcome in the future with increased memory capacity in smart objects, the actual transmission of large Object Models (on the order of Megabytes) from the Smart Object is still a big issue. The communication ability of the current generation of sensor nodes is typically limited, due to power, processing and network synchronisation constraints; hence, the transmission of large files would not be reliable, or fast. For example, a particle Smart-Its device using the Awarecon protocol transmits 64 bytes every 13.11ms at full throughput (equivalent to 4.88KB/s). Hence a 3MB Object Model would take a minimum of 10.74 minutes to send to the projector-camera system, assuming no other devices are also transmitting. Similarly, an 802.15.4

node such as the imote2 with TinyOS and default radio parameters has an average transmission rate around 250 packets per second, each with 28 bytes (equivalent to 7.00KB/s). Hence, a 3MB Object Model would take a minimum of 7.49 minutes to send, assuming no other nodes are also transmitting.

6.7 Pose Calculation, Pose Jitter and Projection Accuracy Experiments

These experiments investigate the pose and projection accuracy an operational system can expect when a target object has been detected in the image. These factors are especially of interest in Augmented Reality and Mixed Reality systems as the virtual image is overlaid either on a view of the physical world, or on the actual physical world with projector-based AR. Mis-registration of the image, or movement of the augmenting image relative to a static object in the real-world (jitter) is immediately detectable by an observer and can distract from the augmentation content itself.

We perform experiments to answer the following research questions:

- R1)** How accurately can an object's location and orientation (pose) be calculated?
- R2)** How accurately can a projection be located on a planar object's surface when it is orthogonal to the projector?
- R3)** What jitter in the calculated pose and projection can be expected?

6.7.1 Design

An accuracy and jitter evaluation dataset was created as part of the experimental procedure to answer questions R1 to R3. The dataset contains the 5 objects from the object appearance library (see Chapter 4) with uniformly planar surfaces. These objects were the Book, Product box, Card, Notepad and Cereal box. The objects, were placed at 6 locations arranged as a grid in X,Y,Z camera coordinate space, with the object front surface parallel to the camera sensor as shown in Figure 6.12. The projector-camera system was placed in a static location at 3m distance from the grid, orthogonal to the object XY plane. The object was moved $\pm 0.5\text{m}$ from this location in the X and Y axes in turn. The grid and movement distance were chosen to maximise the movement size, while ensuring even the largest object was completely within the field of view of the projector at 3m. At each location 100 images of each object were captured and manually annotated with the object bounding box ($6*5*100=3,000$ images). The camera, lens and image capture resolution were identical to those in the object appearance library in Chapter 4. We calculate that with a 12mm fixed camera lens, 2/3" camera sensor and 1280x1024 resolution the 40.27° camera Horizontal Field Of View (HFOV) gives each camera pixel has an effective horizontal resolution of 0.03146° (1 arc-minute, 53 arc-seconds). For a mid-zoom setting (20) of the 1024x768 resolution projector, the HFOV is 26.42° , and Vertical Field Of View (VFOV) is 19.32° . Each pixel has a measured size of approximately 1mm^2 at 3m.

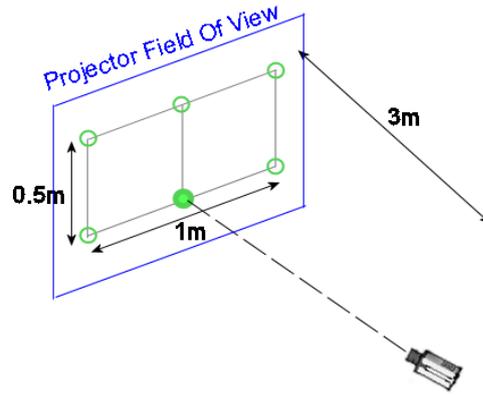


Figure 6.12 The location accuracy and jitter experiment object grid locations, orthogonal to the camera. Object test locations are the green circles.

6.7.2 Procedure

6.7.2.1 Research Question R1

The investigation of question R1 is split into two sub-experiments.

The first investigates orientation accuracy using the rotation images of the 5 objects from the object appearance library. This sub-experiment evaluates both 2D rotation in the camera plane (r_z) between 0° and 350° and general 3D rotation (r_x, r_y) – in this case around the object's Y axis between -70° to $+70^\circ$. Both exclude the 0° orientation as this pose closely matches the one used for training.

The detection system was trained using images of the corresponding object's front surfaces at 3m, from the scale image set. For both the 2D rotation and 3D rotation images a local feature detection step was performed on each test image of the objects, constrained to detect features only inside the object bounding box (simulating a near perfect detection system). The detected features were matched to the training image using Sum-of-Squared Distances (SSD) nearest neighbour matching, establishing feature correspondences. The pose calculation algorithm was provided with the corresponding image and object features and the camera calibration. The pose results and error from the manually measured pose was recorded. Failed detections were excluded from the error calculation (2D rotation had no failures, while for 3D rotation the rate was: Book 33.3%, Box 40%, Card 40%, Notepad 40%, Cereal Box 60% of the -70° to $+70^\circ$ range, with all failures evenly split between the two rotation extremes).

The second sub-experiment investigating location accuracy in the pose calculation algorithm is part of the following procedure to also answer R2 and R3.

6.7.2.2 Research Questions R2 and R3

Initially, the projector intrinsic parameters (optical parameters) and the relative extrinsic pose between projector and camera were calibrated using the method discussed in section 6.4, with the projector lens zoom set to 20 (mid-zoom) and the camera and projector image focused at 3m.

Then, for each of the 5 objects and at each of the 6 object locations in the accuracy and jitter evaluation dataset we perform the method described below.

The detection system was trained using an image of the corresponding object at 3m (the centre of our working range), from the scale images in the object appearance library. A local feature detection step was performed on each of the 100 captured test images of the object, constrained to detect features only inside the object bounding box (simulating a near perfect detection system). The detected features were matched to the training image using SSD nearest neighbour matching, establishing feature correspondences. The pose calculation algorithm was provided with the corresponding image and object features and the camera calibration and the pose results recorded. To answer question R1, the median pose over the 100 images was used when calculating the pose error for each location, to remove location jitter. The pose error was calculated by taking the difference between this median pose and the distance manually measured to the object by a tape-measure in the X,Y,Z camera coordinate system. We assume the manual measurements are accurate to $\pm 5\text{mm}$. Failed detections were excluded from the error calculation (only the notepad had failures, 6/600=0.01% rate).

To answer question R2, the median pose calculated over the 100 images at each object location was used to also project onto the front surface of the object as the accuracy evaluation dataset was being created. The projection image was a 1 pixel line bounding box, exactly the size of the object. The 2D (X,Y) offset of the projected image corners relative to the physical object front corners was measured with a ruler. We assume this measurement is accurate to $\pm 1\text{mm}$.

To answer question R3, we use the individual pose results from the 100 images of the location accuracy experiment. Pose location jitter is calculated as the difference between the individual poses in the 100 images and the median pose for the location. These results are averaged over each location, for every object. For projection jitter we project the bounding box image for each of the 100 calculated poses in sequence, while measuring and recording the 2D (X,Y) offset of the projected image corners relative to the physical object with a ruler. We calculate the median error of the offset results. We assume this measurement is accurate to $\pm 1\text{mm}$.

6.7.3 Apparatus

A 3.4GHz dual core Pentium-4 computer running Windows XP SP2 was used for all experiments. The pose calculation and projection geometry correction algorithms were implemented in C++ using Intel OpenCV API for image processing and OpenSG scenegraph to create the image for projection (see section 6.3.2 for more implementation detail).

6.7.4 Results

The 95th percentile measurements in the tables below give the error values at which the corresponding cumulative distribution reached 0.95, where 95% of all measurements will have an error equal or smaller than that value.

6.7.4.1 Research Question R1

As can be seen in Table 6.4, the median orientation error and 95th percentile results over all objects are similar for both 2D in-plane (Mdn=0.93°, P95=1.26) and general 3D rotation (Mdn=1.02°, P95=1.60).

The most accurately detected 2D in-plane rotation was for the Book object (Mdn=0.56°), with the least accurate being the Box (Mdn=1.33°). The most accurately

detected 3D rotation was for the Card object (Mdn=0.51°), with the least accurate being the Notepad (Mdn=1.60°).

Table 6.4 Median Rotation Error results for each object, over -70° to +70° Out-of-plane and 10° to 350° in-plane.

Object	2D Rotation Error (°)	3D Rotation Error (°)
Book	0.56	1.59
Box	1.33	1.03
Card	0.84	0.51
Notepad	0.93	1.60
Cereal Box	1.00	0.72
Median Error	0.93	1.02
95th Percentile	1.26	1.60

As can be seen in Table 6.5, over all objects the Z-axis (distance to object) errors (Mdn=12.13mm) were generally higher than X and Y axis errors (Mdn=4.70 and 7.15mm respectively). Similarly, the variation in the Z-axis errors between objects was greater than both the X and Y axes, as demonstrated by the larger value for the 95th Percentile (P95=23.63). The most accurately detected object was the Cereal box (Mdn=8.66mm) for 3D error from the true location, while the least accurately detected object was the Book (Mdn=26.82mm). The median combined 3D location error of the detection and pose calculation system over all the objects was 14.25mm, P95=25.69.

Table 6.5 Median Location Calculation Error in X,Y,Z Camera Coordinate System for each object, averaged over all grid locations

Object	X Error (mm)	Y Error (mm)	Z Error (mm)	Combined 3D Error (mm)
Book	7.22	5.14	25.31	26.82
Box	4.97	11.76	16.87	21.16
Card	2.19	7.15	12.13	14.25
Notepad	2.60	8.11	5.46	10.14
Cereal Box	4.70	6.12	3.94	8.66
Median Error	4.70	7.15	12.13	14.25
95th Percentile	6.77	11.03	23.63	25.69

6.7.4.2 Research Question R2

Table 6.6 Median Projection Location Error on the Object X,Y front surface plane for each object, averaged over all grid locations.

Object	X Error (mm)	Y Error (mm)	Combined 2D Error (mm)
Book	7.75	9.50	12.26
Box	18.00	11.00	21.10
Card	6.25	20.25	21.19
Notepad	7.50	15.00	16.77
Cereal Box	3.50	10.25	10.83
Median Error	7.50	11.00	16.77
95th Percentile	15.95	19.20	21.17

As can be seen in Table 6.6, for projection the error in the X-axis is generally lower than in the Y-axis, except for the Box object (Mdn=18.00mm). However, the variation in the error in the Y-axis between objects is lower than in the X-axis. The object with the most accurate projection was the Cereal Box (Combined 2D Error Mdn=10.83mm), while the Card was the least accurate (Mdn=21.19mm). The median 2D location error of the projection over all the objects was 16.77mm, P95=21.17.

6.7.4.3 Research Question R3

As can be seen in Table 6.7, over all objects the 1.64mm median jitter in the Z-axis (distance to object) is much greater than both the X and Y axes (Mdn=0.20 and 0.26mm respectively). Similarly, the variation in the Z-axis errors between objects was greater than both the X and Y axes. The Cereal Box object has the highest jitter (4.15mm median 3D Jitter) around the median location, with the Card having the lowest (1.20mm median 3D Jitter). Over all objects, the median combined 3D jitter around the median object locations was 1.65mm, P95=4.04.

With the mid-zoom projector setting, the size of 1 pixel was 1mm² at 3m distance from projector. The projection jitter for the Card, Notepad and Cereal Box objects was observed at less than 1mm, too small to measure accurately with a ruler. Consequently, it was estimated as 0.5mm for the Card, Notepad and Cereal Box X and Y axes.

Table 6.7 Median 3D Object Location Jitter from Median Location for each object, averaged over all grid locations.

Object	X Jitter (mm)	Y Jitter (mm)	Z Jitter (mm)	Combined 3D Jitter (mm)
Book	0.20	0.26	1.60	1.63
Box	0.98	0.86	3.65	3.87
Card	0.17	0.07	1.18	1.20
Notepad	0.13	0.08	1.64	1.65
Cereal Box	0.98	0.63	3.98	4.15
Median Jitter	0.20	0.26	1.64	1.65
95th Percentile	0.98	0.81	3.92	4.04

Table 6.8 Median Projection Location Jitter from Median Location on the Object front surface plane for each object, averaged over all grid locations.

Object	X Jitter (mm)	Y Jitter (mm)	Combined 2D Jitter (mm)
Book	1.00	1.00	1.41
Box	11.00	1.00	11.05
Card	0.50	0.50	0.71
Notepad	0.50	0.50	0.71
Cereal Box	0.50	0.50	0.71
Median Jitter	0.50	0.50	0.71
95th Percentile	9.00	1.00	9.12

As can be see in Table 6.8, the median Y-axis projection jitter from the median pose is generally equal in the X and Y axes, except for the Box object. Additionally, the variation in the jitter in the X-axis between objects is much higher than in the Y-axis, as can be seen from the 95th Percentile results (P95=9.00 for X, 1.00 for Y). The objects

with the least jitter were the Card, Notepad and Cereal Box (all estimated at 0.71mm of 2D jitter), while the Box had the highest (Mdn=11.05mm). The median combined 2D jitter of the projection from the mean object locations, over all the objects was 0.71mm, P95=9.12. This is approximately equivalent to 1 projector pixel jitter at 3m.

6.7.5 Discussion

6.7.5.1 Research Question R1

To address R1 we performed two sub-experiments measuring location and orientation calculation accuracy following object detection. The orientation experiment was split into 2D and 3D object rotation, however, the results (Table 6.4) from both parts of the experiment were almost identical. This suggests that the magnitude of the orientation calculation error is consistent, irrespective of the type of rotation the object undergoes. The orientation calculation was on average accurate to 1.02° or better. This does not quite meet the rotation aim we set in section 6.3 (a maximum of 1° median error).

As can be seen in the location experiment results (Table 6.5), the distance to object (Z) error is significantly higher than error in the camera X,Y plane. This is due to the difficulty involved in monocular camera systems calculating depth. Monocular systems rely on the perspective effect caused by a pinhole camera model. In this case, the distance of features on the object from each other increases the closer the object is to the camera, and decreases when the object moves further away. Due to the finite resolution of the camera, the exact pose therefore becomes more uncertain at larger distances to the object. Even a single pixel change in the location of features relative to each other can make a big difference. Stereoscopic camera systems alleviate this problem (to some extent) as they can make use of a known baseline separation between the cameras to triangulate the features. However, although increasing the stereo baseline increases the Z accuracy, it is a trade-off with the useable tracking volume, as both cameras require the same features in their field of view.

6.7.5.2 Research Question R2

The experiment to address R2 aims to understand how accurately the combined system (detection, pose calculation and projection) can project onto objects in the real world. Hence, the projection error shown in Table 6.6 is a combination of the accuracy of the camera intrinsic parameters calibration, location error from the pose calculation, the accuracy of the projector intrinsic parameters calibration and the accuracy of the calibrated transformation between the camera and projector (extrinsic parameters). The higher error, but lower variation in the Y-axis relative to the X-axis suggests possible calibration error, producing a fixed offset in the Y-axis.

Any error in the Z-axis (distance to object) from the pose calculation will appear as scaling errors in the projected image, increasing the measured X,Y errors. Consequently, the error in location and projection can be compared best using the respective combined 3D and combined 2D error figures. In this case, the results show that following an accurate calibration of both projector and camera intrinsic and extrinsic parameters, the median combined 2D projection error (16.77mm) is similar to the median combined 3D error in the pose calculation step (14.25mm). This suggests that the major source of error in the projection is the pose calculation step. The 16.77mm median projection location error we achieve from the combined system meets the original implementation aim in 6.3 (a maximum of 20mm median error).

6.7.5.3 Research Question R3

To address R3 we measured the jitter of individual poses from the median pose for both the location pose calculation (Table 6.7) and for projection (Table 6.8). As can be seen when comparing Table 6.5 and Table 6.7, (with exception of the card object) generally objects with higher combined 3D location errors have higher jitter. Additionally, it can be seen when comparing Table 6.7 and Table 6.8 that the Combined 3D location jitter (Mdn=1.65mm) is much higher than the combined 2D jitter in the projection (Mdn=0.71mm). Looking closer at location jitter in Table 6.7 we see that (similar to the location error in Table 6.5) the largest component of the 3D jitter is from error in the Z-axis (Mdn=1.64mm).

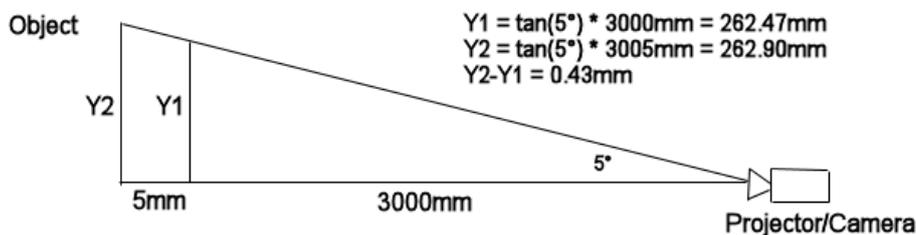


Figure 6.13 Error in Z-axis location for objects at a large distance from the camera causes smaller error in the X,Y location error of the projection

The reason this large Z-axis jitter does not translate into large jitter in the projection X and Y axes is because error in the Z-axis contributes much less than error in the X,Y axis, when objects are at a large distance to the projector. For example, with an object at 3m, 5° off the camera axis, error of 5mm in the Z-axis leads to only 0.43mm error in both the X and Y axes of the projection, as illustrated in Figure 6.13.

The results from the experiments are representative of the average accuracy obtainable with planar objects and the current camera and projector optical characteristics. Varying the camera lens or projector optical parameters will vary the results. For example, increasing the focal length lens (zooming-in) of the camera allows a higher accuracy in the pose calculation due to the higher spatial resolution, however, it trades-off field of view so large objects may no longer fit in the camera image.

6.8 Conclusion

In this chapter we presented an architecture design and implementation validating the concept of cooperative augmentation. The implementation is a working system which enables us to investigate different aspects of the architecture and Cooperative Augmentation concept.

The implementation enables augmentation of smart objects with a display capability without changing their natural appearance, using projector-camera systems. Our approach can locate and track mobile objects in the environment, align the projection with the object's surfaces and correct for surface colour so the display appears undistorted and visible to a user. The main challenges in our approach are visual detection of smart objects, keeping the projection synchronised when the object is moved or manipulated and correcting the projection for non-ideal surface colours and textures.

Our system implementation is currently geared more towards accuracy in detection and pose calculation, for example, we use a high resolution machine vision camera with 1280x1024 pixel resolution. However there is a trade-off between accuracy and detection runtime, hence our system does not currently achieve real-time detection or tracking. We will never achieve good accuracy from a poor camera image (e.g. from a 320x240 pixel web-camera), however we can eventually achieve real-time detection with high resolution cameras either through hardware improvements or implementation improvements. One major improvement can be readily made by converting more of the detection processing to use the GPU.

From the pose calculation and projection experiment we determined that our system calculates the pose of planar objects with a median 3D error of 14.25mm from the correct position. The combined system can also project onto a planar object with a median 2D error in the location of the projection around 16.77mm.

While this accuracy is sufficient for large objects such as the chemical container (see section 4.3), this might be more problematic with small objects due to their size. Here, a large error on a small object such as the cup can potentially offset the projection enough that part of the display is not visible, hence the importance of projection accuracy depends to some extent on object size. To help in detecting small objects, it may be useful to further characterise the detection algorithms and incorporate prioritisation of algorithms which have a higher detection performance with small objects into the method selection, as object size can be determined from the 3D model.

The jitter experiments showed that although the median 3D pose jitter was 1.65mm, in the combined system projection only a median of 0.71mm jitter was observed, due to error in the Z-axis (distance to object) contributing significantly less to the projection jitter than error in the X and Y axes. While small, this jitter is visible in active projections when observing at close range. Jitter is also visible when objects are static for the reasons discussed in section 2.6.5. However, if smart objects possess movement sensing capability it is possible to smooth the pose to remove jitter only when we know the object is static. This avoids the lag on instantaneous movement typically associated with continuous smoothing.

In section 6.5 we found that smart objects are typically limited in memory. Hence we studied the memory requirements for storing the Object Model. For the memory requirement we determined that the largest component was either the object appearance description or content for projection (e.g. images and video), but this was application specific. These findings are important, as the network transmission time is typically prohibitive for large Object Models. Hence, full appearance models with large or complex amounts of content to project must be stored outside the smart object on the network. This is not a problem as long as the object maintains a link to the location of the content on the external network resource and the external resource remains valid.

Chapter 7 Demonstration Applications

Three applications were developed to demonstrate the capability of the system architecture. These demonstrate three distinctly different areas of the architecture and discuss some of the issues involved in developing applications with our architecture:

1. A smart chemical container demonstrator, illustrating the dynamics of the whole cooperative process from initial object registration to knowledge updating. This uses pose results from the vision-based detection system to infer whether the smart container object is stored in the correct location, and with the correct chemicals. A warning projection is requested if detected in an incorrect location.
2. A smart photograph album demonstrator, illustrating interaction methods in the architecture. Here we concentrate on three interaction methods, allowing manipulation of the object location, object geometry and interaction with projected buttons to browse photographs.
3. A smart cooking scenario, illustrating multiple projector-camera systems augmenting multiple cooperating smart objects in order to provide the user with context-dependant recipe instructions to accomplish a cooking task.

7.1 Smart Cooperative Chemical Containers

This demonstrator aims to illustrate the dynamics of the whole Cooperative Augmentation process by following a smart chemical container object from registration on initial entry into the environment, to first detection, use of embedded sensing, projection, interaction, knowledge updating then final exit from the environment.

To motivate this demonstrator we use an industrial goods warehouse scenario where two smart chemical containers enter a warehouse and we would like the containers to warn the employee visually when a container is stored outside the correct storage area. We use an identical approach to that presented by Strohbach, Gellersen et al., where safety-critical storage areas are monitored by the smart chemical containers themselves [Strohbach, Gellersen et al. 2004]. Here rules for the safe handling of objects and materials are embedded in the objects and objects cooperate to detect hazardous situations (such as being stored next to reactive chemicals) using embedded sensing.

7.1.1 Object Model

The Object Model appearance knowledge is initially trained with images of the Chemical Container at 3m, from the scale images in the object appearance library (see Chapter 4). The LAB Colour detection algorithm is run on the image in the annotated

area containing the object, to extract a 16x16x16 bin LAB histogram. For this demonstrator we designate a 1m² and 0.5m high 3D volume in the environment as being an approved storage area (X=1 to 2m, Y=0 to 0.5m, Z=1 to 2m). States are defined for when the object is in the correct storage area, and the incorrect area, as shown in Figure 7.3. An identical object model is embedded in all Chemical Container objects.

7.1.2 Registration

As shown in Figure 7.1 (left), an employee enters the environment with two chemical container smart objects. The objects enter proximity of the projector-camera system; detect the presence of a display service and register. This process transfers Object Model from the object to the projector-camera system.

The projector-camera system registers the objects, and returns a confirmation message to the containers. On receipt of this message the containers begin sending sensor events to the projector-camera system. In this case, they are being carried by the employee so embedded accelerometer sensors generate movement events.

7.1.3 Detection

The registering objects trigger the detection process in the projector-camera system. Here the challenge is to simultaneously detect mobile or static objects and distinguish between objects with similar appearances.



Figure 7.1 (left) New objects arrives in environment, (centre) An employee walks with containers, (right) The employee places one object on the floor

The steerable projector now rotates from its current position to search the environment. As the objects have just entered, the system does not know their location. Consequently, the projector system uses a creeping line search pattern with a horizontal major axis to thoroughly search the whole environment.

The projector uses the appearance knowledge embedded in the Object Model and the sensor events to configure its detection process. In this case the containers store knowledge of a colour histogram, and sense they are moving. This knowledge triggers the method selection step to choose colour and movement detection processes. The movement process generates a motion mask which is used by the colour detection process to constrain its search for the object by masking the back-projection result of the object's colour histogram.

As the two chemical containers look identical, two possible objects are identified in the image. It is not currently possible for the camera to distinguish between the objects.

Consequently the steerable projector tracks the moving areas in the camera image by centring their centre of gravity.

Both objects generate movement event messages while they are being carried by the employee. However, when an employee places one of the containers on the floor, as shown in Figure 7.1 (right), the container's movement sensors stop sending movement events. The projector-camera system now only detects one moving area and the system can differentiate between the objects directly based on sensing. A 3D location and orientation is now calculated (for an example see Figure 7.2) and sent wirelessly to the containers, completing the Object Model.

7.1.4 Projection

Once an object's 3D location and orientation is calculated by the projector-camera system, objects can request projection of content on their surfaces. Here the challenge is to correct the projection for the orientation of the object, and variations in its surface colour to ensure the most undistorted and visible projection.

In this case, the container detects it was put down in the wrong storage area based on the location it was sent and requests a warning message is projected (see Figure 7.2). The projector-camera system projects the warning message on the front surface of the container objects with geometrical correction, to appear undistorted.



Figure 7.2 (left) Detected container with green wireframe 3D Model superimposed on the camera view using calculated pose, (right) A warning message projection on two chemical containers

7.1.5 Manipulating the Object

When projecting onto objects, the object can respond to sensed manipulation or network events by dynamically modifying the projected content. The challenge here is to keep the projection aligned with the object as it is manipulated or moved.

The employee sees the projected message and picks up the object. The detection process continues to track it and generate 3D location and pose information. Consequently, the message appears to remain fixed to its surface as long as the surface is visible to the projector system. When the object is in the correct area it requests the projection stops. The employee puts down the container and sees the message disappear. The projector-camera system keeps tracking the objects.

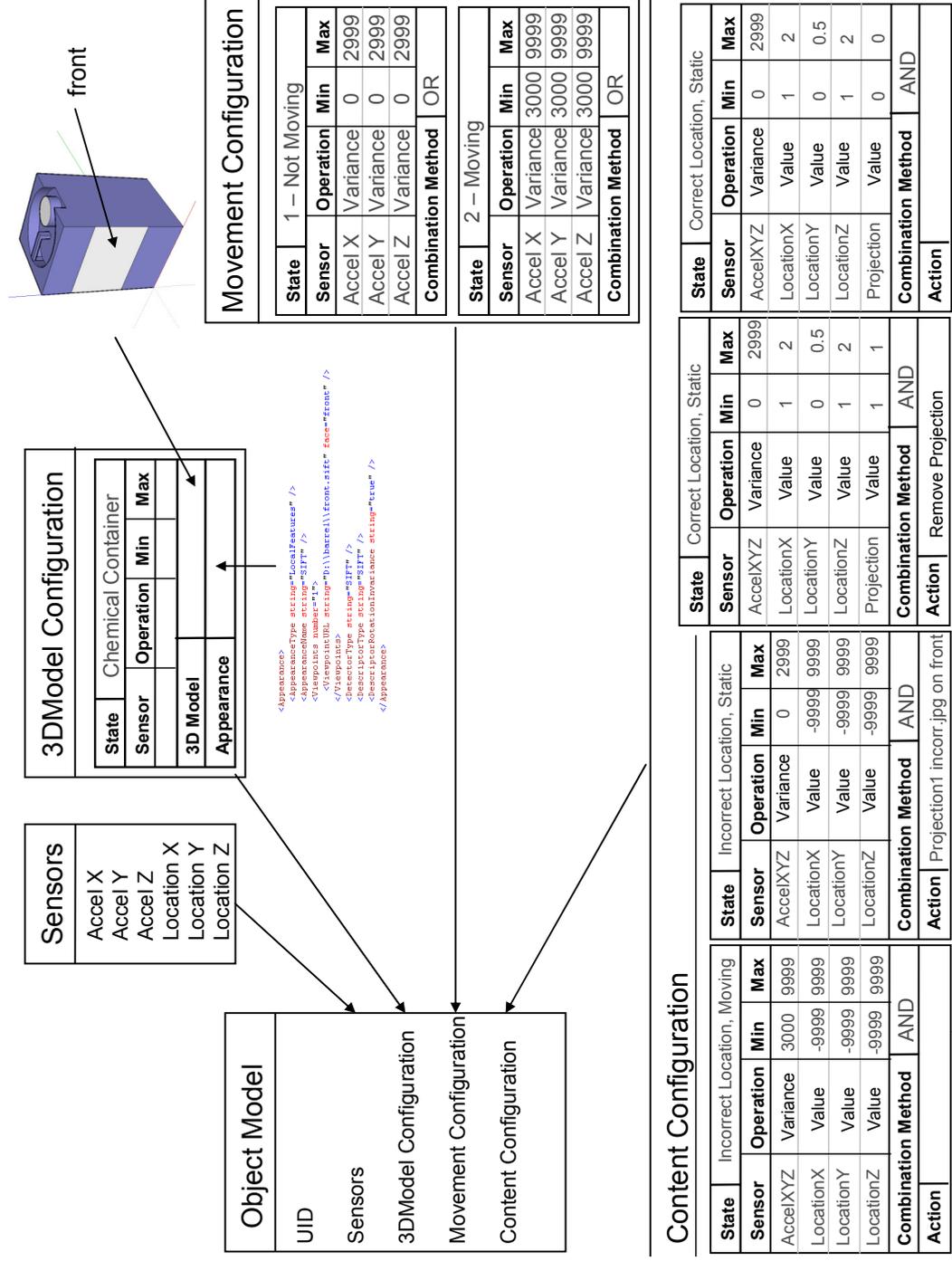


Figure 7.3 Partial Object Model representation of Chemical Container Demonstrator

7.1.6 Knowledge Updating

If objects enter the environment with only partial knowledge of their appearance, their knowledge can be increased over time by performing extra detection processes and re-embedding the result into the Object Model. The challenge here is how to make the knowledge extraction accurate, given that the initial knowledge was incomplete.

The two containers entered the environment only with knowledge of their colour, so the projector-camera system extracts more appearance knowledge over time. In this case, the SIFT algorithm [Lowe 2004] is used to detect scale and rotation-invariant features on the object just put down, as shown in Figure 7.2. The SIFT descriptors are calculated on small image patches around the detected interest points. The resulting 128 value feature vectors are mapped from the image to locations on the object's 3D model using the known 3D location and orientation of the container.

If the object is manipulated so it is rotated from its original pose new features will be detected as they come into view. The projector-camera system manages the Object Model local feature database to add new features or update the database if the object appearance is changed. The new local feature appearance knowledge is sent to the smart containers to be embedded in the Object Model and used for faster, more accurate detection in future.

7.1.7 Objects Departing the Environment

When objects depart the proximity of the projector-camera system, their virtual object representation is removed by the projector system and the projector is free to track other objects. Here, the employee moves to the exit with the container that was never put down. This container continues to generate motion events. As there are no other moving objects or projections active, the projector system tracks the carried object, as shown in Figure 7.4.

As the employee exits through the door with the object, the system loses sight of the object and it no longer responds to messages from the projector-camera system. The system assumes it has departed the environment after a short time-out.

The projector-camera system then returns to the last-known position of the other container objects. If no objects are detected the projector system begins an expanding square search pattern centred on their former locations.

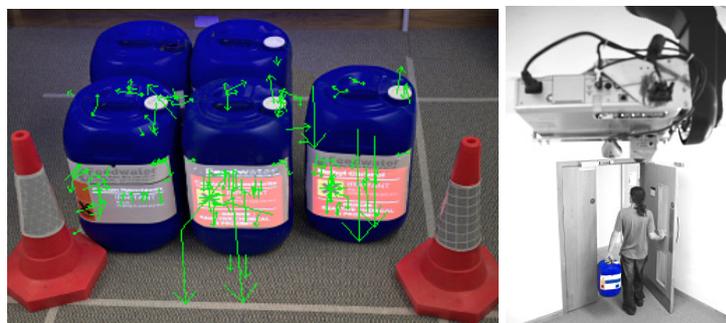


Figure 7.4 (left) Scale and rotation invariant local features detected on chemical containers, (right) A container leaves the environment with the employee

7.1.8 Discussion

The chemical container demonstrator provides important validation for our architecture and approach. Firstly, it successfully illustrates visual display on an object both while moving and when static. In both cases the projection appears fixed to the display location on the object. Secondly, it illustrates the interactivity of our approach, as the projected display changes automatically based on the states defined in the Object Model when the user moves the object to the correct location. Thirdly, it illustrates the benefit of embedded sensing in the detection process. Here movement sensing makes the detection more robust when there are two identical objects by constraining the detection and masking the distracting object. Fourthly, we demonstrate the knowledge extraction and updating process by detecting local features and re-embedding this appearance knowledge into the object after initial detection with the colour cue. Fifthly, we show that the combined system detection, pose calculation and projection output is accurate enough to be visible on the barrel, both when it is mobile and when static.

7.1.8.1 Projection Resolution

On a chemical container with an area of size 200x130mm suitable for projection, the achievable projector resolution with a fixed mid-zoom setting (20) and object at 3m is around 1 pixel per millimetre. This allows a resolution of 200x130 pixels on the barrel when it is orthogonal to the projector. A typical font can be created from a minimum of 8x8 pixels, allowing a maximum of around 25x16 text characters to be displayed. However, any geometric correction required due to the relative pose of the container surface will reduce this resolution further. Consequently, the projection is more suitable for large text warning messages, images and video than for large blocks of text. Changing the focal length of the projector (by zooming) to achieve a higher resolution projection is possible, but has the trade-off of reducing the field of view.

7.1.8.2 Object Cooperation to Overcome Occlusion

A safety-critical system requires certain attributes such as guaranteed availability, reliability, integrity and dependability. The scenario presented by Strohbach, Gellersen et al. [Strohbach, Gellersen et al. 2004] relies on the containers themselves to sense when hazard situations occur using only ultrasound sensors on the objects and cooperation with other objects via a wireless network.

Interactive projection in a safety-critical storage scenario faces the problem of guaranteed availability, as projector-camera systems rely on direct line of sight to the object. Occlusion of the object prevents both detection and augmentation with displays. However, in this case, it would be possible for objects to cooperate and achieve a display. For example, if we imagine a scenario where two identical objects exist in the same environment; the first object senses it has been stored outside an approved area, but is occluded from the projector-camera system by the other object, which is stored in front of it. In this case the framework is aware of the object, but cannot detect or project a warning message on it; it is only able to detect and project on the second object.

To enable the first object to use the second as a display surface, we introduce an abstraction to our framework of an Object Class. This allows smart objects to share common attributes (similar to the Object-Oriented Programming (OOP) paradigm) in their Object Model. Shared attributes could be shape and appearance (such as the smart chemical containers), common functionality (such as two smart cups with different

appearances), or be part of a common task (such as individual smart ingredients in a recipe task).

7.1.8.3 Object Knowledge Sharing

We allow objects in the same class to share knowledge using the same mechanism with which objects share sensor data. This would allow a brand-new object to enter the environment, query the network to see if any other objects of the same class exist. If an object of the same appearance is present the object can request parts of the Object Model (or the URL), such as recently trained appearance knowledge, to help overcome the hurdle of initial detection.

The smart object is now composed of public and private data. Public data is the Object Model components shared with the projector-camera system. Private data consists of the raw sensor values and program logic encoded in the Object Model to change the content model, the movement model and the geometry model. The private data remains with the physical object or Object Proxy.

The occluded chemical container object in our scenario now queries the network to see if any objects of the same class exist. The visible smart object responds, and the object can query the object location. If an object knows its current location, it is visible to the projector-camera system and can support projection. We allow objects in the same class to modify each other's projections only if they have the same geometry and no other projection is currently active. In this case, the two objects are identical, so the occluded chemical container requests a warning projection on the visible container. The occluded container monitors its location and as soon as it is visible, or in the correct storage area, it removes the projection from the visible object.

Allowing objects to query other object's locations and sensors allows flexibility in application design. For example, it allows modelling of spatial relationships between objects (such as distance or angle between two objects, whether the object is left, right, in-front or behind, etc..) as described by Kortuem et al. [Kortuem, Kray et al. 2005]. These relationships form context in the environment containing the objects. This context allows us to dynamically adapt our projection, for example, it enables applications such as a game which is only projected on a set of objects when they are all brought together into the same location.

7.2 Smart Photograph Album

The Photograph Album is a manufactured object designed specifically to demonstrate three interaction methods in the cooperative augmentation framework:

1. Manipulation of object location
2. Interaction by object geometry manipulation
3. Interaction with projected user interfaces

The object itself is a physical book, but is given a new appearance using a dustcover on the outside, and a new appearance inside with a pasted-in page. A Smart-Its device attached to the album detects when it is moving with accelerometers, and detects when it is opened with a light sensor. This scenario involves two users, one of whom has just returned from a recent holiday and would like to show the second user their

photographs. Similar to the Magic Book demonstrator developed by Billinghurst et al. with fiducial-marker based AR [Billinghurst, Kato et al. 2001], when the book object is detected by the projector-camera system it becomes automatically augmented with the photograph display.

7.2.1 Object Model

The Object Model appearance knowledge is initially trained with images of the Photograph Album captured with the object placed orthogonal to the camera at 3m distance, and manually annotated with an object bounding box. Two images are captured – the front cover and inside when opened, representing two possible object geometries (closed and open). See Figure 7.5 for an example of the front and inside. Local features are extracted from the annotated bounding box regions using the SIFT detection algorithm. 3D Model and appearance states are defined for when the object is open and closed, as shown in Figure 7.6.

The object uses interactive projected buttons to browse left and right through images in the album. We constrain the interface so users cannot press left and right simultaneously, which allows us to model both buttons with a single button state. The projected left and right buttons now simply return the number of the next image to change to. The buttons are modelled as interactive areas, in which the user’s finger is detected and matched to a fingertip template by the detection application. The location of the interactive areas is specified as surfaces defined in the object’s 3D Model.

The “Content Model” is a hierarchy of albums, images and buttons which defines the image to project, based on the button state and whether the object is open or closed. We consider each projection a separate state with its own “Display Model” stored in the Content Model. Each Display Model stores the button locations and button state value, together with the state value the buttons return when pressed, as shown in Figure 7.7.

There are 3 albums, each with an image for the front cover, with the name of the album superimposed on the top. Each album has 5 images for projection inside when the album is opened, representing individual button states (see the numbers in Figure 7.7). For each projection, the left buttons are defined in the Display Model to return the value of the image to the left of the current image. Similarly, the right buttons return the value of the image to the right. The values wrap around, so pressing right from image 5 returns to image 1 and pressing left from image 1 goes to image 5.



Figure 7.5 Photograph Album smart object with Projection (left) front cover, button state 10 (centre) being opened, (right) inside, button state 5

7.2.2 Registration and Detection

Two users enter the environment with the first carrying the smart Photo Album. The object registers and transmits its Object Model to the projector-camera system. The smart object detects it is moving with accelerometers, and detects it is closed with the light sensor. The object evaluates its 3D Model configuration states and broadcasts it is in state 1 – Book Closed, sending the projector-camera system a 3D Model of the closed object and a the appearance description of the cover of the album.

The projector-camera system uses the appearance description to search for the object in the camera image. The first user places the object down on a table and it detects it has stopped moving. The camera detects the closed object on the table and calculates its location and orientation.

7.2.3 Projection and Manipulation

The object requests projection of the first album image (image 0 in Figure 7.7) on the album cover. The first user touches the right button. The user's finger is detected in the interactive area by the camera system, activating the button and changing the button state to 10. The Object receives the new button state and requests projection of the second album image (image 10 in Figure 7.7) on the album cover.

The first user opens the cover of the album, which appears to the object as a change in light level value. The object evaluates its 3D Model configuration states and changes to state 2 – Book Open, updating the projector-camera system with a new 3D Model of the open object and a new appearance model of the inside of the album. The updating of 3D Model geometry automatically removes the projection on the cover. The object requests projection of image 11 on the inside of the album. The first user passes the album to the second user, sat around the corner of the table. The projection follows the object movement, so it appears attached to the object. The second user browses through the images using the left and right buttons and then closes the object. The close action is again detected by the change in light levels and the 3D Model configuration state returned to 1- Book Closed. The projection returns to image 10. The first user picks up the album and exits the environment.

7.2.4 Discussion

The photograph album application demonstrates projection on the surface of the smart object. Users can interact directly with the projected buttons on the object to change the projected content. The object senses physical changes such as movement and manipulation of its articulated geometry, such as opening the book. Both factors have an impact on the detection process. In a vision-only approach any change in appearance or form of the object can cause detection failure. In our approach the object updates the projector-camera system with new appearance knowledge when it detects it has been opened, enabling tracking to continue uninterrupted. Hence, sensor information supports detection robustness.

In the demonstration implementation there are 3 albums and each album has 5 images for projection inside when the album is opened. However, as each image requires its own button state in the Photograph Album model, the number of albums and images is only limited by the storage requirements (of both Object Model and images) and the 16-bit integer used for representing individual button states (a total of 65,535 images).

This application demonstrated that a user can interact with an object while being detected, tracked and projected. However, several practical issues emerged from the finger detection method. It was found both the accuracy of pose calculation and jitter has a large effect on the finger detection method. For button activation our fingertip detection method requires a motion gradient towards then followed by away from the button location, combined with a matched semi-circular fingertip template within the button area. However, when extracting images of the active button detection area any jitter or incorrect pose calculation causes an offset area to be extracted. When used in background subtraction this image easily causes a motion gradient to be established. If the area extracted also contains either projection or the object's surface contains texture which appears rounded like a finger, then a false positive occurs. Similarly, if the image extracted is offset too far from the correct detection area it can cause a real finger interaction to be missed (a false negative).

In the photo album demonstrator we initially projected a rounded arrow into each of the button areas, but this was found to give too many false positives and the projection had to be changed to a different triangular arrow appearance. This has important implications for this interaction method in the architecture, as it suggests either this method is not suitable for use in conditions with high jitter or high chance of incorrect pose calculation (such as with moving objects), or a new more robust method for finger detection is required (one possibility would be the pattern oriented detectors proposed by Borkowski et al. [Borkowski, Crowley et al. 2006]).

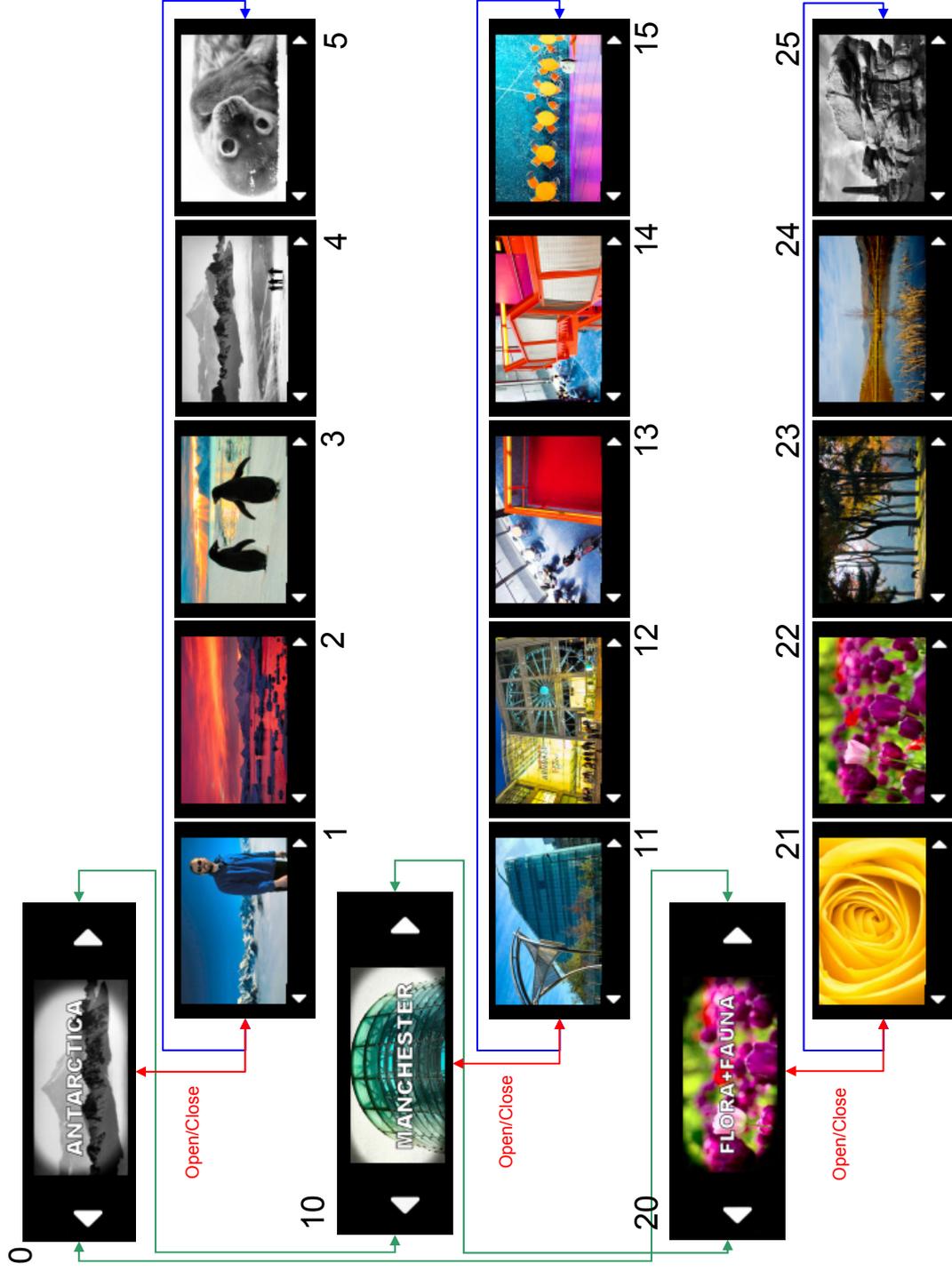


Figure 7.7 Photograph Album Content to Project

7.3 Smart Cooking

This demonstrator is designed to illustrate multiple objects augmented with displays from multiple projector-camera systems. Here we use a task based cooking scenario where smart objects cooperate in order to provide the user with context-dependant recipe instructions to boil an egg.

We motivate this demonstrator with an scenario similar to the MIT Smart Kitchen project CounterIntelligence [Bonanni, Lee et al. 2005], which distributed the recipe instructions into the environment, with an interactive projection of a recipe book onto the work-surface. The user followed the recipe and instructions would be projected at the correct time on the work-surface, using a smart-environment approach with cameras and sensors to detect interaction distributed in the kitchen.

In our approach we focus on projection directly on the objects, rather than projection at a central location in the environment. This is one of the main characteristics of the Cooperative Augmentation approach, as it allows revealing of hidden knowledge inside the object (such as its temperature) or direct feedback to user interaction with the object, where it belongs spatially in the real-world.

Hence, instead of taking a smart-environment approach we take a smart object approach, by making use of sensors embedded in the individual objects themselves (specifically a smart recipe book, smart egg-box, smart pan, smart salt, and smart stove). These sensors are used to infer the current context and project the next instruction step onto both the recipe book and on the objects themselves.

The smart recipe book contains a series of recipes, each with printed photos of how the meal looks. The user can move around the environment (with the book being tracked with the steerable projector) and select a recipe by turning the pages of the book (similar to the scenario described in section B.2). Each recipe states only the ingredients required and the average time it takes to cook. To select a recipe, the user touches a projected interactive “cook” button on the smart recipe book.

Instead of the smart environment approach, with a single object (the recipe book, or the environment) controlling the interaction, we break down the cooking task into sequential sub tasks and distribute the tasks to the smart objects in the environment. When a recipe is selected in the book, each object involved in the recipe is dynamically programmed with the sub-tasks by the recipe book. Task distribution and object programming can be performed automatically from the task description using the RuleCaster approach by Bischoff and Kortuem [Bischoff and Kortuem 2006].

7.3.1 Hard-Boiled Egg Recipe

The task we chose to illustrate the kitchen scenario was to hard-boil an egg. The recipe can be split into 3 parts: ingredients, equipment required and the method. As it is challenging to augment an individual egg due to its size, we instead augment the egg-box with a sensor node.

Ingredients

1. Eggs (inside egg-box)

2. Salt
3. Water

Equipment

1. Pan
2. Stove
3. Sink

Method

1. Add egg from box to saucepan
2. Add cold water from sink to pan, covering the egg
3. Place pan on stove
4. Add a pinch of salt to water
5. Turn on stove, to highest temperature setting
6. Wait until water is boiling.
7. Turn down stove to simmer for 7 minutes
8. Turn off stove
9. Empty hot water in sink

7.3.2 Object Model

In this demonstrator implementation we simulate the RuleCaster dynamic programming by developing Object Models for the chosen task and embedding them in each object by hand. The sequential sub-tasks of the recipe are considered as individual states of the overall recipe. These states are numbered and used to synchronise the recipe across the smart objects. Initially, the recipe book broadcasts recipe state value of 0. Objects assigned sub-tasks sense the recipe state on the network and begin their sub-task when the state variable is the correct value. Each object completing a sub-task increments, then re-broadcasts the state variable on the network. We develop a set of rules which form the application logic for each task. These are represented in the Object Models as the Content Models (Content Models are described in 7.2.1).

Note: The complete content models of the egg-box, pan, salt and stove objects can be seen in Figure B.4 to Figure B.8, grouped by recipe state variable and arranged sequentially in recipe state order 0-12. We present only excerpts from these figures suitable for understanding our demonstrator in the task procedure description below.

7.3.3 Task Procedure

We assume the egg-box, pan, salt and stove are all present in the environment with the recipe book.

The egg-box uses a light sensor to detect when it is open (high light level) and closed (low light level). The pan uses a force sensor to detect the weight of the contents, when it is placed on a surface (high force value) and when it is picked up (0 or very low force value). Additionally, the pan uses a temperature sensor to detect when the pan is at the correct temperature for cooking the egg. The salt uses a force sensor to detect when it is picked up. The stove uses a gas knob position sensor (simple variable resistor voltage value converted 0-100% with the Smart-It's A-to-D converter).

We use one steerable projector-camera system centrally mounted in the environment and one fixed projector-camera system with a view down onto the work-surface to detect the objects. When the user stands in front the work-surface they occlude the steerable projector, hence we use the fixed projector-camera system for detection and projection in this location. Objects moving between the two systems experience a short loss in tracking when moving between the two projector-camera systems, as the user typically occludes the steerable projector before the object enters the work-surface projector’s field of view.

Recipe State 0

When the initial state is broadcast by the Recipe Book, the egg-box requests a projection of “Add egg to pan”, whether it is open or closed, as shown in Figure 7.8.

We use the pan’s force sensor to sense when an egg is added to the pan. We assume eggs weigh over 30g (typically eggs weigh 35-75g). Consequently, the pan requests a projection of “Add egg to pan”, until the force detected increases over 30g. When this occurs the projection is removed and the new recipe state 1 is broadcast.

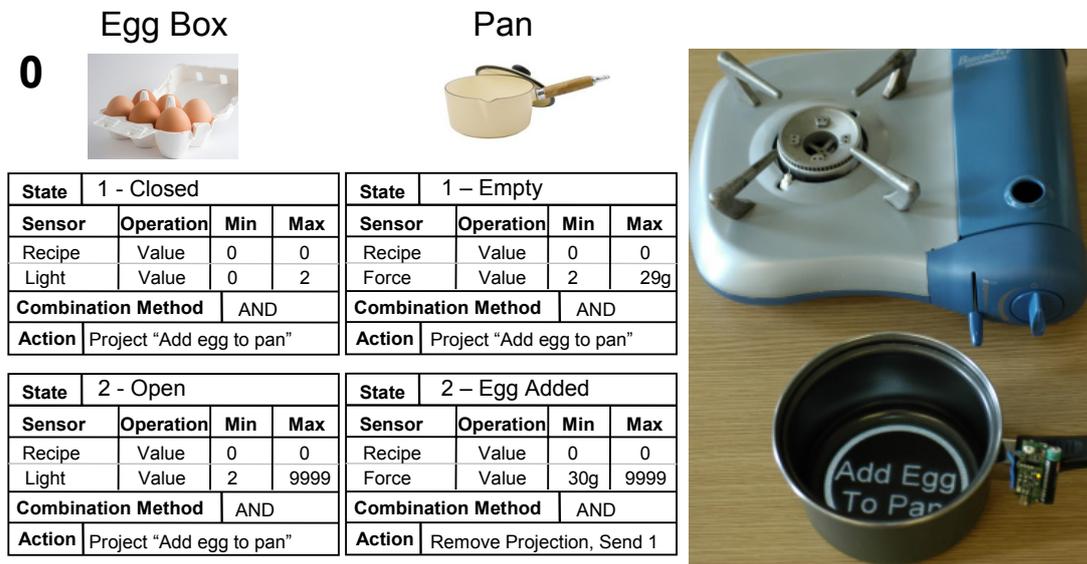


Figure 7.8 (left) Recipe State 0, (right) Add Egg Projection inside pan

Recipe State 1

The projection on the egg-box is removed automatically when state 1 is broadcast. The egg-box plays no further part in this scenario.

The pan now requests a projection asking the user to “Add cold water to the pan, to cover the egg”. When the force decreases, we infer the object has been picked up. When the pan detects the temperature has decreased significantly below ambient (20° is assumed as ambient) we infer the user has added cold water to the pan. The pan projects “Place pan on stove” and the new recipe state 2 is broadcast.

Recipe State 2

The pan monitors the force sensor and the distance to the stove. When it detects a high force value again and the distance to the smart stove object is less than 0.25m, it infers it must have been placed on the stove. The pan removes the projection and the new recipe state 3 is broadcast.

Recipe State 3

The pan requests the projection “Add a pinch of salt” throughout this state, while it is on the stove.

The salt object requests the projection “Add a pinch of salt” if it detects it is put down (static), distant from the pan. When the salt detects it has been picked up (low force value) and is within 0.25m of the pan the new recipe state 4 is broadcast.

Recipe State 4

The salt removes the projection when it detects it has been put back down and the new recipe state 5 is broadcast.

Recipe State 5

The “Add a pinch of salt” projection on the pan is automatically removed.

The stove object requests a projection “Turn On Stove, to 100% setting” if the stove setting is less than 100%. When the stove setting is turned to 100%, the projection is removed and the new recipe state 6 is broadcast.

Recipe State 6

The pan object projects the temperature from the temperature sensor and a “Wait until water boils” message as long as the pan is on the stove and the temperature is below 100°. When the temperature reaches 100° the new recipe state 7 is broadcast.

Recipe State 7

The stove object requests a projection “Turn Down Stove to Simmer (40% setting)” if the stove setting is outside the value range 31 to 49%. When the stove setting is within the value range 31 to 49% the projection is removed and the new recipe state 8 is broadcast.

8

Stove 

Pan 

<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>State</th> <td colspan="3">5 – Pan Too Cold</td> </tr> <tr> <th>Sensor</th> <th>Operation</th> <th>Min</th> <th>Max</th> </tr> </thead> <tbody> <tr> <td>Recipe</td> <td>Value</td> <td>8</td> <td>8</td> </tr> <tr> <td>Object Pan</td> <td>Temp</td> <td>0</td> <td>90°</td> </tr> <tr> <td colspan="2">Combination Method</td> <td colspan="2">AND</td> </tr> <tr> <td>Action</td> <td colspan="3">Project Pan too cold, adjust to simmer</td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>State</th> <td colspan="3">6 – Pan Too Hot</td> </tr> <tr> <th>Sensor</th> <th>Operation</th> <th>Min</th> <th>Max</th> </tr> </thead> <tbody> <tr> <td>Recipe</td> <td>Value</td> <td>8</td> <td>8</td> </tr> <tr> <td>Object Pan</td> <td>Temp</td> <td>99</td> <td>9999</td> </tr> <tr> <td colspan="2">Combination Method</td> <td colspan="2">AND</td> </tr> <tr> <td>Action</td> <td colspan="3">Project Pan too hot, adjust to simmer</td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>State</th> <td colspan="3">7 – Pan Correct Temp</td> </tr> <tr> <th>Sensor</th> <th>Operation</th> <th>Min</th> <th>Max</th> </tr> </thead> <tbody> <tr> <td>Recipe</td> <td>Value</td> <td>8</td> <td>8</td> </tr> <tr> <td>Object Pan</td> <td>Temp</td> <td>91</td> <td>98°</td> </tr> <tr> <td colspan="2">Combination Method</td> <td colspan="2">AND</td> </tr> <tr> <td>Action</td> <td colspan="3">Remove Projection</td> </tr> </tbody> </table>	State	5 – Pan Too Cold			Sensor	Operation	Min	Max	Recipe	Value	8	8	Object Pan	Temp	0	90°	Combination Method		AND		Action	Project Pan too cold, adjust to simmer			State	6 – Pan Too Hot			Sensor	Operation	Min	Max	Recipe	Value	8	8	Object Pan	Temp	99	9999	Combination Method		AND		Action	Project Pan too hot, adjust to simmer			State	7 – Pan Correct Temp			Sensor	Operation	Min	Max	Recipe	Value	8	8	Object Pan	Temp	91	98°	Combination Method		AND		Action	Remove Projection			<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>State</th> <td colspan="3">9 – Simmering 7 mins</td> </tr> <tr> <th>Sensor</th> <th>Operation</th> <th>Min</th> <th>Max</th> </tr> </thead> <tbody> <tr> <td>Recipe</td> <td>Value</td> <td>8</td> <td>8</td> </tr> <tr> <td>Force</td> <td>Value</td> <td>30g</td> <td>9999</td> </tr> <tr> <td>Object Stove</td> <td>Distance</td> <td>0</td> <td>0.25m</td> </tr> <tr> <td>Time</td> <td>Value</td> <td>0</td> <td>7min</td> </tr> <tr> <td colspan="2">Combination Method</td> <td colspan="2">AND</td> </tr> <tr> <td>Action</td> <td colspan="3">Project countdown timer</td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>State</th> <td colspan="3">10 – Simmering 7 mins</td> </tr> <tr> <th>Sensor</th> <th>Operation</th> <th>Min</th> <th>Max</th> </tr> </thead> <tbody> <tr> <td>Recipe</td> <td>Value</td> <td>8</td> <td>8</td> </tr> <tr> <td>Force</td> <td>Value</td> <td>30g</td> <td>9999</td> </tr> <tr> <td>Object Stove</td> <td>Distance</td> <td>0</td> <td>0.25m</td> </tr> <tr> <td>Time</td> <td>Value</td> <td>7min</td> <td>7min</td> </tr> <tr> <td colspan="2">Combination Method</td> <td colspan="2">AND</td> </tr> <tr> <td>Action</td> <td colspan="3">Remove Projection, Send 9</td> </tr> </tbody> </table>	State	9 – Simmering 7 mins			Sensor	Operation	Min	Max	Recipe	Value	8	8	Force	Value	30g	9999	Object Stove	Distance	0	0.25m	Time	Value	0	7min	Combination Method		AND		Action	Project countdown timer			State	10 – Simmering 7 mins			Sensor	Operation	Min	Max	Recipe	Value	8	8	Force	Value	30g	9999	Object Stove	Distance	0	0.25m	Time	Value	7min	7min	Combination Method		AND		Action	Remove Projection, Send 9		
State	5 – Pan Too Cold																																																																																																																																								
Sensor	Operation	Min	Max																																																																																																																																						
Recipe	Value	8	8																																																																																																																																						
Object Pan	Temp	0	90°																																																																																																																																						
Combination Method		AND																																																																																																																																							
Action	Project Pan too cold, adjust to simmer																																																																																																																																								
State	6 – Pan Too Hot																																																																																																																																								
Sensor	Operation	Min	Max																																																																																																																																						
Recipe	Value	8	8																																																																																																																																						
Object Pan	Temp	99	9999																																																																																																																																						
Combination Method		AND																																																																																																																																							
Action	Project Pan too hot, adjust to simmer																																																																																																																																								
State	7 – Pan Correct Temp																																																																																																																																								
Sensor	Operation	Min	Max																																																																																																																																						
Recipe	Value	8	8																																																																																																																																						
Object Pan	Temp	91	98°																																																																																																																																						
Combination Method		AND																																																																																																																																							
Action	Remove Projection																																																																																																																																								
State	9 – Simmering 7 mins																																																																																																																																								
Sensor	Operation	Min	Max																																																																																																																																						
Recipe	Value	8	8																																																																																																																																						
Force	Value	30g	9999																																																																																																																																						
Object Stove	Distance	0	0.25m																																																																																																																																						
Time	Value	0	7min																																																																																																																																						
Combination Method		AND																																																																																																																																							
Action	Project countdown timer																																																																																																																																								
State	10 – Simmering 7 mins																																																																																																																																								
Sensor	Operation	Min	Max																																																																																																																																						
Recipe	Value	8	8																																																																																																																																						
Force	Value	30g	9999																																																																																																																																						
Object Stove	Distance	0	0.25m																																																																																																																																						
Time	Value	7min	7min																																																																																																																																						
Combination Method		AND																																																																																																																																							
Action	Remove Projection, Send 9																																																																																																																																								

Figure 7.9 Recipe State 8

Recipe State 8

The stove regulates the pan temperature by requesting the temperature from the pan object. If the temperature value reduces below 91° the simmering will stop, so the stove requests a projection “Pan too cold – turn up stove to simmer”. If the temperature value

increases above 98° the pan is boiling, so the stove requests a projection “Pan too hot – turn down stove to simmer”. When the temperature is between 91 and 98° any projection is removed, as shown in Figure 7.9.

The pan object requests projection of a countdown timer for 7 minutes while the pan is simmering on the stove. The timer can take the form of a 7minute video of a countdown. When the timer ends the projection is removed and the new recipe state 9 is broadcast.

Recipe State 9

The stove object requests a projection “Turn Off stove” while the stove setting is above 0%. When the stove is turned to 0% (off) the projection is removed and the new recipe state 10 is broadcast.

Recipe State 10

The pan object requests a projection “Empty hot water into sink” while the object is static on the stove and not near the sink. When the pan detects it has been picked up, is over 0.25m from the stove and closer than 0.25m to the sink the new recipe state 11 is broadcast.

Recipe State 11

When the object detects it is picked up and the temperature has decrease significantly below the simmering temperature, it infers the water has been emptied. The pan requests a projection “Enjoy your meal!” and the new recipe state 12 is broadcast.

Recipe State 12

When the pan object is placed down on a surface the projection is removed. This completes the recipe.

7.3.4 Discussion

The CounterIntelligence project [Bonanni, Lee et al. 2005] used a centralised projection approach in their smart environment, where the projected recipe appeared at one location on the work-surface in the kitchen environment. This provides a single point of focus for the user, who always knows where the current recipe instruction is located. Similarly, we can mechanically align our projector to the fixed projection surface to eliminate any geometrical distortion. However, we can illustrate one of the downsides of this and the smart environment approach if we imagine a hot pan on the stove. We want to warn the user that the pan is hot (as shown in the CounterIntelligence project), but because the temperature sensor and projector is located over the stove/work-surface area we can only project the temperature onto the pan when the pan is on the stove. If we take the pan off the stove it is still hot, but we can no longer sense this or project the warning onto the pan.

In contrast, the approach taken in this thesis by the Cooperative Augmentation framework is to project directly onto the objects and use their sensing capability. One major benefit of our approach is that we obtain knowledge of the object’s geometry from the smart object. This is significant as it allows us to ignore surrounding surfaces and use projection whenever the object is visible to the projector, in any environment in which a projector-camera system is located - not just those with nearby surfaces suitable to projection. In the smart cooking scenario this approach allows us to provide visual feedback directly on the objects, where it belongs, whenever they are in view of a

projector-camera system. In this case a hot smart pan senses its own temperature continuously and can request a warning projection whenever it is hot enough to burn. This projection will appear fixed to the object as it is moved around the environment by using different projector-camera systems to detect, track and project on it.

7.3.4.1 Coping with Objects Unsuitable for Projection

A number of the objects in the smart cooking scenario are physically small, such as the eggs and the salt. As described in section 5.3, small objects are challenging to detect and track accurately. Similarly, small objects have little space for projection. One solution supported by our architecture is to project onto other nearby smart objects (as described in section 7.1.8).

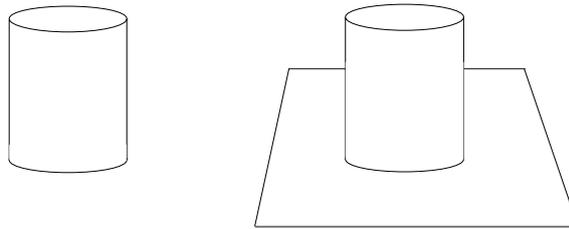


Figure 7.10 (left) Example salt 3D model, (right) 3D model with added base projection area

It is also possible to extend an object's 3D model by making assumptions. For example, we can extend the 3D model of the salt to include a large rectangular area next to, or centred at the base of the cylinder, as shown in Figure 7.10. This would allow projection around the cylinder of the salt itself, but assumes the salt is stood with its base on a flat surface and there are no other objects occluding the projection area. Similarly, the methods for geometry recovery discussed in section 2.3.6 could be used to dynamically model surfaces around the object to allow projection.

However, displays created next to the object have the drawback that the display is no longer seamlessly integrated with the object. Hence, the user may find the display projected on the surface of random objects in the environment. This breaks the association with the original smart object which is implicit when projecting on its surfaces, and could cause users to become confused when displays appear in an unexpected spatial location, as reported by Sukaviriya et al. [Sukaviriya, Podlaseck et al. 2003].

7.3.4.2 Implications of Dealing with Multiple Objects

The most challenging aspect of dealing with multiple objects in this scenario was their synchronisation, so that each object was aware of what to sense, which states it could change to, and what to project at any point in time. This has an important implication for our framework, as any scenarios with multiple objects and involving an aspect of time will face the same problem.

Here we introduced a relatively naïve solution to synchronisation by having each object broadcast a shared global variable, updating the sequential recipe state in all objects. However, this method can only be used in scenarios where the objects are known and fixed a-priori; it does not allow another object to be spontaneously incorporated in the recipe without a global re-programming.

There is great scope for failure in this demonstrator, as we consider the recipe as a series of states. Any failure to sense a particular state correctly, such as false negatives where an event is not detected (e.g. adding the egg to the pan), or false positives where an event is detected even though it didn't happen (e.g. the salt weight sensor detects it is picked up due to a low weight reading, but in reality it is just empty) will cause a failure in the recipe. The user may be able to proceed from a false positive, as the recipe should wait for the objects to enter the next state, however if an event is never sensed (false negative) due to incorrect Object Model sensor threshold values, incorrect object programming or a hardware failure, then the recipe will stall and the user will not be able to proceed. In practice, the current smart cooking demonstrator is not particularly robust, but still serves as a good concept illustration.

7.3.4.3 Time

Our architecture currently does not address the concept of time, hence in this scenario when the smart pan object has to wait until the egg is cooked we project a video which lasts for the required length. This method has a disadvantage in our scenario; due to our current implementation of video playing in the architecture it cannot be paused if the pan is taken off the stove (so that cooking stops) and then re-started from the same position when the pan is replaced on the stove. Instead, in the current implementation the video would be re-started from the beginning.

To explicitly support time a specific timer capability would have to be implemented either in the smart object, running in parallel with the Object Model processing, or on the network. The timer could be modelled to stop and start by event messages sent from the content model states and its state queried as a normal sensor in the framework.

7.3.4.4 Dynamic Object Programming of Multi-Purpose Objects

The physical form and external appearance of the majority of smart objects will typically remain constant over time. Similarly, the number and type of sensors physically embedded in the object will remain constant. However, the use of an object may change frequently. For example, the salt can be used in a variety of ways in cooking (added to water to increase boiling temperature, added to sauces, on fish and chips, etc...) or possibly outside the home to spread on the ground to stop ice forming on paths. Hence, we need a method to allow multi-purpose smart objects to be automatically included in different scenarios.

The cooperative augmentation framework provides a method for updating an object's Object Model dynamically, for example, in response to new appearance knowledge extracted by the detection process. However, we are not restricted to just updating appearance knowledge. Multi-purpose objects can be dynamically programmed by updating the Content Models in the Object Model. Each of the ingredients and pieces of equipment in the kitchen scenario are multi-purpose objects. Hence we use automatic task distribution and object programming directly from the task description using the RuleCaster approach by Bischoff and Kortuem [Bischoff and Kortuem 2006].

However, as this re-programs the whole content model, this approach only allows objects to be part of a single scenario at time. To allow an object to be truly shared dynamically among a number of uses would require a method of fusing content models that avoided conflicts, such as content models from two scenarios requesting different projections on the same surface at the same time.

7.4 Conclusion

In this chapter we presented three demonstration applications to evaluate the Cooperative Augmentation architecture implementation described in Chapter 6.

A feature shared by all three demonstrators was that the knowledge was distributed in the objects, not in the environment. Our Cooperative Augmentation approach uses smart objects rather than smart environments to achieve projected displays on object's surfaces. In the smart cooking scenario this meant all the objects cooperated to monitor the cooking process, rather than a single controlling object or the environment. The smart chemical container actively increased its knowledge by extracting more appearance information, demonstrating re-embedding of knowledge into the object.

Additionally, the smart chemical container objects supported direct interaction via our architecture, while simultaneously monitoring their workflow. For example, the projection was displayed when the user misplaced the object and stopped when the user corrected the container's position. Here the object actively controls the projection based on the states pre-defined in its Object Model.

In contrast, the smart photograph album changed the photograph displayed when user interaction was detected via finger sensing, demonstrating another interaction modality of our architecture.

The use of sensing was shown to greatly benefit the detection process. In the smart chemical container demonstrator, movement sensing constrained the detection process when detecting moving objects and discriminated between objects with identical appearances. In the smart photograph album sensing detected user manipulation of the object geometry and allowed the object to update the projector-camera system with new appearance and geometry knowledge for uninterrupted tracking.

We identified limitations in projection due to low resolution projection achievable on the smart chemical container. Hence, we suggest objects should incorporate an adaptation mechanism to adjust what they project based on the achievable display resolution. For example, if only a low resolution is available then detailed text could be replaced with successively more abstract information with larger font size, or a pictogram or symbol.

From the smart cooking demonstration we also found that some objects that are very difficult to augment with embedded sensing and computation either due to their size, shape, their use or their nature. For example, the eggs in the smart cooking demonstrator are disposable objects, hence it would not likely be economically feasible to augment every egg with embedded sensing and computation.

Similarly, we found that some object surfaces are not suitable for projection (such as very small, transparent or reflective surfaces). For example, the eggs in the smart cooking demonstrator cannot easily support projection of complex information due to their size. In the case of reflective surfaces a coping method has been proposed using multiple projectors [Park, Lee et al. 2005], however this requires known surface geometry and explicitly tracked users to calculate the projection required to minimise reflection.

These last two findings are very significant for our approach, as they suggest there is likely to be a class of everyday real-world objects which cannot be used with Cooperative Augmentation.

Chapter 8 Conclusion

The primary objective of this thesis has been to develop a means for smart objects to cooperate with projector-camera systems to achieve a display capability without adding dedicated tracking hardware to every object or changing the natural appearance, purpose and function of the object.

Displays allow any smart object to deliver visual feedback to users from implicit and explicit interaction with information represented or sensed by the physical object. This feedback has the potential to provide a balanced interface, as we support physical objects as both input and output medium simultaneously. This contributes to the central vision of Ubiquitous Computing by enabling users to address tasks in physical space with direct manipulation and have feedback on the objects themselves, where it belongs in the real world.

8.1 Contributions and Conclusions

The core contribution of this thesis is the development of a new approach called Cooperative Augmentation which enables smart objects to cooperate with projector-camera systems to achieve a display capability. This framework formalises a mechanism for smart objects and multiple distributed projector-camera systems to cooperate to solve the object's output problem and achieve interactive projected displays on their surfaces.

The Cooperative Augmentation framework describes the role of the smart object, projector-camera system and the cooperative process. Specifically, the approach embeds the knowledge required for detection and projection in the smart object. This allows us to assume projector-camera systems offer generic projection services for all smart objects.

To program the smart object we develop a state-machine programming method. The smart objects were modelled to give the projector-camera system enough information about their appearance, form and capabilities to be able to detect and track them visually. Object sensors are modelled to allow the projector-camera system to make use of movement sensors for constraining the detection process and to make use of other sensors for interaction detection. In Chapter 5 we show that use of this sensing in detection provides a significant improvement in detection performance.

While low-level, our programming method is lightweight, easily conceptualised and requires little information beyond which sensors we want to use, how to combine them and what sensor ranges we want to assign to a particular state. This allows smart objects to change projections in response to sensor events either sensed directly by the object, or remotely (for example, the movement of another associated object).

An architecture for the Cooperative Augmentation framework was implemented and experimentally evaluated in Chapter 6 to validate the concept. We present three demonstration applications using the architecture in Chapter 7, developed to demonstrate three different aspects of the architecture. The applications showed that smart objects with different sizes, shapes and appearances could be successfully detected and tracked by the projector camera system. Additionally, they demonstrated that using our framework the different geometries and surface colours could be corrected so the projection appeared undistorted and visible to an observer. Hence, we conclude that we achieved our original goal of enabling interactive projected displays on smart objects.

A central problem to achieving displays on smart objects is their detection and tracking. We investigated vision-based object detection by performing the two experimental studies presented in Chapter 4 and Chapter 5. We present seven insights from these chapters:

1. When investigating natural appearance vision-based detection algorithms we found that the use of scale and rotation invariant algorithms provides performance benefits over single scale and rotation-variant algorithms, without loss of discrimination. Hence we use them in our example framework implementation in Chapter 6.
2. We also learned that detection performance varies when the algorithm is trained at different distances and found the best training distances for each algorithm based on our object appearance library dataset. This is important, as this knowledge allows us both to help determine the best algorithm to use when detecting the object at runtime, and informs us of the best distances to extract extra appearance knowledge in the learning process for maximising detection performance.
3. When investigating rotation we found that an object's appearance changes when it is rotated in 3D (i.e. not in the camera plane), but this change can be much greater than with scaling, as whole surfaces can appear or disappear. Hence, to robustly detect an object in any 3D pose we found that we need to train our detection algorithms with multiple viewpoints.
4. The results of the cooperative detection experiment in Chapter 5 show a very clear performance gain for all four natural appearance detection cues when we added movement sensing in the object, indicating the combination of different sensing modalities is important in detection. The use of movement sensing constrained the search space, which translates to increased robustness to clutter and distractions in the real-world environment, and helped to discriminate between objects with identical appearances. The improvement was seen for all algorithms, suggesting this can be generalised to other detection algorithms.
5. We found that with the use of movement sensing, simple algorithms achieve similar or better detection performance to complex detection algorithms and that the runtime of the majority of algorithms was reduced. This has important implications as it means performance can be maintained or improved while reducing the amount of processing power required for detection, or that overall detection speed can be increased.
6. The results of the cooperative detection experiment confirms that different objects require different appearance representations and detection methods, validating our proposed multi-cue approach.

7. The use of multiple cues improved detection performance for all objects; however, little performance improvement was seen when using more than the best 2 cues. The percentage improvement in detection performance was also generally greater when using movement sensing.

These findings indicate that any environment using vision detection would greatly benefit from the fact that smart objects cooperate, by both informing the environment about object-specific knowledge (such as the object's appearance) and embedded sensor readings.

Appendix A provides an additional contribution by analysing the benefits and drawbacks to different designs of steerable projector and illustrating the construction of both a moving-head and moving-mirror prototype. The two prototypes were built from components, which involved both the choice of the appropriate technology and the subsequent integration of the technology with the computer system. The objective was to develop a system enabling detection and tracking of mobile objects and interactive display of graphical user interfaces anywhere in the environment. Of the available technologies presented in Chapter 2, we consider computer vision detection combined with steerable projection best able to satisfy this objective.

While our steerable projector-camera systems have similar abilities to related projects with steerable-projectors [Pinhanez 2001; Borkowski, Riff et al. 2003; Butz, Schneider et al. 2004; Ehnes, Hirota et al. 2004], the equipment design is original. Specifically, for our moving head steerable projector we use a single arm yoke for pan and tilt, rather than twin-arms, allowing easier calibration of video projectors which have horizontally offset lenses (i.e. the majority), as the projector can be attached so the projector's centre of projection is close to the centre of rotation of the pan and tilt unit.

8.2 Benefits of our approach

With the Cooperative Augmentation approach we aim to make life easier for both the user and the smart object. By displaying information on objects, where it belongs in the real world, we can make use of human spatial memory to aid interaction [Butz and Krüger 2003]. Similarly, interfaces tightly coupled to an artefact's surfaces can allow a direct mapping for interaction, which may reduce the level of indirection and hence the cognitive load. For example, the orientation of an artefact could be used as a direct input method while the projected display provides the user with instant feedback to manipulation of the object itself, such as the change of a highlighted option.

With projected displays users are no longer constrained to single static location displaying information about smart objects. Instead, objects are tracked and the displays follow the objects. Steerable and mobile projector-camera systems further enhance and encourage the mobility of the objects, so users no longer have to sit in a specific location to read an interactive projected book. The use of projected displays also allows multiple people to see the display and interact simultaneously.

Objects can be more descriptive themselves, either directly or indirectly. For example, they can directly label themselves with their names in a foreign language to help students learn [Intille, Lee et al. 2003], or project assembly instructions directly onto pieces of furniture to decrease difficulty and reduce errors in the assembly task easier for novice users, as described in section B.3.

This approach provides a possible solution to one of the typical problems for users in Ubiquitous Computing environments – knowing which objects are smart and can be interacted with. As objects can label themselves, projected displays support easy discovery for users. One example application would be a searchlight application, which would detect all smart objects in the environment and reveal information about their status or supported interaction methods on their surfaces.

In addition to making inherent object properties explicit (where the invisible becomes visible) we can use the displays to reveal or visualise state changes in the object. For example, a smart object configuration and debugging display that allows programmers real-time modification of the smart object's state configurations.

Similarly, displays can be used for inspection, to reveal hidden contents or relationships. For example, two objects without a built-in location system can be associated and display content linked to a particular state where they are in proximity. These displays can also visualise information stored in the object such as a record of use, for example, how a wine bottle was stored while maturing.

Our approach also has potential benefits for employers and employees as it could replace steps in work processes. For example, it has been demonstrated that smart objects can embed rules from the large rulebooks for safe handling of chemical containers [Strohbach, Gellersen et al. 2004]. Employees are currently responsible for safe handling, but may not remember all the rules in detail. By using smart containers and projection services, there is the potential to either remove some of the responsibility from the employees or make the interaction faster and simpler for the employee. Safety critical storage locations could be monitored by the chemical containers themselves and warn the employees by visual notification when a container is stored outside the approved storage area or near to reactive chemicals. In this case the job of the employee is made much easier if the container can identify itself to the employee visually and guide the employee to the correct location.

8.3 Limitations

While our work provides a proof of concept for the Cooperative Augmentation framework, there are a number of technical limitations which would make its direct deployment in the real-world infeasible. We aim to identify some of these limitations in this section and suggest possible solutions.

Robust vision-based object detection in the real-world is a hard problem and the visual detection system we implemented is not a perfect detection system. Consequently, many objects achieve less than 100% detection performance with our natural appearance methods due to problems with scale, rotation, defocus blur, fast motion blur, partial or total occlusion, lighting and shadows or distracting objects. Practically, this means an object may not be immediately detected on entry to an environment (if at all). However, as we demonstrate with the cooperative detection experiment in Chapter 5, there are ways to improve the detection performance without writing new improved detection algorithms, by making use of embedded movement sensing in the object.

However, there are some limitations to an approach only involving movement. To detect static objects, the only way we can use movement sensing is to exclude moving areas in the image from the detection process. While this confers some similarities to

the scenario where there is a moving object and static background, the detection performance will vary between performance without sensing and with sensing, depending on the size of the moving area in the image.

In this case, greater knowledge of the object's context from other sensors may help us detect the object, for example, using an embedded electronic compass or any other available location system. Similarly, if the object has external light sensors a structured light approach would also be possible, as discussed in section 2.7.2.

The smart chemical container demonstration application in Chapter 7 raised another limitation of our system, which was the resolution of the projection on the chemical containers. The projector we used was an XGA (1024x768 pixel) projector, and it created a relatively low resolution 200x130 pixel display with 1mm² pixels at a distance of 3m when set to mid-zoom. To increase this resolution either a higher resolution projector can be purchased (High Definition 1920x1080 pixel projectors are now common), or the projector zoom can be used dynamically to zoom-in, with the trade-off of reducing the field of view

From the smart cooking demonstration application we found that some objects are very difficult to augment with embedded sensing and computation either due to their size, shape, their use or their nature. Similarly, we found that some object surfaces are not suitable for projection (such as very small, transparent or reflective surfaces). These two findings illustrate a significant limitation of our approach, as they suggest there is likely to be a class of everyday real-world objects which cannot be used with Cooperative Augmentation.

8.4 Future Work

The three main challenges in our framework implementation are robust visual detection of smart objects, correcting the projection for non-planar geometry and non-ideal surface colour, and keeping the projection synchronised when the object is moved or manipulated. These are the areas where improvements to the implementation would bring the most benefit.

For detection specifically, more research is required on how different combinations of training knowledge and sensing change the detection performance, which computer vision algorithms are best suited to detecting the objects and to characterise what impact different movement sensor types have on robustness of detection.

The question is also how to make it more robust? We have seen that movement sensors embedded in smart objects can provide a significant increase in performance, so perhaps this can be generalised to other types of sensors. In this case the question would be which sensors, and how many more sensors would help? As section 2.7 illustrated, it has been shown that both inertial and light sensors are valuable in the detection and pose-calculation process, but there should be a comprehensive study of which sensors add the most value and which sensors could be best combined. For example, if the object has embedded inclinometer or 3D accelerometers, we know this can help in pose calculation, but only if we know our camera orientation relative to gravity vector (as discussed in section 5.3.4).

The detection system in the framework implementation can optionally make use of the graphics card GPU for speeding up complex image processing tasks. Typically the performance increase of 10-15% is seen over the CPU algorithms. However, the CPU-GPU transfer time is currently a major bottleneck, especially for small amounts of data

such as results to reduction operations or control data for the image processing algorithms on the GPU. Ideally, the solution is to move as much processing as possible to the GPU to optimise performance, while only performing initial upload of a camera image and download of the final detection result. However, only relatively few of the complex computer vision algorithms we use have been transferred to the GPU [Strzodka, Ihrke et al. 2003; Montemayor, Pantrigo et al. 2004; Cabido, Montemayor et al. 2005; Fung and Mann 2005; Klein and Murray 2006; Sinha, Frahm et al. 2006; Leiva, Sanz et al. 2007] and many must still be written or further optimised to best make use of the parallel stream processing paradigm.

Both the detection approach and projective texturing geometric distortion correction method we use rely on a known 3D model of the object. While it is not a stretch to assume that ‘newly purchased’ smart objects brought into the environment would automatically have knowledge of their appearance and form, it does not address the problem of smart object prototyping or of legacy objects which we may possess and make smart ourselves by adding computation. These objects have no 3D or appearance models. While one possible solution would be to purchase commercial 3D object scanning equipment, this is an expensive option. In contrast, by using computer vision two approaches are readily available – firstly by using structured light with a projector-camera system, for example, the approach proposed by Borkowski et al. to detect planar surfaces in the environment with a steerable projector [Borkowski, Riff et al. 2003], or the approaches discussed in section 2.3.6. Secondly by using structure-from-motion techniques, either with direct affine-invariant local feature modelling [Rothganger, Lazebnik et al. 2006], or by using a SLAM approach [Davison and Murray 2002; Davison 2003; Chekhlov, Gee et al. 2007; Klein and Murray 2007]. Here we can imagine a user would only need to hold the object up to the camera and rotate it for a 3D and appearance model to be dynamically created.

Open questions also remain in the area concerning location of projections on an object. Specifically, if an object does not care about the exact projection location, or would always like the most visible, readable and useable projection, this would involve the framework automatically changing the location of the projection based on object visibility or orientation. The challenge here is how to determine the best strategy to ensure the most visible location on an object’s surfaces for the user is used? We would likely either need to track the user, or make assumptions about the user’s location.

We address scalability in our architecture by using a concept of an “environment” which is equivalent to a defined group of projectors and cameras contained in a constrained spatial area, such as a room. Similarly, for steerable projector-camera systems we introduce a concept of “system focus” to determine which object to track when multiple objects are present. However, it is still unclear how scalable these approaches are and how many objects can be seamlessly detected, tracked and augmented as they travel through space.

Finally, we have achieved a realisation of the Cooperative Augmentation concept that is flexible and efficient enough that user studies to evaluate the user experience become feasible. We believe the benefit of our approach can be seen best when directly comparing traditional AR displays (e.g. traditional monitors, HMD, handheld or mobile computers) against projected displays on the surfaces of objects.

Bibliography

- Aixut, T., Y. L. d. Meneses, et al. (2003). Constraining deformable templates for shape recognition. 6th Conference on Quality Control by Artificial Vision (QCAV).
- Antifakos, S., F. Michahelles, et al. (2002). Proactive Instructions for Furniture Assembly. Ubiquitous Computing (UbiComp 2002). Göteborg, Sweden, Springer Verlag.
- Antifakos, S., F. Michahelles, et al. (2004). Towards Situation-Aware Affordances: An Experimental Study. Pervasive Computing, Vienna, Austria, LNCS.
- Armstrong, M. and A. Zisserman (1995). Robust object tracking. Asian Conference on Computer Vision.
- Aron, M., G. Simon, et al. (2004). Handling Uncertain Sensor Data in Vision-Based Camera Tracking. Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society.
- Ashdown, M., M. Flagg, et al. (2004). A Flexible Projector-Camera System for Multi-Planar Displays. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 2004, Washington D. C., IEEE.
- Ashdown, M. and Y. Sato (2005). Steerable Projector Calibration. Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops - Volume 03, IEEE Computer Society.
- Azuma, R., Y. Baillet, et al. (2001). "Recent Advances in Augmented Reality." IEEE Computer Graphics and Applications 21(6): 34-47.
- Azuma, R. T. (1997). "A Survey of Augmented Reality." Presence: Teleoperators and Virtual Environments 6(4): 355-385.
- Bandyopadhyay, D., R. Raskar, et al. (2001). Dynamic Shader Lamps: Painting on Movable Objects. Proceedings of the IEEE and ACM International Symposium on Augmented Reality (ISAR'01), IEEE Computer Society.
- Bartelmus, C. (2007). "Linux Infrared Remote Control Webpage." Retrieved October, 2005, from <http://lirc.org/>.
- Beigl, M. and H. Gellersen (2003). Smart-Its: An Embedded Platform for Smart Objects. Smart Objects Conference (sOc). Grenoble, France
- Belongie, S., J. Malik, et al. (2001). Matching Shapes. International Conference on Computer Vision (ICCV).
- Belongie, S., J. Malik, et al. (2002). "Shape Matching and Object Recognition Using Shape Contexts." IEEE Transactions on Pattern Analysis and Machine Intelligence 24(4): 509-522.

- Bhasker, E. S., P. Sinha, et al. (2006). "Asynchronous Distributed Calibration for Scalable and Reconfigurable Multi-Projector Displays." IEEE Transactions on Visualization and Computer Graphics 12(5): 1101-1108.
- Billinghurst, M., H. Kato, et al. (2001). "The MagicBook, a transitional AR interface." Computers & Graphics 25: 745-753.
- Bimber, O., F. Coriand, et al. (2005). Superimposing pictorial artwork with projected imagery. ACM SIGGRAPH 2005 Courses. Los Angeles, California, ACM.
- Bimber, O. and A. Emmerling (2006). "Multi-Focal Projection: A Multi-Projector Technique for Increasing Focal Depth." IEEE Transactions on Visualization and Computer Graphics (TVCG).
- Bimber, O., A. Emmerling, et al. (2005). Embedded entertainment with smart projectors. IEEE Computer, 38: 56-63.
- Bimber, O. and R. Raskar (2005). Spatial Augmented Reality: Merging Real and Virtual Worlds, A. K. Peters, Ltd.
- Bischoff, U. and G. Kortuem (2006). RuleCaster: A Macroprogramming System for Sensor Networks. Proceedings OOPSLA Workshop on Building Software for Sensor Networks, Portland, Oregon, USA.
- Block, F., A. Schmidt, et al. (2004). Towards a Playful User Interface for Home Entertainment Systems. EUSAI 2004, Springer LNCS.
- Bonanni, L., C. H. Lee, et al. (2005). Counter Intelligence: Augmented Reality Kitchen. Extended Abstracts of Computer Human Interaction (CHI) 2005, Portland, OR.
- Borkowski, S. (2006). Steerable Interfaces for Interactive Environments. laboratoire GRAVIR – IMAG. Grenoble, INRIA Rhone-Alpes. PhD.
- Borkowski, S., J. L. Crowley, et al. (2006). User-Centric Design of a Vision System for Interactive Applications. Proceedings of the Fourth IEEE International Conference on Computer Vision Systems, IEEE Computer Society.
- Borkowski, S., J. Letessier, et al. (2004). Spatial Control of Interactive Surfaces in an Augmented Environment. 9th IFIP Working Conference on Engineering for Human-Computer Interaction (EHCI'04). Hamburg, Germany, Springer Berlin / Heidelberg. Volume 3425/2005: 228-244.
- Borkowski, S., O. Riff, et al. (2003). Projecting rectified images in an augmented environment. ProCams Workshop. International Conference on Computer Vision, ICCV 2003 Nice, France, IEEE Computer Society Press.
- Bourgeois, S., H. Martinsson, et al. (2005). A Practical Guide to Marker Based and Hybrid Visual Registration for AR Industrial Applications Lecture Notes in Computer Science 3691, Computer Analysis of Images and Patterns, Springer Berlin / Heidelberg. 3691/2005: 669-676.
- Bradski, G. R. (1998). "Computer Vision Face Tracking For Use in a Perceptual User Interface." Intel Technology Journal Q2.
- Brasnett, P., L. Mihaylova, et al. (2005). Particle Filtering with Multiple Cues for Object Tracking in Video Sequences. SPIE's 17th Annual Symp. on Electronic Imaging, Science and Technology, San Jose California, USA.
- Brooks, F. P. (1999). What's Real About Virtual Reality? IEEE Computer Graphics and Applications IEEE Computer Society Press. 19: 16-27.
- Brown, M. and D. G. Lowe (2002). Invariant features from interest point groups. British Machine Vision Conference (BMVC 2002). Cardiff, Wales.
- Brown, M., A. Majumder, et al. (2005). "Camera-Based Calibration Techniques for Seamless Multiprojector Displays." IEEE Transactions on Visualization and Computer Graphics 11(2): 193-206.

- Butz, A., M. Groß, et al. (2004). TUISTER: a tangible UI for hierarchical structures. Proceedings of the 9th international conference on Intelligent user interfaces. Funchal, Madeira, Portugal, ACM.
- Butz, A. and A. Krüger (2003). A Generalized Peephole Metaphor for Augmented Reality and Instrumented Environments. The International Workshop on Software Technology for Augmented Reality Systems (STARS), Tokyo, Japan.
- Butz, A., M. Schneider, et al. (2004). SearchLight - A Lightweight Search Function for Pervasive Environments. Pervasive2004. Vienna, Austria, Springer LNCS.
- Buxton, W. (1990). A Three-State Model of Graphical Input. Human-Computer Interaction - INTERACT '90. D. Diaper and e. al. Amsterdam, Elsevier Science Publishers B.V. (North-Holland): 449-456.
- Cabido, R., A. S. Montemayor, et al. (2005). Hardware-Accelerated Template Matching. Pattern Recognition and Image Analysis, Springer LNCS. 3522/2005: 691-698.
- Canny, J. (1986). "A Computational Approach To Edge Detection." IEEE Transactions on Pattern Analysis and Machine Intelligence(8): 679-714.
- Cao, X. and R. Balakrishnan (2006). Interacting with dynamically defined information spaces using a handheld projector and a pen. Proceedings of the 19th annual ACM symposium on User interface software and technology. Montreux, Switzerland, ACM.
- Cao, X., C. Forlines, et al. (2007). Multi-user interaction using handheld projectors. Proceedings of the 20th annual ACM symposium on User interface software and technology. Newport, Rhode Island, USA, ACM.
- Chandraker, M. K., C. Stock, et al. (2003). Real-Time Camera Pose in a Room. Computer Vision Systems: 98-110.
- Chekhlov, D., A. Gee, et al. (2007). Ninja on a Plane: Automatic Discovery of Physical Planes for Augmented Reality Using Visual SLAM. International Symposium on Mixed and Augmented Reality (ISMAR07). Nara, Japan.
- Chen, H., R. Sukthankar, et al. (2002). Scalable alignment of large-format multi-projector displays using camera homography trees. Proceedings of the conference on Visualization '02. Boston, Massachusetts, IEEE Computer Society.
- Cheng, K. and K. Pulo (2003). Direct Interaction with Large-Scale Display Systems using Infrared Laser Tracking Devices. Conferences in Research and Practice in Information Technology Series (CRPIT2003).
- Comaniciu, D. and P. Meer (2002). "Mean Shift: A Robust Approach Toward Feature Space Analysis." IEEE Transactions on Pattern Analysis and Machine Intelligence 24(5): 603-619.
- Dao, V. N., K. Hosoi, et al. (2007). A semi-automatic realtime calibration technique for a handheld projector. Proceedings of the 2007 ACM symposium on Virtual reality software and technology. Newport Beach, California, ACM.
- David, P. and D. DeMenthon (2005). Object Recognition in High Clutter Images Using Line Features. Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2, IEEE Computer Society.
- David, P., D. DeMenthon, et al. (2002). SoftPOSIT: Simultaneous Pose and Correspondence Determination. Proceedings of the 7th European Conference on Computer Vision-Part III, Springer-Verlag.
- David, P., D. DeMenthon, et al. (2003). Simultaneous pose and correspondence determination using line features. Computer Vision and Pattern Recognition (CVPR), IEEE.

- Davison, A. J. (2003). Real-Time Simultaneous Localisation and Mapping with a Single Camera International Conference on Computer Vision (ICCV 2003). Nice, France, IEEE.
- Davison, A. J. and D. W. Murray (2002). "Real-Time Simultaneous Localisation and Mapping with a Single Camera." IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7): 865-880.
- Decker, C., M. Beigl, et al. (2004). eSeal - a system for enhanced electronic assertion of authenticity and integrity of sealed items. In Proceedings of the Pervasive Computing, volume 3001 of Lecture Notes in Computer Science (LNCS), Springer Verlag LNCS.
- Decker, C., A. Krohn, et al. (2005). The Particle Computer System. In the proceedings of the ACM/IEEE Fourth International Conference on Information Processing in Sensor Networks (IPSN05). Los Angeles.
- DeMenthon, D. and L.S. Davis (1995). "Model-Based Object Pose in 25 Lines of Code." International Journal of Computer Vision 15: 123-141.
- Deriche, R. and O. Faugeras (1990). "Tracking Line segments." Image and Vision Computing 8(4): 261-270.
- Dietz, P. and D. Leigh (2001). DiamondTouch: a multi-user touch technology. Proceedings of the 14th annual ACM symposium on User interface software and technology. Orlando, Florida, ACM.
- DisappearingComputer. (2002). "European IST. The Disappearing Computer Initiative." Retrieved 10th October, 2007, from <http://www.disappearing-computer.net/>.
- Drummond, T. and R. Cipolla (2002). "Real-time visual tracking of complex structures." IEEE Transactions on Pattern Analysis and Machine Intelligence 27: 932-946.
- Ehnes, J. and M. Hirose (2006). Finding the Perfect Projection System – Human Perception of Projection Quality Depending on Distance and Projection Angle. Embedded and Ubiquitous Computing.
- Ehnes, J., K. Hirota, et al. (2004). Projected Augmentation - Augmented Reality using Rotatable Video Projectors. Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04) - Volume 00, IEEE Computer Society.
- Ehnes, J., K. Hirota, et al. (2005). Projected Augmentation II: A Scalable Architecture for Multi Projector Based AR-Systems Based on 'Projected Applications'. Proceedings of the Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society.
- F.Mokhtarian (1995). "Silhouette based Isolated Object Recognition through Curvature Scale Space." IEEE Trans. on Pattern Analysis and Machine Intelligence 17(5): 539-544.
- F.Mokhtarian and S. Abbasi (2005). "Robust automatic selection of optimal views in multi-view free-form object recognition." Pattern Recognition 38(7): 1021-1031.
- F.Mokhtarian, N. Khalili, et al. (2001). "Multi-scale free-form 3D object recognition using 3D models " Image and Vision Computing 19(5): 271-281.
- Felzenszwalb, P. F. (2005). "Representation and Detection of Deformable Shapes." IEEE Trans. Pattern Anal. Mach. Intell. 27(2): 208-220.
- Ferrari, V., T. Tuytelaars, et al. (2001). Makerless augmented reality with a real-time affine region tracker. IEEE and ACM ISAR.
- Fiala, M. (2005). "ARTag Webpage." Retrieved 03/03/05, from <http://www.cv.iit.nrc.ca/research/ar/artag/>.

- Fiala, M. (2005). ARTag, a Fiducial Marker System Using Digital Techniques. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) IEEE Computer Society.
- Fischler, M. A. and R. C. Bolles (1981). "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." Commun. ACM 24(6): 381-395.
- Flagg, M., J. W. Summet, et al. (2005). Improving the Speed of Virtual Rear Projection: A GPU-Centric Architecture. IEEE International Workshop on Projector-Camera Systems (PROCAMS 2005) held in conjunction with IEEE International Conference on Computer Vision & Pattern Recognition (CVPR 2005). San Diego, California, USA.
- Foxlin, E. and L. Naimark (2003). VIS-Tracker: A Wearable Vision-Inertial Self-Tracker. Proceedings of the IEEE Virtual Reality 2003, IEEE Computer Society.
- Fujii, K., M. D. Grossberg, et al. (2005). A Projector-Camera System with Real-Time Photometric Adaptation for Dynamic Environments. Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01, IEEE Computer Society.
- Fung, J. and S. Mann (2005). OpenVIDIA: parallel GPU computer vision. Proceedings of the 13th annual ACM international conference on Multimedia. Hilton, Singapore, ACM.
- Gee, A. P. and W. W. Mayol-Cuevas (2006). Real-Time Model-Based SLAM Using Line Segments. Advances in Visual Computing, Second International Symposium (ISVC 2006), Lake Tahoe, NV, USA, Springer LNCS.
- Gellersen, H.-W., M. Beigl, et al. (1999). The MediaCup: Awareness Technology Embedded in a Everyday Object. Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing. Karlsruhe, Germany, Springer-Verlag.
- Genc, Y., S. Riedel, et al. (2002). Markerless Tracking for AR: A Learning-Based Approach. International Symposium in Mixed and Augmented Reality (ISMAR 2002).
- Geoffrion, J. R. (2005). "Understanding Digital Camera Resolution " Retrieved 10th March, 2005, from <http://www.luminous-landscape.com/tutorials/understanding-series/res-demyst.shtml>.
- Giebel, J., D. M. Gavrilu, et al. (2004). "A Bayesian Framework for Multi-cue 3D Object Tracking " Lecture Notes in Computer Science Computer Vision - ECCV2004 3024/2004: 241-252.
- Gold, S., A. Rangarajan, et al. (1998). "New Algorithms for 2D and 3D Point Matching: Pose Estimation and Correspondence." Pattern Recognition 31(8): 1019-1031.
- Gordon, I. and D. G. Lowe (2004). Scene modelling, recognition and tracking with invariant image features. International Symposium on Mixed and Augmented Reality (ISMAR2004), Arlington, VA, USA.
- Gordon, I. and D. G. Lowe (2006). What and where: 3D object recognition with accurate pose. Toward Category-Level Object Recognition. J. Ponce, M. Hebert, C. Schmid and A. Zisserman, Springer-Verlag: 67-82.
- Grossberg, M. D., H. Peri, et al. (2004). "Making one object look like another: Controlling appearance using a projector-camera system." IEEE Computer Vision and Pattern Recognition 1: 452-459.
- Grundhöfer, A. and O. Bimber (2008). "Real-Time Adaptive Radiometric Compensation." IEEE Transactions on Visualization & Computer Graphics (TVCG) 14(1): 97-108.

- Grundhöfer, A., M. Seeger, et al. (2007). Dynamic Adaptation of Projected Imperceptible Codes. IEEE International Symposium on Mixed and Augmented Reality (ISMAR'07), Nara, Japan.
- Han, J. Y. (2005). Low-cost multi-touch sensing through frustrated total internal reflection. Proceedings of the 18th annual ACM symposium on User interface software and technology. Seattle, WA, USA, ACM.
- Harris, C. (1993). Tracking with rigid models. Active vision, MIT Press: 59-73.
- Harris, C. and M. Stephens (1988). A combined corner and edge detector. Proceedings of the 4th Alvey Vision Conference:.
- Hartley, R. and A. Zisserman (2003). Multiple View Geometry in Computer Vision, Cambridge University Press.
- Hirose, R. and H. Saito (2005). A Vision-Based AR Registration Method Utilizing Edges and Vertices of 3D Model. ICAT 2005. Christchurch, New Zealand.
- Holleis, P. (2005). "EiToolkit." Retrieved 03/01/07, 2007, from <https://wiki.medien.ifi.lmu.de/view/HCILab/EiToolkit>.
- Holmquist, L. E., H.-W. Gellersen, et al. (2004). Building Intelligent Environments with Smart-Its. IEEE Computer Graphics & Applications: 56-64.
- Inc, C. T. (2007). "Crossbow Technology Inc Wireless Sensor Networks." Retrieved 1st April, 2008, from <http://www.xbow.com/Home/HomePage.aspx>.
- IntelOpenCV. (2007). "Open Source Computer Vision Library (OpenCV)." Retrieved July, 2004, from <http://www.intel.com/technology/computing/opencv/>.
- Intille, S. S., V. Lee, et al. (2003). Ubiquitous Computing in the Living Room: Concept Sketches and an Implementation of a Persistent User Interface. UBICOMP 2003 Video Program, Seattle, Washington, USA, Springer LNCS.
- Ipiña, D. L. d., P. Mendonça, et al. (2002). "TRIP: a Low-Cost Vision-Based Location System for Ubiquitous Computing,." Personal and Ubiquitous Computing Journal 6(3): 206-219.
- Isard, M. and A. Blake (1998). "CONDENSATION—Conditional Density Propagation for Visual Tracking." International Journal of Computer Vision (IJCV) 29(1): 5-28.
- Isard, M. and A. Blake (1998). A Smoothing Filter for CONDENSATION. Proceedings of the 5th European Conference on Computer Vision-Volume I - Volume I, Springer-Verlag.
- Ishii, H. and B. Ullmer (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. Proceedings of the SIGCHI conference on Human factors in computing systems. Atlanta, Georgia, United States, ACM.
- Johnson, T. and H. Fuchs (2007). Real-Time Projector Tracking on Complex Geometry Using Ordinary Imagery. IEEE International Workshop on Projector-Camera Systems (ProCams2007). Minneapolis, MN, USA, IEEE.
- Jurie, F. and M. Dhome (2002). "Hyperplane Approximation for Template Matching." IEEE Trans. Pattern Anal. Mach. Intell. 24(7): 996-1000.
- Kale, A., K. Kwan, et al. (2004). Epipolar Constrained User Push Button Selection in Projected Interfaces. 1st CVPR workshop on Real time Vision for Human Computer Interaction. Washington DC, USA.
- Kalman, R. E. (1960). "A New Approach to Linear Filtering and Prediction Problems." Transactions of the ASME - Journal of Basic Engineering 82: 35-45.
- Karitsuka, T. and K. Sato (2003). A Wearable Mixed Reality with an On-Board Projector. Proceedings of the The 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society.

- Kato, H. and M. Billinghurst (1999). Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System. Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality, IEEE Computer Society.
- Kirstein, C. and H. Müller (1998). Interaction with a Projection Screen Using a Camera-Tracked Laser Pointer. The International Conference on Multimedia Modeling (MMM'98), IEEE Computer Society Press.
- Kjeldsen, R. (2005). Exploiting the Flexibility of Vision-Based User Interactions, IBM Technical Report.
- Kjeldsen, R., C. Pinhanez, et al. (2002). Interacting with Steerable Projected Displays. Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition, IEEE Computer Society.
- Klein, G. and T. Drummond (2003). Robust visual tracking for non-instrumental augmented reality. The Second IEEE and ACM International Symposium on Mixed and Augmented Reality.
- Klein, G. and T. Drummond (2004). Sensor Fusion and Occlusion Refinement for Tablet-Based AR. Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society.
- Klein, G. and D. Murray (2006). Full-3D Edge Tracking with a Particle Filter. British Machine Vision Conference (BMVC06), Edinburgh, UK.
- Klein, G. and D. Murray (2007). Parallel Tracking and Mapping for Small AR Workspaces. International Symposium on Mixed and Augmented Reality (ISMAR'07). Nara, Japan.
- Koike, H., Y. Sato, et al. (2001). "Integrating paper and digital information on EnhancedDesk: a method for realtime finger tracking on an augmented desk system." ACM Trans. Comput.-Hum. Interact. 8(4): 307-322.
- Koller, D., K. Danilidis, et al. (1993). "Model-based object tracking in monocular image sequences of road traffic scenes." Int. J. Comput. Vision 10(3): 257-281.
- Kölsch, M. and M. Turk (2002). Keyboards without Keyboards: A Survey of Virtual Keyboards. Sensing and Input for Media-centric Systems (SIMS 02).
- Kortuem, G., D. Alford, et al. (2007). Sensor Networks or Smart Artifacts? An Exploration of Organizational Issues of An Industrial Health and Safety Monitoring System. The Ninth International Conference on Ubiquitous Computing (UbiComp 2007). Innsbruck, Austria, Springer.
- Kortuem, G., C. Kray, et al. (2005). Sensing and visualizing spatial relations of mobile devices. Proceedings of the 18th annual ACM symposium on User interface software and technology. Seattle, WA, USA, ACM.
- Kotake, D., K. Satoh, et al. (2005). A Hybrid and Linear Registration Method Utilizing Inclination Constraint. Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society.
- Kotake, D., K. Satoh, et al. (2007). A Fast Initialization Method for Edge-based Registration Using an Inclination Constraint. Proceedings of the 6th IEEE/ACM International Symposium on Mixed and Augmented Reality. Nara, Japan, IEEE Computer Society.
- Krohn, A., M. Beigl, et al. (2005). Inexpensive and Automatic Calibration for Acceleration Sensors Ubiquitous Computing Systems: 245-258.
- Laberge, D. and J.-F. Lapointe (2003). An Auto-Calibrated Laser-Pointing Interface for Collaborative Environments. Ninth International Conference on Virtual Systems and Multimedia (VSMM 2003), Montréal, Québec, Canada.

- Lamdan, Y. and H. J. Wolfson (1998). Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. International Conference on Computer Vision (ICCV).
- Lamming, M. and D. Bohm (2003). SPECS: Personal Pervasive Systems. Computer. 36: 109-111.
- Lampe, M. and M. Strassner (2003). The potential of RFID for moveable asset management. Workshop on Ubiquitous Commerce at UbiComp 2003.
- Lee, J. C., P. H. Dietz, et al. (2004). Automatic projector calibration with embedded light sensors. Proceedings of the 17th annual ACM symposium on User interface software and technology. Santa Fe, NM, USA, ACM.
- Lee, J. C., S. E. Hudson, et al. (2005). Moveable interactive projected displays using projector based tracking. Proceedings of the 18th annual ACM symposium on User interface software and technology. Seattle, WA, USA, ACM.
- Leibe, B. and B. Schiele (2004). Scale Invariant Object Categorization Using a Scale-Adaptive Mean-Shift Search. DAGM'04 Annual Pattern Recognition Symposium, Tuebingen, Germany, Springer LNCS.
- Leiva, L. A., A. Sanz, et al. (2007). Planar tracking using the GPU for augmented reality and games. ACM SIGGRAPH 2007 posters. San Diego, California, ACM.
- Lepetit, V. and P. Fua (2005). "Monocular model-based 3D tracking of rigid objects." Found. Trends. Comput. Graph. Vis. 1(1): 1-89.
- Letessier, J. and F. Bérard (2004). Visual tracking of bare fingers for interactive surfaces. Proceedings of the 17th annual ACM symposium on User interface software and technology. Santa Fe, NM, USA, ACM.
- Levas, A., C. Pinhanez, et al. (2003). An Architecture and Framework for Steerable Interface Systems. UbiComp 2003: Ubiquitous Computing. Seattle, Washington, USA, Springer LNCS.
- Levy, B., S. Petitjean, et al. (2002). "Least Squares Conformal Maps for Automatic Texture Atlas Generation." ACM Transactions on Graphics 21(3): 162-170.
- Lewis, J. and K. Potosnack (1997). Keys and Keyboards. Handbook of Human-Computer Interaction. M. Helander, T. Landauer and P. Prabhu. Amsterdam, North-Holland: 1285-1316.
- Li, P. and F. Chaumette (2004). Image Cues Fusion for Object Tracking Based on Particle Filter. International Workshop on Articulated Motion and Deformable Objects (AMDO04), Palma de Mallorca, Spain, LNCS
- Lindeberg, T. (1990). "Scale-Space for Discrete Signals." IEEE Trans. Pattern Anal. Mach. Intell. 12(3): 234-254.
- Lowe, D. G. (1992). "Robust model-based motion tracking through the integration of search and estimation." Int. J. Comput. Vision 8(2): 113-122.
- Lowe, D. G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints." International Journal of Computer Vision 60(2): 91-110.
- Majumder, A. and G. Welch (2001). Computer Graphics Optique: Optical Superposition of Projected Computer Graphics. Eurographics Workshop on Virtual Environment/ Immersive Projection Technology.
- Malbezin, P., W. Piekarski, et al. (2002). Measuring ARToolkit Accuracy in Long Distance Tracking Experiments. 1st Int'l Augmented Reality Toolkit Workshop, IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2002). Darmstadt, Germany.
- Matas, J., O. Chum, et al. (2002). Robust wide baseline stereo from maximally stable extremal regions. British Machine Vision Conference, London, UK.

- Matsumoto, T., D. Horiguchi, et al. (2006). Z-agon: mobile multi-display browser cube. CHI '06 extended abstracts on Human factors in computing systems. Montréal, Québec, Canada, ACM.
- Mikolajczyk, K. and C. Schmid (2002). An Affine Invariant Interest Point Detector. Proceedings of the 7th European Conference on Computer Vision-Part I, Springer-Verlag.
- Mikolajczyk, K. and C. Schmid (2004). "Scale & Affine Invariant Interest Point Detectors." Int. J. Comput. Vision 60(1): 63-86.
- Mikolajczyk, K. and C. Schmid (2005). "A Performance Evaluation of Local Descriptors." IEEE Trans. Pattern Anal. Mach. Intell. 27(10): 1615-1630.
- Mikolajczyk, K., T. Tuytelaars, et al. (2005). "A Comparison of Affine Region Detectors." Int. J. Comput. Vision 65(1-2): 43-72.
- Milgram, P. and F. Kishino (1994). "A Taxonomy of Mixed Reality Visual Displays." IEICE Transactions on Information Systems, E77-D(12).
- MitsubishiElectricCorporation. (2008). "Mitsubishi PK20 Pocket Projector Website." Retrieved 19th March, 2008, from <http://global.mitsubishielectric.com/bu/projectors/products/home/pk20.html>.
- Molyneaux, D. and H. Gellersen (2006). Cooperatively Augmenting Smart Objects with Projector-Camera Systems. 3rd IEEE International Workshop on Projector-Camera systems (ProCams 2006) New York, USA.
- Molyneaux, D., H. Gellersen, et al. (2007). Cooperative Augmentation of Smart Objects with Projector-Camera Systems. UbiComp 2007: Ubiquitous Computing. Innsbruck, Austria, Springer LNCS.
- Molyneaux, D., H. Gellersen, et al. (2008). Vision-Based Detection of Mobile Smart Objects. 3rd IEEE European Conference on Smart Sensing and Context (EuroSSC). Zurich, Switzerland, Springer LNCS (to appear).
- Montemayor, A. S., J. J. Pantrigo, et al. (2004). Particle filter on GPUs for real-time tracking. ACM SIGGRAPH 2004 Posters. Los Angeles, California, ACM.
- Morishima, S., T. Yotsukura, et al. (2000). HyperMask: Talking Head Projected onto Real Object. International Conference on Multimedia Modeling (MMM 2000), Nagano, Japan.
- Murase, H. and S. K. Nayar (1995). "Visual Learning and Recognition of 3D Objects from Appearance." International Journal on Computer Vision 14(1): 5-24.
- Nachman, L., R. Kling, et al. (2005). The Intel Mote platform: a Bluetooth-based sensor network for industrial monitoring. Proceedings of the 4th international symposium on Information processing in sensor networks. Los Angeles, California, IEEE Press.
- Naimark and Foxlin (2002). Circular Data Matrix Fiducial Systems and Robust Image Processing for a Wearable Vision-Inertial self Tracker. IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2002). Darmstadt, Germany, September-October.
- Nakamura, N. and R. Hiraie (2002). Active Projector: Image correction for moving image over uneven screens. 15th Annual ACM Symposium on User Interface Software and Technology (UIST2002).
- Nakazato, Y., M. Kanabara, et al. (2004). Discreet Markers for User Localisation. Eighth International Symposium on Wearable computers (ISWC 2004). Arlington, VA, USA.
- Nayar, S. K., H. Peri, et al. (2003). A projection system with radiometric compensation for screen imperfections. ICCV Workshop on Projector-Camera Systems (PROCAMS). Nice, France.

- Newman, J., A. Bornik, et al. (2007). Tracking for distributed mixed reality environments. IEEE VR 2007 Workshop on Trends and Issues in Tracking for Virtual Environments. Charlotte, NC, USA, IEEE Shaker Verlag.
- Norman, D. A. (1988). The Design of Everyday Things. Boston, Massachusetts, MIT Press.
- Oberkamp, D., D. F. DeMenthon, et al. (1996). "Iterative pose estimation using coplanar feature points." Comput. Vis. Image Underst. 63(3): 495-511.
- Oh, J. Y. and W. Stuerzlinger (2002). Laser Pointers as Collaborative Pointing Devices. Graphics Interface 2002. Stürzlinger and McCool, AK Peters and CHCCS: 141-149.
- OpenSG. (2007). "OpenSG Website." Retrieved January, 2007, from <http://www.opensg.org/>.
- Park, H., M.-H. Lee, et al. (2005). Specularity-Free Projection on Nonplanar Surface. Advances in Multimedia Information Processing - PCM 2005, Springer Berlin / Heidelberg.
- Park, H., M.-H. Lee, et al. (2006). "Surface-independent direct-projected augmented reality." Lecture Notes in Computer Science 3852, Computer Vision: 892-901.
- Park, H., M.-H. Lee, et al. (2007). Content Adaptive Embedding of Complementary Patterns for Nonintrusive Direct-Projected Augmented Reality. International Conference on Human-Computer Interaction (HCI International'07) Beijing, China.
- Park, H., M.-H. Lee, et al. (2006). "Undistorted Projection onto Dynamic Surface " Springer LNCS - Advances in Image and Video Technology 4319/2006: 582-590.
- Park, H. and J. Park (2004). Invisible Marker Tracking for AR. Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04). Arlington, VA, USA.
- Pingali, G., C. Pinhanez, et al. (2003). Steerable Interfaces for Pervasive Computing Spaces. Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, IEEE Computer Society.
- Pingali, G., C. Pinhanez, et al. (2002). User-Following Displays. IEEE International Conference on Multimedia and Expo 2002 (ICME'02). Lausanne, Switzerland.
- Pinhanez, C., R. Kjeldsen, et al. (2001). Transforming Surfaces into Touch-Screens, IBM Research Report
- Pinhanez, C., R. Kjeldsen, et al. (2002). Ubiquitous Interactive Graphics. IBM Research Report RC22495 (W0205-143).
- Pinhanez, C., F. Nielsen, et al. (1999). Projecting computer graphics on moving surfaces: a simple calibration and tracking method. ACM SIGGRAPH 99 Conference abstracts and applications. Los Angeles, California, United States, ACM Press.
- Pinhanez, C. and M. Podlaseck (2005). To Frame or Not to Frame: The Role and Design of Frameless Displays in Ubiquitous Applications. Ubicomp 2005. Tokyo, Japan.
- Pinhanez, C. S. (2001). The Everywhere Displays Projector: A Device to Create Ubiquitous Graphical Interfaces. Proceedings of the 3rd international conference on Ubiquitous Computing. Atlanta, Georgia, USA, Springer-Verlag.
- Podlaseck, M., C. Pinhanez, et al. (2003). On interfaces projected onto real-world objects. CHI '03 extended abstracts on Human factors in computing systems. Ft. Lauderdale, Florida, USA, ACM.

- Pupilli, M. and A. Calway (2006). Real-Time Camera Tracking Using Known 3D Models and a Particle Filter. Proceedings of the 18th International Conference on Pattern Recognition - Volume 01, IEEE Computer Society.
- Raij, A. and M. Pollefeys (2004). Auto-calibration of multi-projector display walls. International Conference on Pattern Recognition, Cambridge, UK, IEEE Computer Society.
- Rapp, S., G. Michelitsch, et al. (2004). Spotlight Navigation: Interaction with a Handheld Projection Device. International Conference on Pervasive Computing 2004 Video Proceedings.
- Raskar, R., P. Beardsley, et al. (2004). RFIG lamps: interacting with a self-describing world via photosensing wireless tags and projectors. ACM SIGGRAPH 2004. Boston, Massachusetts, ACM.
- Raskar, R., P. Beardsley, et al. (2006). RFIG lamps: interacting with a self-describing world via photosensing wireless tags and projectors. ACM SIGGRAPH 2006 Courses. Boston, Massachusetts, ACM.
- Raskar, R., H. Nii, et al. (2007). "Prakash: lighting aware motion capture using photosensing markers and multiplexed illuminators." ACM Transactions on Graphics 26(3): 36.
- Raskar, R., J. VanBaar, et al. (2005). iLamps: geometrically aware and self-configuring projectors. ACM SIGGRAPH 2005 Courses. Los Angeles, California, ACM.
- Raskar, R., G. Welch, et al. (1999). Spatially augmented reality. Proceedings of the international workshop on Augmented reality : placing artificial objects in real scenes: placing artificial objects in real scenes. Bellevue, Washington, United States, A. K. Peters, Ltd.
- Rekimoto, J. (2002). SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves. Minneapolis, Minnesota, USA, ACM.
- Rensink, R. A. (2000). "When Good Observers Go Bad: Change Blindness, Inattentional Blindness, and Visual Experience." Psyche 6(09).
- Robertson, C. and J. Robinson (1999). Live paper: video augmentation to simulate interactive paper. Proceedings of the seventh ACM international conference on Multimedia (Part 2). Orlando, Florida, United States, ACM.
- Rolland, J. P., Y. Baillot, et al. (2001). A Survey of Tracking Technology for Virtual Environments. Fundamentals of Wearable Computers and Augmented Reality. Mahwah, NJ, USA, Lawrence Erlbaum Assoc. Inc.: 67–112.
- Rosten, E. and T. Drummond (2005). Fusing points and lines for high performance tracking. IEEE International Conference on Computer Vision, IEEE.
- Rosten, E. and T. Drummond (2006). Machine learning for high-speed corner detection. European Conference on Computer Vision.
- Rothganger, F., S. Lazebnik, et al. (2006). "3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints." Int. J. Comput. Vision 66(3): 231-259.
- Salvi, J., J. Pages, et al. (2004). "Pattern Codification Strategies in Structured Light Systems." Pattern Recognition 37(4): 827-849.
- Santos, P., A. Stork, et al. (2006). Innovative geometric pose reconstruction for marker-based single camera tracking. Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications. Hong Kong, China, ACM.

- Schaffalitzky, F. and A. Zisserman (2001). Viewpoint invariant texture matching and wide baseline stereo. International Conference on Computer Vision (ICCV), Vancouver, BC, Canada, IEEE.
- Schiele, B. and J. L. Crowley (2000). "Recognition without Correspondence using Multidimensional Receptive Field Histograms." International Journal of Computer Vision (IJCV) 36(1): 31-50.
- Schmalstieg, D. and D. Wagner (2007). Experiences with Handheld Augmented Reality. The Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2007), Nara, Japan, IEEE and ACM.
- Schmidt, A., M. Kranz, et al. (2005). Interacting with the ubiquitous computer: towards embedding interaction. Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies. Grenoble, France, ACM.
- Schmidt, A., M. Strohbach, et al. (2002). Ubiquitous Interaction - Using Surfaces in Everyday Environments as Pointing Devices. 7th ERCIM Workshop "User Interfaces For All" Springer Verlag LNCS.
- Sheridan, J. G., B. W. Short, et al. (2003). Exploring Cube Affordance: Towards A Classification Of Non-Verbal Dynamics Of Physical Interfaces For Wearable Computing. IEE Eurowearable 2003, IEE Press. Birmingham, UK.
- Siegemund, F. and C. Flörkemeier (2003). Interaction in Pervasive Computing Settings Using Bluetooth-Enabled Active Tags and Passive RFID Technology Together with Mobile Phones. First IEEE International Conference on Pervasive Computing and Communications (PerCom), IEEE Computer Society.
- Sinha, S. N., J.-M. Frahm, et al. (2006). GPU-Based Video Feature Tracking and Matching. EDGE 2006, workshop on Edge Computing Using New Commodity Architectures. Chapel Hill.
- Sinha, S. N. and M. Pollefeys (2006). "Pan-tilt-zoom camera calibration and high-resolution mosaic generation." Comput. Vis. Image Underst. 103(3): 170-183.
- Spassova, L. (2004). Fluid Beam -- Konzeption und Realisierung eines Display-Kontinuums mittels einer steuerbaren Projektor-Kamera-Einheit. Fakultät für Informatik. Saarbrücken, Germany, Universität des Saarlandes. http://w5.cs.uni-sb.de/publication/file/244/Mira_Spassova_Diplom_Fluid_Beam.pdf.
- Spengler, M. and B. Schiele (2001). Towards Robust Multi-cue Integration for Visual Tracking. Proceedings of the Second International Workshop on Computer Vision Systems, Springer-Verlag.
- Stauffer, C. and W. E. L. Grimson (1999). Adaptive background mixture models for real-time tracking. Computer Vision Pattern Recognition (CVPR).
- Strohbach, M. (2004). The Smart-its Platform for Embedded Context-Aware Systems. First International Workshop on Wearable and Implantable Body Sensor Networks. London.
- Strohbach, M., H.-W. Gellersen, et al. (2004). Cooperative Artefacts: Assessing Real World Situations with Embedded Technology. UbiComp2004. Nottingham, England, Springer LNCS.
- Strzodka, R., I. Ihrke, et al. (2003). A Graphics Hardware Implementation of the Generalized Hough Transform for fast Object Recognition, Scale, and 3D Pose Detection. Proceedings of the 12th International Conference on Image Analysis and Processing, IEEE Computer Society.
- Sukaviriya, N., M. Podlaseck, et al. (2003). Embedding Interactions in a Retail Store Environment: The Design and Lessons Learned. Ninth IFIP International

- Conference on Human-Computer Interaction (INTERACT'03). Zurich, Switzerland.
- Sukthankar, R., R. Stockton, et al. (2001). Smarter Presentations: Exploiting Homography in Camera-Projector Systems. International Conference on Computer Vision. Vancouver, Canada.
- Summet, J., M. Flagg, et al. (2007). "Shadow Elimination and Blinding Light Suppression for Interactive Projected Displays." IEEE Transactions on Visualization & Computer Graphics (TVCG) 13(3).
- Summet, J. and R. Sukthankar (2005). Tracking Locations of Moving Hand-held Displays Using Projected Light. Proceedings of Pervasive 2005. Munich, Germany, Springer LNCS.
- Swain, M. J. and D. H. Ballard (1991). "Color indexing." International Journal of Computer Vision 7(1): 11-32.
- Terrenghi, L., M. Kranz, et al. (2006). "A cube to learn: a tangible user interface for the design of a learning appliance " Personal and Ubiquitous Computing Journal 10(2-3): 153-158.
- TexasInstruments. (2005). "Texas Instruments DLP Website." Retrieved 10th March, 2005, from www.dlp.com.
- Turk, M. and A. Pentland (1991). "Eigenfaces for Recognition." Journal of Cognitive Neuroscience 3(1): 71-86.
- Underkoffler, J., B. Ullmer, et al. (1999). Emancipated pixels: real-world graphics in the luminous room. Proceedings of the 26th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co.
- UPnP(TM)Forum, C. M. o. t. (2003). "Welcome to the UPnP(TM) Forum." Retrieved 27/04/05, 2005, from <http://www.upnp.org/>
- Vacchetti, L. and V. Lepetit (2004). "Stable Real-Time 3D Tracking Using Online and Offline Information." IEEE Trans. Pattern Anal. Mach. Intell. 26(10): 1385-1391.
- Vacchetti, L., V. Lepetit, et al. (2004). Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society.
- Valli, A. (2005). "Notes on natural interaction." Retrieved 17th March, 2005, from www.naturalinteraction.org.
- VanLaerhoven, K. (2006). "CommonSense Project." Retrieved 01/01/08, from <http://eis.comp.lancs.ac.uk/index.php?id=commonsense>.
- VanLaerhoven, K., N. Villar, et al. (2003). Using an autonomous cube for basic navigation and input. Proceedings of the 5th international conference on Multimodal interfaces. Vancouver, British Columbia, Canada, ACM.
- VanRhijn, A. and J. D. Mulder (2005). An Analysis of Orientation Prediction and Filtering Methods for VR/AR. Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality, IEEE Computer Society.
- Wagner, D. and D. Schmalstieg (2006). Handheld Augmented Reality Displays. Proceedings of the IEEE conference on Virtual Reality, IEEE Computer Society.
- Weiser, M. (1991). The computer for the 21st century. Scientific American. 3: 94-104.
- Weiser, M. (1996). "Ubiquitous Computing." Retrieved 1st April, 2005, from <http://www.ubiq.com/hypertext/weiser/UbiHome.html>.
- Weiser, M. and J. S. Brown (1996). "Designing Calm Technology." PowerGrid Journal 1(1).

- Welch, G. and G. Bishop (1997). SCAAT: incremental tracking with incomplete information. Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co.
- Wellner, P. (1993). "Interacting with paper on the DigitalDesk." Commun. ACM 36(7): 87-96.
- Wilson, A. D. (2004). TouchLight: an imaging touch screen and display for gesture-based interaction. Proceedings of the 6th international conference on Multimodal interfaces. State College, PA, USA, ACM.
- Wilson, A. D. (2005). PlayAnywhere: a compact interactive tabletop projection-vision system. Proceedings of the 18th annual ACM symposium on User interface software and technology. Seattle, WA, USA, ACM.
- Wolfson, H. J. (1990). Model-based object recognition by geometric hashing. Proceedings of European Conference on Computer Vision (ECCV'90), Springer LNCS.
- Yang, R., D. Gotz, et al. (2001). PixelFlex: a reconfigurable multi-projector display system. Proceedings of the conference on Visualization '01. San Diego, California, IEEE Computer Society.
- Yang, R. and G. Welch (2001). Automatic Projector Display Surface Estimation Using Every-Day Imagery. 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision. Plzen, Czech Republic.
- You, S. and U. Neumann (2001). Fusion of Vision and Gyro Tracking for Robust Augmented Reality Registration. IEEE Virtual Reality Conference 2001 (VR 2001).
- You, S., U. Neumann, et al. (1999). Hybrid Inertial and Vision Tracking for Augmented Reality Registration. Proceedings of the IEEE Virtual Reality, IEEE Computer Society.
- Zhang, X., S. Fronz, et al. (2002). Visual Marker Detection and Decoding in AR Systems: A Comparative Study. International Symposium on Mixed and Augmented Reality (ISMAR'02). Darmstadt, Germany.
- Zhang, Z. (2000). "A Flexible New Technique for Camera Calibration." IEEE Trans. Pattern Anal. Mach. Intell. 22(11): 1330-1334.

Appendix A Steerable Projector-Camera System Construction

This appendix describes the design and assembly of two steerable projector-camera systems. We provide an overview of design characteristics of typical systems and present recommendations for those wanting to build similar equipment.

The hardware design used in this research is inspired by four closely related projects: the Everywhere Display by Pinhanez et al., the FLUIDbeam project by Butz et al. the Projected Augmentation projector by Ehnes et al. and the PRIMA project by Borkowski et al. [Pinhanez 2001; Borkowski, Riff et al. 2003; Butz, Schneider et al. 2004; Ehnes, Hirota et al. 2004]. Consequently, we present an example characterisation of one of the steerable projector systems and compare it to other commercial and research systems.

A.1 Steerable Projector-Camera System Design

Steerable Projector-Camera systems share three common classes of components from which the system is constructed:

1. Steering Mechanism
2. Video Projector
3. Camera

These major characteristics of these components are identified in the sections below to allow objective measurement of the performance of steerable projectors and to allow comparison of strengths and weaknesses in different approaches and implementations.

A.2 Steering Mechanism

The two most popular methods to create a steerable projector from a fixed projector are to use either a moving mirror in front of the projector lens, or by mounting the projector in a moving head yoke.

The steering mechanisms currently used in steerable projectors have their roots in the display lighting industry. The mechanisms are usually contained in moving head lights and moving mirror scanners used for dynamic lighting on stage, in discos, or in retail

and entertainment environments. These lights are typically controlled by the uni-directional DMX512/1990 serial protocol (often shortened to DMX).



Figure A.1 (left) Moving mirror display light, (right) Moving head display light [SteinigkeShowtechnikGmbH 2005]

Both types of steering mechanism provide two degrees of freedom – pan and tilt. This freedom is implemented in both systems as rotation of an object (mirror or projector) around the respective axes. The world axis around which this rotation occurs changes depending on the orientation of device mounting, so when referring to pan and tilt in this document we assume it is relative to the local steering mechanism mounting. Typically the steering mechanism is mounted so pan is movement in the horizontal direction and tilt is movement in the vertical plane.

The moving mirror approach was first implemented by IBM for their Everywhere Display prototype in 2000 [Pinhanez 2001] and later by UNC in their PixelFlex reconfigurable multi-projector display [Yang, Gotz et al. 2001].

The moving head approach is used by the majority of steerable projector implementations. First proposed by Nakamura and Hiraie [Nakamura and Hiraie 2002], this steering method is implemented in the FLUIDbeam project by Butz et al. the Projected Augmentation projector by Ehnes et al. and the PRIMA project by Borkowski et al. [Pinhanez 2001; Borkowski, Riff et al. 2003; Butz, Schneider et al. 2004; Ehnes, Hirota et al. 2004].

Purely optical beam steering mechanisms, such as lenses, spatial light modulators or Holographic Optical Elements (HOE) are all possible. However, unlike the mechanical steering methods, such systems are generally not commercially available, hence, will not be discussed further.

A.2.1 Steering Mechanism Characteristics

We identified 12 important characteristics of steering mechanisms:

1. Mechanism Rotation Speed

Faster rotation capability is better, allowing for rapid tracking of objects at close range and fast movement between spatially distant objects. However, high speed is not an absolute requirement, as continuous small movements in different directions once actively tracking an object may never achieve full speed rotation due to the acceleration and deceleration time required. These types of movements would be governed more by the mechanism acceleration and inertia characteristics. We measure both moving mirror and moving head angular rotation speed in degrees per second.

2. Mechanism Acceleration and Inertia

System inertia is dependant on the mass that needs to be rotated – in the case of moving head systems a projector is much more massive than a mirror, so will have higher inertia. High inertia systems are more likely to exhibit control problems such as overshoot when attempting to position accurately. Lower inertia systems are more responsive as they allow faster acceleration and more rapid changes in direction.

We measure angular acceleration in terms of degrees per second squared, but this is a difficult quantity to practically measure, without accurate position feedback from the steering mechanism or accelerometers mounted on the steering mechanism.

3. Mechanism Field of View

From a fixed location, many moving head systems are capable of panning and tilting to cover an area greater than a hemisphere. However, moving mirror systems have a more limited field of view due to optical arrangement of the mirror and projector, as shown in Figure A.2 (left). Here, the rotation in pan is relatively unconstrained except for physical obstruction of the mechanism mounting bracket. Rotation in tilt is much more limited - both by the requirement to keep the reflective mirror surface towards the projection beam when tilting up, and by the projector body itself occluding the light when tilted down.

Field Of View (FOV) is measured in degrees of coverage for each axis separately (pan and tilt). A larger field of view is preferable as it allows the flexibility for a single steerable projector system to create displays in a larger spatial area.

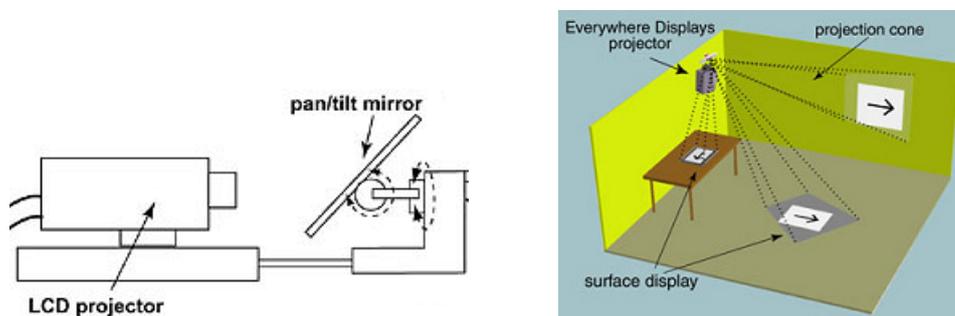


Figure A.2 (left) The IBM Everywhere Display Steerable Projector Design, (right) Everywhere Display Projection Cone

[Pinhanez 2001]

4. Mechanism Positioning Accuracy

The Steering mechanism will have a mechanical accuracy determined by the mechanism FOV angles in pan and tilt and the control interface resolution. On display lighting the DMX control interface is either 8-bit or 16-bit, allowing either 256 or 65536 discrete positions to be resolved respectively. These discrete positions are the number of possible angular positions the pan and tilt unit can be commanded to occupy in each axis. For systems with a high mechanical FOV and low resolution control interface the angle between discrete positions will be large, and small movements between the commanded positions will appear jerky. Conversely with a low mechanical FOV and high control resolution, the angular resolution is much higher; hence changes in display position will look smooth.

We measure the angular positioning resolution in degrees by dividing the mechanical FOV by the number of addressable control positions (e.g. $360^\circ \text{ FOV} / 256 \text{ steps} = 1.4^\circ$)

per step). High accuracy positioning (a lower numerical figure) is recommended, as it allows displays to be located more accurately and less jerky display movement when projecting at distance.

5. Mechanism Positioning Repeatability

While the positioning accuracy measures how accurate a position is, the repeatability measures how close the mechanism returns to the same commanded position following a movement. Over time mechanical systems exhibit wear, hence repeatability is determined both by the mechanical properties of the steering mechanism and the ability of the built-in position feedback sensing system to correctly measure its rotation.

The control interface uses a closed-loop control method, so will command the mechanism to move to a specific position and monitor the movement until it reaches the correct position as determined by the position feedback sensors. These sensors are typically optical break-beam sensors for moving head and resistive sensors for servos in moving mirror systems. The optical sensors use a circular disk with cut holes to create pulses as they pass through the break-beam sensor. By dividing the number of detected holes by the known angular range (or required numerical control range) between the end stops, the yoke system can move to an absolute position by simply moving until it reaches the correct number of holes from the end stop. The resistive sensors are typically potentiometers attached to the rotating spindle which encode position by change in resistance. This is converted to a change in voltage using a voltage divider circuit and compared with the input voltage used to control the servos.

For a repeatable position the mechanism rotation motors and position feedback sensors need to be accurate enough to resolve to the same accuracy as the control system. The control system must also be able to dynamically manage under and overshoot as rotation stops; otherwise it exhibits oscillation around the required position (hunting). A steering mechanism with high repeatability is preferable, as it allows displays to return to the same spatial locations consistently.

6. Position Stability

Positional stability measures how well a system can hold an exact position without un-commanded movement occurring. This characteristic is determined partly by the torque characteristics of the positioning motors, partly by the mass of the load the motors must hold (projector or mirror) and whether any other external forces are applied. For stability, any jitter in the control and position feedback systems must be eliminated. However, using this definition, stability is difficult to measure quantitatively – a steering mechanism is generally either stable or not (for example, if a projector is too heavy for a moving head steering mechanism). Any useful system is required to have positional stability in all possible orientations.

Positional stability could also be defined as the steering mechanism and mounting method's response to rotational movement – for example, whether a panning projector using the moving head method causes unwanted structural response when it starts and stops due to its inertia. Again this is difficult to quantify except as the magnitude of structural response (which would require mounting accelerometers), or possibly as the time taken for the response to settle.

7. Size

The size of a steerable projector is determined mainly by the size of a projector for a moving mirror mechanism. Smaller projectors have smaller lenses, allowing correspondingly smaller mirrors and mechanisms to steer the beam. Moving head systems are necessarily larger than the projector themselves due to their design, which encapsulates the projector at the centre of rotation. Smaller is generally better, as it allows easier mounting and better portability. Size is best measured as the volume occupied by the system in metres cubed.

8. Weight

The weight of a steering mechanism, measured in kilograms (kg), will determine how it can be mounted, and how portable it will be. Due to their small size, moving mirror systems can be very light (hundreds of grams), whereas moving head systems are generally in the tens of kilograms range. Lighter is better, as it increases the portability and mounting options.

9. Mounting Options

For fixed steerable projectors, the type of mounting required will depend heavily on the size, weight and portability of the steering mechanism and projector. The environment will also play a large role, as there is a big structural difference between mounting direct to a concrete ceiling, and mounting to a false ceiling. Generally, a lighter weight system will be more flexible in mounting location than a heavier one.

10. Portability

Although current steerable projectors generally have fixed mountings, portable steerable projector systems would share characteristics with handheld projectors and wearable projectors described in related work section 2.3.2.

11. Image Distortion

Although not easily quantifiable, different beam steering methods cause different amounts and types of distortion in a projected image. For example, moving mirror systems will exhibit more rotation distortion of the uncorrected image than moving head systems, due to optical arrangement caused by the fixed projector and moving mirror, as seen in Figure A.2 (right). Less distortion is always preferred, which is accomplished by projecting as orthogonal to the display surface as possible.

12. Cost

There is a price-performance trade-off as more expensive systems generally have better positioning accuracy, stability, repeatability and build quality. An acceptable balance must always be found between these specifications and price.

A.2.2 Moving Mirror and Moving Head comparison

Moving Mirror

Benefits

- Very fast rotation speed possible with fast acceleration due to low inertia
- Small size and volume

- Lightweight unit, easily mounted - possible to “bolt-on” to the projector
- Can be used with a portable stand or with new generation of small portable projectors as mirror size scales with projector lens size.
- Low cost
- Camera has separate pan and tilt unit, allowing multitasking (e.g. object searching) while not required for interaction or calibration

Problems

- Restricted field of view (Typically around 230° pan, 50° tilt [Pinhanez 2001]), requiring location high in a corner or high along the centre of a wall for largest coverage.
- Only limited 8-bit accuracy pan/tilt positioning units available
- Increased image distortion due to angled mirror surface in light path (primarily causing image rotation distortion)
- Camera requires separate pan and tilt unit, leading to calibration, registration and coordination problems with projector

Moving Head

Benefits

- Fast Rotation speed is possible
- Good for central location in large environments, as many pan and tilt units coverage greater than a hemisphere (e.g. 360° coverage in pan and 270° in tilt)
- Less distortion than moving mirror due to direct projection on surfaces.
- Camera can be directly attached to the projector, so projector-camera calibration can be performed once, with no further registration and coordination problems

Problems

- High inertia can cause poor acceleration, precluding very fast object tracking
- Large Steering mechanism volume
- Heavy unit, requiring professional installation on a wall or ceiling
- No portability
- High Cost
- Camera is attached to projector, so cannot be used for anything else (e.g. searching) while projector static

A.3 Video Projector

The video projector is required to display information on the surfaces of objects as visibly as possible. We identified 10 projector characteristics to take into account:

1. Brightness

The brightness of a projector is typically measured in American National Standards Institute (ANSI) Lumens – which is a measurement of the total amount of light a

projector can produce when projecting a white image. To be clearly visible in ambient light the projector must output a greater amount of light to a given surface area than the ambient lighting.

The perceived brightness of a projected image will vary depending on the size of the projection and the reflective properties of the projection surface. The larger the projected image, the darker it will appear, as the fixed lamp light output is spread over the increasingly larger area of projection surface.

The reflectivity of the projection surface will also have an impact, as projection on low reflectivity surfaces (dark, diffusing, matt surfaces) will appear substantially darker than surfaces with higher reflectivity (bright, reflecting, glossy surfaces). However, there is a trade-off, as very reflective surfaces will also increase the probability of “hotspots”, where the projector and other light sources are clearly visible as bright reflection spots on the surface. This reflectivity measurement is expressed as “gain” for commercial projection screens; however, as steerable projectors just use everyday surfaces in the environment, no surface gain measurements are typically available.

2. Contrast

Contrast (also known as dynamic range) is the apparent difference between projected white light and the absence of projected light (black). In the presence of normal office or home illumination an absolute black (i.e. total darkness) cannot be attained by front projection, as there is always ambient light reflecting from the display surface. Ambient light reduces the difference between the projector’s maximum white and black (i.e. no projection), reducing its contrast and making the image look “washed out”. Higher contrast ratios are preferable for better the readability of the display.

One method of increasing both the brightness and contrast of projected displays was demonstrated by Majumder and Welch [Majumder and Welch 2001], who superimposed two identical projector displays to double the maximum brightness. By changing the display addressing system this also allowed high dynamic range colour images to be displayed.

3. Resolution

As steerable projectors are frequently projecting off-axis or on rotated objects, there can be significant amounts of geometric distortion in the projected image. The correction methods discussed in related work section 2.3.6 distort the projected image, which reduces the effective resolution we are projecting.

Pinhanez et al. claimed their effective resolution was typically reduced from 1024x768 pixels to 640x480 pixels following geometric correction [Pinhanez, Kjeldsen et al. 2002] in their moving mirror Everywhere Display system. Consequently, the greater the resolution of the projector we start with, the better the final image will look.

4. Projector Technology – LCD versus DLP

Liquid Crystal Displays (LCD) are the traditional technology used in projectors – they are relatively cheap, widely available and create a stable image. Projectors typically contain three LCD panels - one for each colour (Red, Green, Blue), each around 1-2 inches in size.

Texas Instruments developed a single chip Digital Light Processor (DLP) engine for projectors using a chip less than an inch square with the surface covered in millions of

miniature reflecting mirrors, allowing projector size to be reduced substantially. Without the need for three separate colour panels and integrating optics, smaller and more lightweight projectors are possible with the equivalent brightness as an LCD projector. However, there is one major drawback with consumer DLP projectors – they possess only a single DLP chip, so a rotating colour wheel must be used to time sequentially display frames of red, green and blue colour, as shown in Figure A.3 (left). In many models a fourth transparent area is also used on the colour wheel to boost light output at the cost of colour resolution, as humans are more sensitive to changes in luminance than colour. These frames are integrated and seen as a full colour image by a human eye.

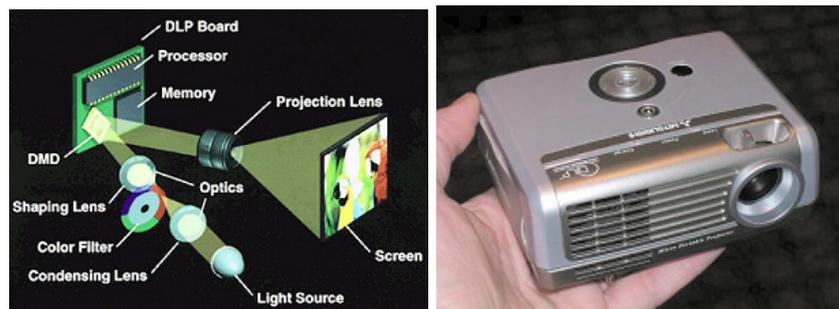


Figure A.3 (left) Single chip DLP projector optics [TexasInstruments 2005], (right) Mitsubishi Pocket Projector [MitsubishiElectricCorporation 2008]

However, cameras can have an exposure shorter than the integration period, so may see only a single colour frame or partial frame. To see the whole frame a camera must either be explicitly synchronised with the projector, or the exposure increased to integrate all the colour components (16.7ms for a 60Hz projector refresh). This problem mainly exists when the camera auto exposes to the projector illumination – typically in darker environments where the projector illumination is brighter than ambient illumination, or where the projected image fills the majority of the camera frame. Hence, the problem can be reduced by manually setting the camera exposure rather than relying on auto-exposure.

5. Lamp Life

The use of a traditional video projector in steerable projectors raises issues with the limited lamp life (generally in the region of a few thousand hours), the high replacement cost and large size. Companies are now producing Light Emitting Diode (LED) projectors and LASER based projectors small enough to fit in the palm of a hand, with light sources that last over 20,000 hours (i.e. the design lifetime of the projector). Although the current projectors are only low brightness and resolution, this technology should be expected to replace metal halide lamps in video projectors. An example LED projector can be seen in Figure A.3 (right).

6. Lens Location

Lens location is another consideration for steerable projector systems. If using a moving head system then the optical Centre of Projection (COP) should ideally be at the Centre Of Rotation (COR) of the pan and tilt unit to make calibration easier. In this spatial arrangement only the rotation component of the projector to world coordinate system transformation changes with the rotation of the projector, rather than a combined

rotation and translation for offset lenses. Consequently, a centre lens projector design should use a dual fork yoke, and an offset lens projector should ideally use a single arm yoke so the COP is always at the COR.

7. Powered Focus

Projectors generally exhibit a good depth of field, with Pinhanez finding that surfaces up to 30° inclination relative to the projector axis generally remaining in focus [Pinhanez, Kjeldsen et al. 2002]. However, computer controlled powered focus allows projection at dynamically varying depths, which is useful when tracking mobile objects.

When available, LASER based projectors will not require any focus control. Their coherent and collimated light output remains in focus at all depths, allowing projected displays completely in focus on objects at extremely oblique angles and simultaneous projection on foreground and background objects with large differences in distance.

8. Powered Zoom

Computer controlled powered zoom allows different sized displays to be projected at different distances. Projector zoom is measured as a ratio between the largest and smallest image produced with the zoom at wide and telephoto respectively. For example, 1:1.3 indicates the image can be resized by 30%. Higher zoom ratios enable projection of smaller images at longer distances (giving higher resolution) or larger images at close distances (but with lower resolution).

It is useful to know what display size and resolution can be expected from a projector at each distance, as this will determine what size objects can be projected on. The actual achieved display resolution will vary with the projector FOV, projector zoom, distance to the display surface, its size and orientation (due to geometric correction), but can be calculated with known object geometry, pose and projector intrinsic parameters.

9. Weight

The weight of a projector is generally proportional to its brightness and cost. Projectors over 3500 Lumens typically cost significantly more and weight more, as they are designed for large displays and may incorporate two lamps.

The projector weight will partly determine which steering method is used - heavier projectors are difficult to use with moving head steering mechanism, as they have a high mass and hence a high inertia. Conversely, high weight is not a problem for moving mirror systems, as the projector itself is statically mounted and only the mirror moves.

10. Cost

The cost of consumer projectors is generally proportional to their resolution and brightness. All other things being equal, a lower cost projector is preferable.

A.4 Camera

We identified 7 relevant characteristics of typical cameras:

1. Camera Type and Mounting

Steerable projector-camera systems use the camera for display calibration and to detect user interaction, hence the steerable projector camera requires a view of the

display surface. An identical view can be provided by making the projector and camera co-axial using a partially reflective optical beam-splitter, as discussed by Fujii et al. [Fujii, Grossberg et al. 2005] and seen in Figure A.4 (left). However, use of a beam-splitter reduces the light output from the projector and the co-axial optics prevent use of the camera with structured light geometry calibration methods, which require an offset camera to recover the surface geometry, as discussed in related work section 2.3.6 and section 6.4.

Instead, moving head steering systems can have a camera mounted directly to the projector, close to the projector lens. This location gives an on-axis view similar to that from a co-axial camera, but avoids the requirement for additional optical elements. The lens offset also typically provides a large enough baseline to be used in structured light calibration. As it is difficult to mount a camera to a moving mirror without adversely affecting the steering performance, in this case a separate pan and tilt camera must be used. This has the additional benefit of allowing the camera to be used separately from the projector, for example, by scanning an environment for objects or for activity detection. However, this approach does introduce registration and coordination problems between the projector display and camera view.

2. Camera Resolution

The spatial resolution of a camera is dependent upon the resolving power of the lens, the Field Of View (FOV) of the lens and the pixel resolution of the image sensor. Traditionally, the measure of resolution is described by the cameras resolving power, expressed in line pairs per millimetre (lp/mm). This is evaluated as the ability to distinguish distinct line patterns from standard test charts, such as that shown in Figure A.4 (centre).

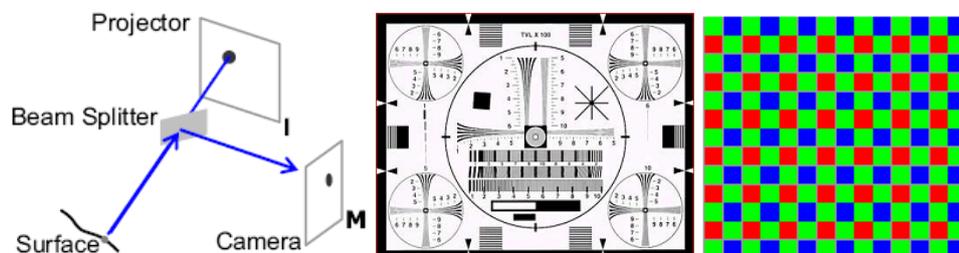


Figure A.4 (left) A Co-axial projector-camera system [Fujii, Grossberg et al. 2005] (centre) Camera Resolution Test Chart [Geoffrion 2005], (right) Bayer Pattern Colour Filter Array [Geoffrion 2005]

A higher resolution camera (i.e. higher quality lens, smaller lens FOV, or higher pixel resolution sensor) will allow detection of objects at greater distances and a generally “clearer” picture of the environment. However, large objects may be too big to fit in the frame when close to the camera with a smaller lens FOV, and increasing resolution with a higher pixel resolution sensor incurs an increased processing cost. For steerable projectors a pixel resolution of 640x480 pixels or higher is recommended, while the lens FOV required is dependant on the environment dimensions and detection scenario.

3. Video Frame Rates

Typical video frame rates are either 25fps for PAL video in Europe or 30fps for NTSC video in the USA and Canada. The use of cameras capable of video frame rates is important in dynamic environments, as it allows applications such as object tracking

and interaction detection based on movement. However, the actual frame rate achieved is both a function of the exposure length (long exposures for darker environments cause low frame-rates) and data transfer time to the computer. Similarly, while it is possible to purchase very high resolution machine vision cameras, due to bandwidth restrictions most are either only capable of very low frame-rates at full resolution or very low resolution at video frame rates. For example, interfaces such as USB 2.0 (480MB/s) and IEEE-1394 Firewire 400MB/s can only capture a resolution of 1280x1024 at a maximum of 27fps.

Commercial machine vision cameras present another solution to increase frame rates through binning pixels or region of interest limiting. Binning achieves a higher frame rate by allowing a sensor to capture information at full resolution then typically halving the resolution transferred to the computer by spatially averaging 4 pixels into one. Despite the averaging, this approach still gives a higher quality result than using a lower resolution sensor camera. In contrast, region of interest limiting allows a small rectangular sub-area of a CMOS camera sensor to be chosen for transfer to the computer, again limiting the final image resolution, but increasing the frame rate. The chosen area can also be dynamically shifted around on the sensor, allowing the potential for very high speed object tracking using small detection windows within the camera's field of view (e.g. 100fps for a 100x100 window). However, both these solutions typically require use of a camera manufacturer's proprietary software API.

4. Colour or Grayscale

Photodiodes in camera sensors are not sensitive to colour, only luminance. To enable colour sensitivity either one image sensor per colour is used or for single sensor cameras a colour filter array (CFA) is placed over the pixel array of the sensor. The most common CFA uses the Bayer pattern, which uses twice as many green filters as red or blue (as the human eye is more sensitive to green light), as shown in Figure A.4 (right). When processing the raw information provided by each photodiode, the camera analyzes each pixel colour value in a neighbourhood, spatially interpolates to create an approximation of the original full-colour image. However, due to this interpolation colour errors can be introduced and the effective image resolution reduced.

The use of a colour sensor is recommended though, as it allows object detection and tracking based on colour (which is a very useful cue, as shown in Chapter 5).

5. Dynamic Range

Photodiodes are only sensitive to a certain range of luminance, called the dynamic range (or contrast range). Anything outside this range is clipped to either pure black or pure white. Hence, a camera sensor capable of resolving a higher dynamic range will capture more information in an image. One factor affecting the dynamic range is the size of the camera sensor, as larger photodiodes can integrate light over a greater area and hence, capture greater differences in the total amount of light reaching the sensor. Consequently, cameras with small $\frac{1}{4}$ inch square sensors (such as cheap web cameras) will not exhibit as high a dynamic range as digital SLR cameras with one inch square sensors. Some commercial CMOS machine vision cameras also allow programmable sensitivity, where the camera has a non-linear response to light. Here a camera can be set to have a linear response in darker areas (or even a boosted response to increase shadow brightness) and a reduced response in highlight areas, allowing detail to be captured in a much larger dynamic range.

6. Optical Distortion

All optical imaging systems introduce distortion into a captured image. The most common distortions in lenses are barrel distortion or pincushion distortion (where straight lines appear curved) and chromatic aberration (where light of different colours converges at different spatial locations). Vignetting is possible in wider angle lenses, causing a noticeable darkening of the image towards its corners. Similarly, in colour digital imaging systems there is also the possibility a purple fringing of objects against high contrast backgrounds which is a combination of chromatic aberration and blooming on the camera sensor where the photodiodes are overloaded.

Many of these distortions can be modelled mathematically and corrections applied to the image, as discussed in related work section 2.6.4. Although higher quality optics cost more, they generally exhibit less distortion, hence require less calibration, modelling and correction.

7. Lens Field of View and Zoom

The Field Of View (FOV) of a camera lens will partially determine its spatial resolution, as described above. A larger FOV from wide-angle lenses allows a system to detect and track objects over a greater area, but at the cost of a lower spatial resolution.

Many cameras have fixed focal length lenses, requiring the lens to be physically replaced to change the FOV to fit the requirement of the environment – for example, using a wide-angle in small environments or telephoto lens when the steerable projector is located far from the objects of interest. One solution is to use a computer controlled zoom lens to extend its useful range. Zoom is beneficial as it allows a single lens to be used for both wide field of view, low spatial resolution applications (tracking of objects close to the camera), and small field of view, high spatial resolution applications (tracking objects at great distances). However, although flexible, typical powered zoom lenses are much larger and heavier than their fixed-focal length counterparts.

A.5 Commercial Steerable Projectors

Although there is a range of thousands of moving display lights, there are currently only four commercial steerable projector systems known to be in production.

These systems are all designed to be used as display lights; hence the majority do not include a camera with the system.

Table A.1 Four commercial steerable projectors

	<p>Publitec Beammover £9,500 for a 4100 Lumen steerable projector. (Based on the Sanyo XP-46 projector.) Publitec are based in Germany. http://www.beamover.com/</p>
---	--

	<p>High-end Digital Light Range DL.1 to DL.3 £15,000 for a 4100 Lumen steerable projector. (Based on the Sanyo XP-46 projector) DL.2 and DL.3 projectors include an attached camera (768x494 pixel resolution with powered focus and zoom) High-end are based in the USA. http://www.highend.com/products/digital_lighting/</p>
	<p>High-end "Orbital One" \$14,000 Steerable Mirror Head A "bolt-on" moving mirror attachment for projectors. http://www.highend.com/products/digital_lighting/orbitalhead.asp</p>
	<p>Active Vision System AV-4 Based on a 3000 Lumen projector. Active Vision are based in Japan. http://www.activevision.jp/english/index.html</p>

A.6 Construction of Steerable Projector Systems

An explicit list of goals and constraints for steerable projector systems was developed, based on the dimensions of the lab environment described in section 6.3 and the important characteristics identified in A2-A4:

- Fixed mounting location with field of view at least equivalent to a hemisphere around the steerable projector, able to project an interface on any surface specified in the area's volume.
- Fast positioning system, able to move a projected interface 180° in both pan and tilt axes in 1 second, or less.
- Stable positioning system
- Angular positioning accuracy of 0.2° or higher.
- Accurately and repeatably position projected interfaces to within 1cm of specified location in the area, at a range of 2.8m (the average distance from the centre of the area to the walls).
- Able to project a focussed image to all locations in the area, visible under normal room illumination levels.
- Able to capture colour images of the projected image at a minimum of 640x480 pixel resolution and video frame rates.
- Cost under £10,000.

These goals are a combination of hardware and software characteristics, which when combined were hoped would create a viable steerable projector system.

A.7 Steerable Projector Hardware

To achieve the goal, steerable projector components meeting the criteria above were specified and purchased. As part of the specification process, we used the characteristics identified in A2-A4:

A.7.1 Projector Selection

We required the projectors to be visible with typical office fluorescent lighting. A average brightness around 3000 Lumens was found to work well in everyday room environments by related work [Butz, Schneider et al. 2004; Ehnes, Hirota et al. 2004], however, due to the ambient light levels that would be encountered, only projectors with a contrast of 500:1 or greater were short-listed.

XGA projectors (1024x768 pixels) were selected as they are the most common resolution for business graphics and available at 3000 Lumens. Higher resolution projectors are much more expensive, and lower end models generally exhibit less than 2000 Lumens brightness.

Computer controlled focus and zoom capability was specified as a requirement due to the limited depth distance in which a projector image appears in focus. This allows projection on varying sizes of objects and at different distances.

The lens location was a major consideration, as it impacts the selection of the type of pan and tilt yoke for the moving head steerable projector system. There was only one projector with a centre lens and the required brightness and resolution specifications - the Sanyo PLC-XP45 projector. Whereas the choice of projectors with the lens offset was much greater, hence these were short-listed.

For moving head steerable projectors the projector weight is a major concern, as pan and tilt platforms have a limited maximum mass they can rotate. Current LCD projector optical designs trade-off between brightness and weight, however, DLP technology enables a smaller optical path, hence bright projectors in a small, light form factor.

The cost of a projector was limited to a maximum of £4,000 to allow for the purchase of other items such as the pan and tilt yoke and cabling.

At the time of purchase, two projectors met the specifications defined above. These specifications are summarised in Table A.2.

Table A.2 Projector Specifications

Projector	LCD / DLP	Brightness (Lumens)	Contrast	Resolution	Powered Zoom and Focus	Weight (kg)	Cost
NEC MT1065	LCD	3200	800:1	XGA	Yes	5.9	£3,500
Casio XJ-450	DLP	2800	1000:1	XGA	Yes	2.7	£3,000

Both projectors were eventually purchased, with the heavier NEC MT1065 used for a moving mirror steerable projector and the Casio XJ-450 used for a moving head design.

A.7.2 Steering Mechanism Selection

Both single and dual fork yokes (as shown in Figure A.5) have built in stepper motors to pan and tilt the moving head. The yokes are capable of absolute positioning via the DMX control protocol. The centre of rotation with a dual fork yoke is the centre of the yoke arms, whereas for the single arm yoke it is in-line with the central bearing. Borkowski et al. provide construction details for a dual fork pan and tilt mounting with an offset pan rotation location [Borkowski 2006], allowing offset lens projectors to be used. However, this requires workshop facilities and Borkowski admits with hindsight he would recommend purchasing off-the-shelf rather than building himself.

In contrast, while the single arm moving head solution allows offset lens projectors to be mounted, the single mounting point places a greater mechanical strain on the arm and tilt stepper motor. Consequently, the projector must be kept as light as possible.



Figure A.5 (left) Futurelight PHS150 Single arm Yoke, (right) Compulite "Luna" dual fork Yoke

The specifications of the single-arm Futurelight PHS-150 (purchased to be used with the Casio XJ-450 projector) are summarised in Table A.3.

Table A.3 Yoke Selection Specifications

Yoke	Field of View/ Coverage	Rotation Speed	Control Resolution	Angular Resolution	Size, Weight	Cost
Future-light PHS-150	540° Pan 265° Tilt	Not published	16-bit	0.008° Pan 0.004° Tilt	0.051m ³ 13kg	£570

A.7.3 Camera Selection

Industrial machine vision cameras offer a good performance and high quality output due to large sensors which gather a lot of light and high quality lenses with low optical distortion. For the moving head steerable projector we selected a PixeLink A742 colour CMOS camera with 1280x1024 pixel resolution, capable of 27 frames per second (fps) at full resolution and up to 104fps with a 640x480 region of interest enabled. The camera uses firewire for data transfer and has a fixed focal-length 12mm C-Mount lens with 40x30° field of view (FOV). For the demonstration applications in this work, the camera is used near its maximum aperture (f1.8) with the focus fixed at 2.5m. This gives an acceptable focus between 1.5m and 5m from the camera, while typically allowing exposure lengths under 40ms with the office lighting.

Additionally, a Logitech Sphere pan and tilt web camera was selected for use with a moving mirror system. This has 320x240 pixel resolution with a built-in lens, and is capable of 30fps. The computer controlled mechanical pan and tilt design has a 180° pan Field Of View (FOV) and 60° tilt FOV. No aperture or focus control is available.

A.8 Moving Head Steerable Projector Construction

Following purchase of the steerable projector hardware components, there were six steps required to create an operational moving-head steerable projector system:

1. Remove disco light head from the pan and tilt yoke
2. Remove the light control circuitry and wiring from the yoke arm and base
3. Re-wire the yoke with power, data and camera cables
4. Fabricate a bracket to attach the Projector and camera to the yoke
5. Attach the bracket and connect the cabling.
6. Mount the whole system in the space where it is intended to be used.

Re-wiring the Yoke

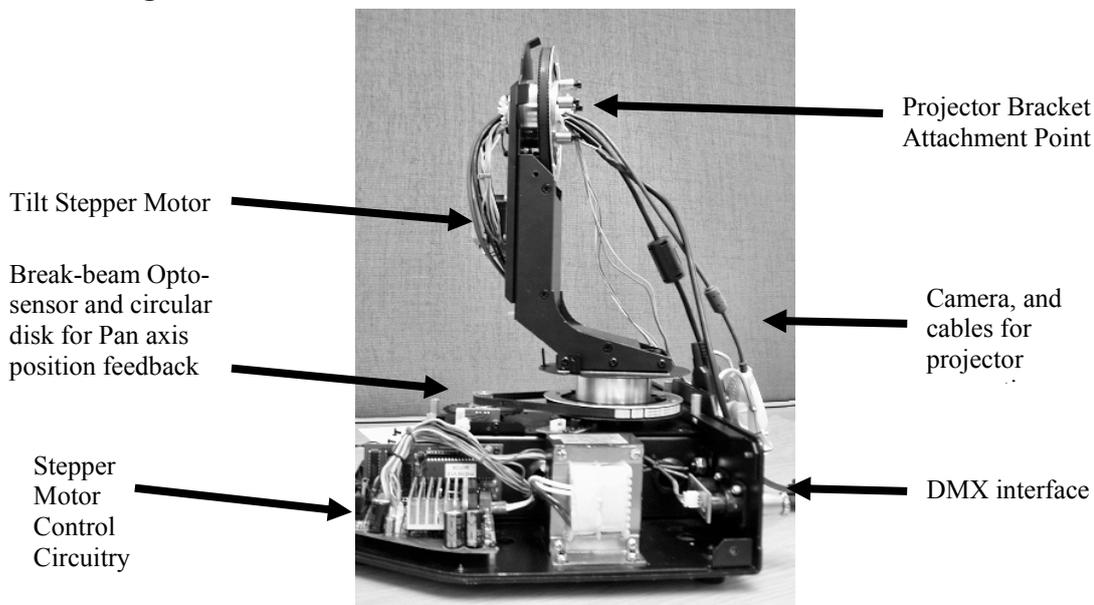


Figure A.6 The pan and tilt yoke uncovered

When purchased, the yoke had control wiring up from the base, through the arm, to the head for controlling the disco light lamp, colour filters and motors.

All control wiring for the head was removed with the exception of five wires, retained for controlling the projector. Power and data cables for the projector and the camera firewire cable were cut then threaded through the yoke body and arm in the place of the removed wiring. The cut cables were then re-soldered inside the yoke base. As there are no slip-rings for transferring the power and data through the rotating bearings, the cables pass through central holes in the bearings to the projector attachment point. Consequently, as the cables will twist as the yoke pans and tilts, a loose loop of each cable was left in the base of the unit to reduce the strain on large movements (i.e. winding-up). An overview of the steerable projector cabling can be seen in Figure A.7

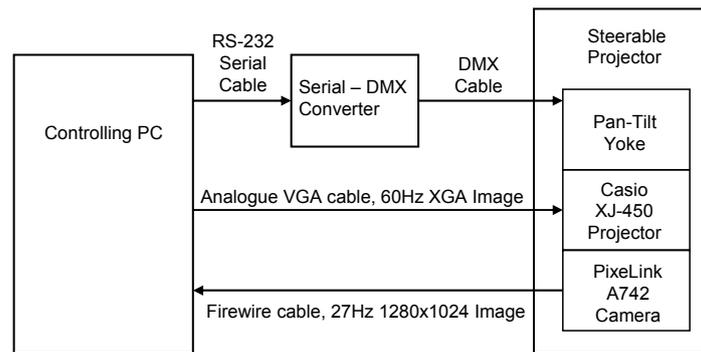


Figure A.7 Steerable Projector Cabling Layout

Projector Bracket Manufacture

The original pan and tilt yoke had the moving lamp head attached by five machine screws. When this was removed, a bracket was required to be fabricated to attach the projector in place of the light. Firstly a paper template of the underside of the projector was created, which had the projector mounting screw holes and cooling air intake holes marked. Then 3mm thick aluminium sheet was purchased and cut to size based on the template. Small holes were drilled in the bracket to aid airflow to the ventilation holes on the underside of the projector. The bracket side was then folded to an exact 90° angle using a metal forming machine in the Lancaster University Engineering Department.

Finally the bracket was attached to the pan and tilt yoke and the projector mounted to the bracket in its upside down ceiling project mode. An extra strip of aluminium was added running around and below the projector on which to mount the camera, and give the bracket more strength.

During mounted trials of the bracket and projector, the bracket appeared more flexible along the 90° fold line than was envisioned, so a second bracket was designed with a back plate to ensure the length of the bracket stayed at a 90° angle. The rear lip of the bracket was further turned up to provide increased stiffness and reduce the possibility of the projector weight bending the bracket down along its length.

This redesigned bracket performs better, however, the projector still appears to “droop” slightly, with the side of the projector furthest from the yoke attachment point a couple of millimetres below the attachment side. In practice, the slightly rotated projector image this causes is not particularly visible and this rotation does not affect object detection or projection accuracy due to the rigidly attached camera. A dual fork system would remove this problem altogether by supporting both ends of the projector equally.

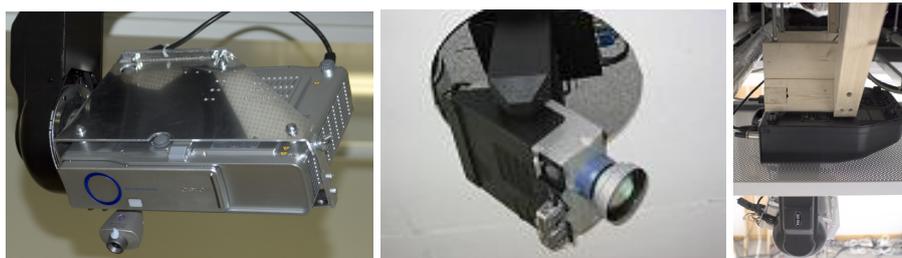


Figure A.8 (left) The fabricated projector bracket attached to the yoke, (centre) FLUIDUM room projector mounting direct to concrete [Butz, Schneider et al. 2004], (right) Wooden Bracing of the four threaded rods from which the projector is hung

Installation Site Constraints

As the projector was required to have a full 360 degree view of the lab environment, ceiling mounting in the centre of the area was the best mounting location. Two mounting possibilities for the Steerable Projector system presented themselves:

1. Mount the system directly on the concrete ceiling - which would be the most stable option,
2. Mount the system hung lower, so that only the moving arm hangs below the false ceiling.

The first option (while preferable for stability) would have caused the projector to be above the false ceiling, requiring a very large hole be cut in the ceiling to allow the projector clear line of site to the walls, so it was decided to mount with the body of the Pan and Tilt yoke just above the false ceiling, and cut a small neat hole in the tiles for the arm to hang down.

Typical mounting methods for pipes and ducting above false ceilings relies on attachments to threaded rods hanging down from the concrete ceiling, attached with shield anchors into the concrete, into which the threaded rod screws. This mounting method was also chosen for the steerable projector, with four rods – one at each corner of the pan/tilt base. The completed and mounted steerable projector can be seen in Figure A.9 (left).

Bracing

The projector and yoke are a dynamic system when the projector is moving, with projector inertia exerting a lot of force via the arm and stepper motor on the mountings. The steerable projector system approximates a heavy object on the end of long thin rods, acting similar to a pendulum. As the projector was mounted horizontally in landscape mode, when moving in the Pan axis a large torsional force is applied to the mountings due to a large portion of the projector weight being off-centre. Replacing the rods with a metal box-like structure with diagonal bracing struts would make the mounting more stable, but would require a commercial fabrication (most likely at great expense). Consequently, an attempt was made to brace the whole structure with wood to provide more stability and reduce movement of the mounting structure in response to projector movement, as shown in Figure A.8 (right).

A.8.1 Moving Head Control

The pan and tilt unit is controlled with the DMX512/1990 protocol. Up to 512 8-bit channels can be controlled from one DMX interface, or multiple channels can be combined for higher bit accuracy.

An RS-232 serial to DMX interface converter was purchased from Milford Instruments, which accepts two sequential byte values from the PC's serial port (i.e. channel number 0 to 255, channel value 0 to 255) for conversion to DMX. Data is sent over DMX using RS-485 signalling and cabling. DMX is transmitted serially at

250kbps and is grouped into packets of up to 513 bytes. A full packet takes approximately 23ms to send, corresponding to a 44Hz update rate. The purchased yoke has four 8-bit channels for controlling the pan and tilt, with one 8-bit channel for movement speed control. The positioning channels are split into high and low byte values, and re-combined at yoke to form an absolute 16-bit position value.

The DMX interface is a uni-directional control interface, hence there is no accurate position feedback or confirmation of commanded actions. This open-loop control system affects how software is written on the controlling PC, as it is not possible to simply issue a move command and assume the pan and tilt unit has moved instantaneously to the commanded position. Instead either an incremental approach must be made, with a timed control loop continuously sending small movement increments to try and reduce the commanded movement to physical position disparity, or the projector motion simulated in software using equations of motion, as discussed in section 6.2.3.

It would be very useful to have exact position feedback when performing large manoeuvres of the unit, during high speed manoeuvring or for better performance when tracking objects with the camera. This can be accomplished by adding rotary optical position encoders to the stepper motor spindles.



Figure A.9 (left) The operational moving head steerable projector system, (centre) The tripod-mounted moving mirror steerable projector system, (right) Close-up of operational moving mirror yoke and pan-tilt camera

A.9 Moving Mirror Steerable Projector Construction

A moving mirror steerable projector has a more limited optical design than moving head projectors, with a smaller Field Of View (FOV) and increased optical distortion caused by rotation in the plane of the image when the mirror is panned.

The closer to the projector lens, the smaller the mirror can be while still reflecting a full image. However, the closer to the projector lens the smaller the field of view as the projector itself blocks large areas that the mirror can point to. Hence, a balance must be found between mirror size, distance to projector and attainable field of view.

The motion mechanism used to move the mirror is required to rotate the mirror in the two dimensions of pan and tilt independently. Consequently, systems similar to the design of the dual fork yoke (above) are often used. By mounting a mirror inside the yoke, these designs typically reach 180° pan and 60° tilt FOV.

A first surface mirror (with silvering on the front, glass on the rear) can also be used for less distortion and to avoid the ghost images seen when using ordinary rear-surface mirrors. However, the front surfaces are extremely fragile as any contact with other objects can easily scratch the silvering (even when cleaning with a lens cloth).

The construction of the moving mirror steerable projector required seven steps:

1. Fabricate a bracket to attach the mirror pan and tilt yoke to the projector.
2. Fabricate a dual fork pan and tilt yoke incorporating pan and tilt servos.
3. Cut first-surface mirror to size, to allow pan rotation without fouling.
4. Mount mirror onto yoke.
5. Connect the projector, servo control and USB camera cables, as shown in Figure A.10.
6. Mount camera on projector.
7. Mount the whole system in the space where it is intended to be used.

The constructed system was mounted on a display light tripod, with the projector 1.8m from the ground, as seen in Figure A.9 (centre). This allows some portability, as the tripod can be easily moved around in the environment.

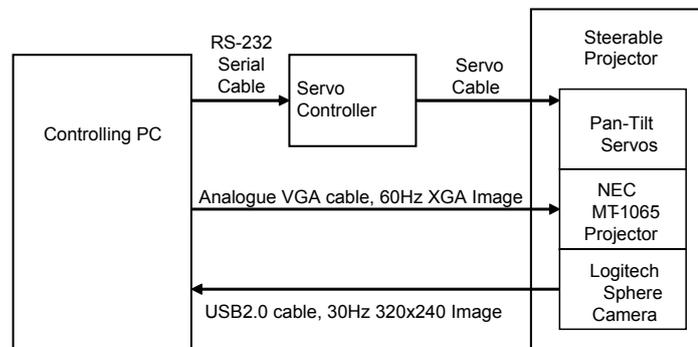


Figure A.10 Moving Mirror Steerable Projector Cabling

A.10 Characterisation of Moving Head Mechanism

The basic specifications of the moving head steerable projector components were known from published literature. However, following construction, the mechanical performance of the combined system was unknown. We use the moving head steerable projector for the experiments and demonstration applications presented in this thesis, hence, the steering mechanism and projector characteristics (A2-A3) are evaluated.

To characterise the moving head system a series of six experiments was performed to evaluate movement speed, field of view, positioning accuracy, positioning repeatability, positional stability and mechanism acceleration response.

A.10.1 Design

The **Movement Speed** evaluation consists of two experiments.

The yoke purchased has a possible 220 rotational speed settings, however, no published information could be found on the corresponding angular speed of rotation. Five separate maximum speed 360° rotations in the Pan axis and five separate 180° rotations in the Tilt axis were timed and mean averaged.

This experiment was repeated, but with the yoke set to minimum speed. The five recorded times were again mean averaged.

The addressable **Field of View (FOV)** was measured by moving the projector at minimum speed through five separate 360° rotations in the Pan axis and five separate 180° rotations in the Tilt axis, while measuring the difference in DMX control values.

The formula used to calculate addressable field of view is:

$$FOVRange(^{\circ}) = \left(\frac{\text{Total decimal DMX control range}}{\text{Measured difference in DMX control values}} \right) * \text{Angle moved} \quad (\text{A.1})$$

The **Positioning Accuracy** of the yoke was measured as angular resolution, calculated using the angular field of view measurements from above divided by the addressable DMX range.

To measure **Positioning Repeatability** three separate pan and tilt values were chosen, equivalent to aiming the projector at the centre of the North, East and South walls in the Experimental Systems Lab. A white cross was projected in the centre of the projectors image and lines drawn on the three walls equivalent to the cross.

The projector was then moved five times 90° in a random pan direction and tilted 180° down (could not tilt up, as the yoke cannot physically move this far), then returned to the original values aimed at the centre of the wall.

The **Positional Stability** was measured as the time it took for the whole steerable projector image to cease moving following arrival of the yoke at the commanded position after a 180° rotation (chosen to allow the yoke time to reach full speed). Ideally this figure would be close to zero, which would indicate a stable system with stiff mountings.

Five 180° pans were performed at the yoke's slowest movement speed, and the time taken for movement to cease on arrival at a pre-marked location was recorded. Similarly, five 180° tilt movements were performed at the yoke's slowest speed.

The **Mechanism Acceleration Response** is difficult to measure without position feedback or mounting accelerometers on the hardware. Similarly, inertia is difficult to calculate, due to the difficulties integrating the masses of the projector, bracket and arm in three dimensions. Hence, the projector movement and structural response was observed visually during 90°, 180° and 360° pan rotations and 180° tilt rotation for the characterisations above.

A.10.2 Procedure

For measurements of 90°, 180° or 360° rotations around the Pan axis, the following procedure was used:

1. The projector was used to project a large cross 1 pixel wide in the centre of its display.
2. The projector was rotated until horizontal, and perpendicular to centre of the South wall for 180 and 360° movements, or East wall for 90° moves.
3. The initial DMX setting was noted
4. The wall was marked in pencil with a line at the location of the vertical line of the projected cross.

5. A destination line was drawn in the centre of another wall at the required rotation. No mark was required for 360° movements.
6. The projector was rotated until the projected line was aligned with the destination mark
7. The final DMX setting was noted.

For measurements of 180° rotations around the Tilt axis, the following procedure was used:

1. The projector was used to project a large cross 1 pixel wide in the centre of its display.
2. The projector was rotated until horizontal, and perpendicular to the South wall.
3. The initial DMX setting was noted.
4. The height from the ceiling to the horizontal line of the cross was measured.
5. The North wall opposite was marked with a pencil line at the same height below the ceiling (we assume the ceiling is horizontal).
6. The projector was rotated until it had moved through a full 180° and the projected line was aligned with the mark on the North wall.
7. The final DMX setting was noted.

A.10.3 Results

Table A.4 Moving Head Steerable Projector Characterisation

Characteristic	Result
Minimum Movement Speed	19°/s Pan 13°/s Tilt
Maximum Movement Speed	180°/s Pan 180°/s Tilt
Field of View	539.5° Pan 253° Tilt
Positioning Accuracy (Angular Resolution)	0.016° Pan 0.004° Tilt
Positioning Repeatability	<1mm error at 2.84m
Positional Stability For Pan	6s at min speed 5s at mid-speed 5s at max speed
Positional Stability For Tilt	<1s at min speed <1s at mid-speed 1s at max speed

A.10.4 Discussion

Field of View / Coverage

Although the published FOV values were 540° Pan and 265° Tilt, the mean average of these calculated values was: 539.5° and 253.0° respectively (to the nearest 0.5°), as can be seen in Table A.4.

Positioning Accuracy

The yoke control system maps a 0° rotation to a 16-bit DMX control value of 0 high byte, 0 low byte. In the pan axis the measured 539.5° FOV is mapped to values of 255 high byte, 255 low byte, while in the tilt axis it is the measured 253.0° mapped to values of 255 high byte, 255 low byte.

While performing this characterisation of the positioning accuracy, it was discovered that, in reality, the claimed 16-bit Pan positioning accuracy of the yoke only resolved physically to 15-bits. On the low byte values, a decimal value greater than 131 overlaps into the next high byte range. For example, the position of the yoke effectively moves from 0,131 to 1,0 instead of 0,132.

Positional Stability

As shown in Table A.4, the slowest pan speed this was found to be an average of 6 seconds, due to the projector inertia causing oscillation in the steering arm and mountings. However, interestingly, the time required for stabilisation decreased if the yoke was panned faster. This suggests there may be either some structural resonance created with the slower speed pans, or the pan stepper motor is having trouble with overshoot and is hunting for the correct position. However, for tilt there was little oscillation at any movement speed. In fact, at the slowest and half tilt speeds the stabilisation time could not be accurately measured as it was less than a second.

Mechanism Acceleration, Inertia and Response Time

By subjective observation of the system, the current mounting of the projector appears to exhibit high inertia, as there is a large structural response exhibited to start and stop movement events in pan. This is reflected in the positional stability results as the long time taken for the projector to stop oscillating following a pan movement.

A.11 Characterisation of Projector

Although the published specifications give most relevant detail about a model of projector, it is possible for individual projectors to vary due to manufacturing tolerances. Consequently the purchased projector was characterised.

Useful zoom range

Printed text typically has a resolution between 150 and 600 dots per inch (dpi), whereas a computer monitor is around 72 dpi. With a 1024x768 pixel resolution projector, to project a 72dpi image the final image size would be a tiny 14.7 by 10.7 inch image. However, projectors are used primarily for their ability to display large images, rather than high resolution images.

Acceptable resolution depends on viewing distance, with displays viewed from larger distances subtending a smaller angle in the eye, hence subjectively appearing higher resolution than if the same display was viewed at close range. Ways to boost resolution of projected displays (apart from using a higher resolution projector) by using multiple superimposed projected displays and super-resolution techniques are demonstrated by Majumder and Welch [Majumder and Welch 2001].

The Casio XJ-450 projector has power zoom range settings of 0-40. When projecting perpendicular to a display surface at a distance of 2.89m, zoom settings above 20 produced images larger than 1.3 by 1m, but at low resolution with pixel sizes greater than 1mm^2 . This pixelation was very visible and occasionally distracting when viewed from 5m distance. As the majority of displays would be viewed from less than 5m distance in the demonstrations in Chapter 7, it was decided to limit the projector to zoom settings of 20 and below for interactive use. This provides a reasonable balance in the trade-off of keeping projected pixel sizes as small as possible (hence, display dpi as high as possible) with maximising projection area.

Display Size versus Distance

It is useful to know what size and resolution can be expected from a display at what distance, as this will determine what size objects can be projected on. Two zoom settings were measured – zoom 0 (smallest image possible) and zoom 20 (largest image). The width and height of an image displayed on a planar surface perpendicular to the projector was measured at a number of distances for both zoom settings. The results shown in Figure A.11 can be used in our Cooperative Augmentation architecture to calculate which objects are in the projector FOV, as discussed in section 6.3.

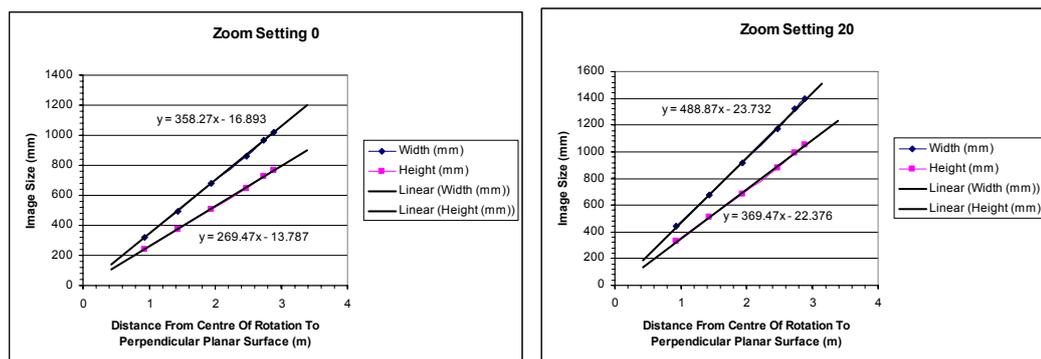


Figure A.11 (left) Image Size versus Distance at Zoom 0, (right) Image Size versus Distance at Zoom 20 (mid-zoom)

Image Resolution at the centre of a display perpendicular to the projector was also measured at varying distances for both the zoom settings, as shown in Figure A.12.

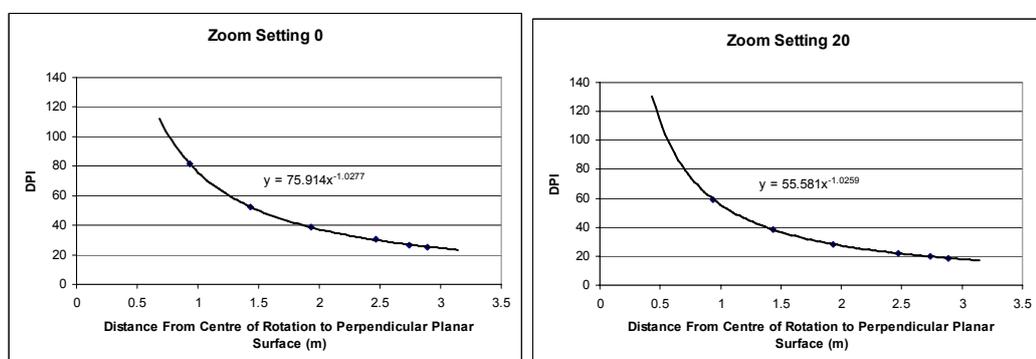


Figure A.12 (left) Display resolution versus distance to projection surface at Zoom 0, (right) Display resolution versus distance to projection surface at Zoom 20

Focus versus Distance

Most video projectors have a good depth of field. It was found empirically that the Casio XJ-450 has an approximate 1m depth of field, within which an acceptably focussed display will appear.

For display on the walls of the eastern end of the experimental systems lab the focus can be set to a distance of 3m and it will appear acceptably in focus everywhere except in the very corners of the area. However, for applications which are required to track moving objects or project on very oblique surfaces in front of the walls, dynamic focussing is required.

The projector focus settings were measured, again at the two zoom settings (zoom 0 and 20). The results shown in Figure A.13 are used in our Cooperative Augmentation architecture for dynamic focus control, as discussed in section 6.3.2.

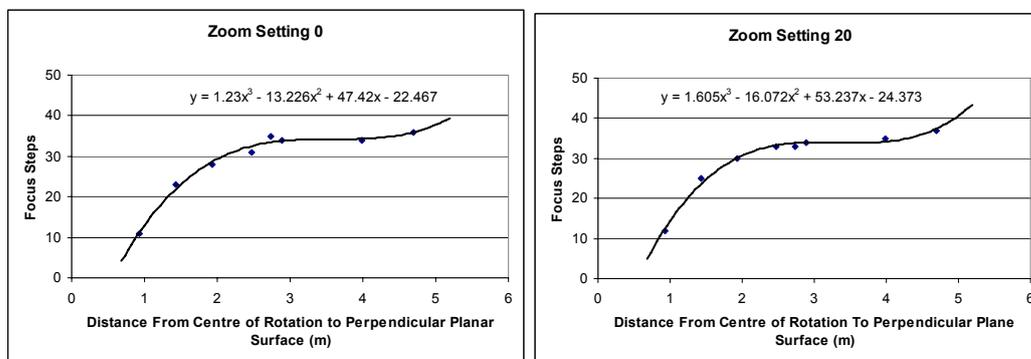


Figure A.13 (left) Focus steps versus distance to display surface at Zoom 0, (right) Focus steps versus distance to display surface at Zoom 20

A.12 Steerable Projector System Comparison

The constructed moving head steerable projector system was compared to other steerable projector-camera systems known to exist [Pinhanez 2001; Yang, Gotz et al. 2001; Borkowski, Riff et al. 2003; Butz, Schneider et al. 2004; Ehnes, Hirota et al. 2004]. However, many steerable projector systems do not have published performance measurements; hence characteristics have only been compared where there is data available on a minimum of two systems. As can be seen in comparison Table A.5 to Table A.7, the three aspects of steering mechanism, projector and camera were all compared.

For the steering mechanism, our pan and tilt yoke compares favourably with other systems. The greater pan FOV has benefits, as it potentially allows object tracking beyond the 360° position without the requirement for “unwinding”. However, it is this large range combined with the reduced 15-bit Pan positioning accuracy (discussed in A.9.5) which gives a much lower pan angular resolution than the FLUIDUM and Hirose Lab systems.

Table A.5 Steerable Projector Steering Mechanism Comparison

	Lancaster	IBM	FLUIDUM, Saarland + Munich	INRIA, Grenoble	Hirose Lab, Japan	PixelFlex, UNC
Steering Method	Moving Head	Moving Mirror	Moving Head	Moving Head	Moving Head	Moving Mirror
Construction	DIY	DIY	Purchased Publitec Beammover	DIY	Purchased Active Vision AV-4	DIY
Field of View	540° Pan 265° Tilt	230° Pan 50° Tilt	333° Pan 270° Tilt	354° Pan 90° Tilt	360° Pan 240° Tilt	Not published
Maximum Speed	180°/s Pan 180°/s Tilt	Not published	Not published	146°/s Pan 80°/s Tilt	80°/s Pan 80°/s Tilt	Not published
Positioning Accuracy (Angular Resolution)	0.016° Pan 0.004° Tilt	Not published	0.005° Pan 0.004° Tilt	0.11° Pan 0.18° Tilt	0.01° Pan 0.01° Tilt	Not published
Mounting	Fixed, Ceiling	Portable or Fixed	Fixed, Ceiling	Fixed, Ceiling	Portable, Desktop	Fixed, Ceiling
Size (Volume)	0.051m ³	Not published	Not published	Not published	0.13 m ³	Not published
Weight	13kg	Not published	Not published	Not published	32kg	Not published

On paper the INRIA steering system does not appear particularly impressive, with its relatively low speed and effective 12-bit pan and 9-bit tilt positioning accuracy. However, the applications demonstrated by Borkowski et al. [Borkowski 2006] and associated videos show it to be a very capable system. This suggests that a viable steerable mechanism need not be faster or more accurate than the INRIA system when used for relatively close range projection applications.

The Hirose Lab AV-4 system was classified as portable as Ehnes et al. demonstrated applications with the steerable projector sat on the desktop [Ehnes, Hirota et al. 2004], despite this, their system is more than twice as heavy as the Lancaster system, so portability is subjective in this case.

Table A.6 Steerable Projector Video Projector Comparison

	Lancaster	IBM	FLUIDUM, Saarland + Munich	INRIA, Grenoble	Hirose Lab, Japan	PixelFlex, UNC
Projector	Casio XJ-450	Sharp XG-P10	Sanyo PLC-XP45	Epson Powerlite 730c	Not published	Proxima DP6850
Brightness	2800	3000	3300	2000	3000	1500
Contrast	1000:1	250:1	800:1	400:1	Not published	100:1
Resolution	XGA	XGA	XGA	XGA	XGA	XGA
Weight	2.4kg	7.6kg	8.4kg	2.0kg	Not published	6.0kg
Auto Focus and Zoom	Yes	Yes	Yes	No	Yes	Yes

It can be seen in Table A.6 that the projector brightness, contrast and weight in steerable systems varies considerably. All systems have been built and installed in relatively small room-sized environments with (theoretically) controllable or limited

natural light. Consequently the INRIA and PixelFlex systems should not be handicapped at all by their lower brightness. In contrast, while the Lancaster projector is suitable for use, it could still benefit from higher brightness, as sunny days cause the projector display to be washed-out and difficult to read in the lab environment.

However, increasing brightness would increase projector weight and require a new pan and tilt yoke, or a change to a Moving Mirror steering mechanism.

When comparing the camera characteristics in Table A.7, it can be seen that while the Lancaster camera system offers a higher resolution to other systems, it is limited to a maximum frame rate of 27fps. All other systems are capable of 30fps with exception of the INRIA system which uses a 25fps PAL camera. Although the FLUIDUM camera is capable of very high resolution still images, it is handicapped for dynamic applications by the low 320x240 video resolution. However, this low resolution is offset somewhat by the 3x optical zoom present on the camera, which allows dynamic zooming to achieve reasonable video performance for small field of view applications (such as projected button interaction detection).

The PixelFlex camera was only used as a room camera, with no steerability. However, it was positioned so that it had a full view of the possible projection cones of all the steerable projectors in the multi-projector array [Yang, Gotz et al. 2001].

Table A.7 Steerable Projector Camera Comparison

	Lancaster	IBM	FLUIDUM, Saarland + Munich	INRIA, Grenoble	Hirose Lab, Japan	PixelFlex, UNC North Carolina
Camera	PixeLink A742	Sony EVI- D100	Canon S40 Digital Camera	Not published	Sony DFW- VL500	Not published
Resolution	1280x1024	768x494 NTSC	4 Megapixel	720x576 PAL	640x480	720x480 NTSC
Zoom	No	Yes	Yes	Yes	Yes	No
Video Capable	Max 27fps	Yes	Only at 320x240	Yes	Yes	Yes
Steerable Projector Mounted	Yes	No, pan and tilt camera	Yes	Yes	Yes	No
Used with Room Cameras	No	Yes (User following displays)	No	Yes	No	This was a room camera

A zoom capability on the Lancaster camera would also increase flexibility, allowing both a close-up view for interaction detection with projected images, and a large FOV for tracking of large objects in the lab area. Currently the projector image only covers a small portion of the whole image, further reducing the useable resolution of the image.

A.13 Conclusion

Two steerable projector-camera systems were successfully constructed – a moving mirror and a moving head system. In this chapter we identified characteristics important

to consider when purchasing or constructing a steerable projector-camera system. Finally, we evaluated the characteristics of the steering mechanism and projector in the constructed moving head system. This hardware and steerable system is used for the experiments and demonstration applications presented in this thesis.

Appendix B **Smart Object Programming Examples**

The state machine model used in the smart object enables a developer to define both relationships between objects and the states of objects in which events should occur. The three examples below illustrate the consideration required when programming the state machine to use object input modalities defined in section 6.2.4 for interaction:

B.1 Rough Handling Detection

Objects in a warehouse use embedded accelerometers to detect rough handling and warn employees of possible damage. If rough handling is detected they request a projection on their surfaces asking an employee to check them for damage. In this case objects can use either the 1st input modality (location and orientation) or 4th modality (other physical manipulation of object) to determine when to project the message. The programmer must decide here whether to use camera based location information which may less accurately detect rough handling and is only available when the object is visible, or embedded sensors such as accelerometers which may be more accurate at detecting rough handling, but consume more power.

Figure B.1 shows a diagrammatical representation of this program, designed using 4 states, an embedded accelerometer to detect rough handling and 2 projections. An employee is required to confirm they have seen the warning message and checked the object for damage in the second projection by clicking an interactive button. The typical state transition sequence is shown by the arrows; however, the state can change to “During Rough Handling” at any point as the limits are set to include all possible values for the button and projection variables. To test for state transition, in this application the result of all sensor operations are combined using boolean AND.

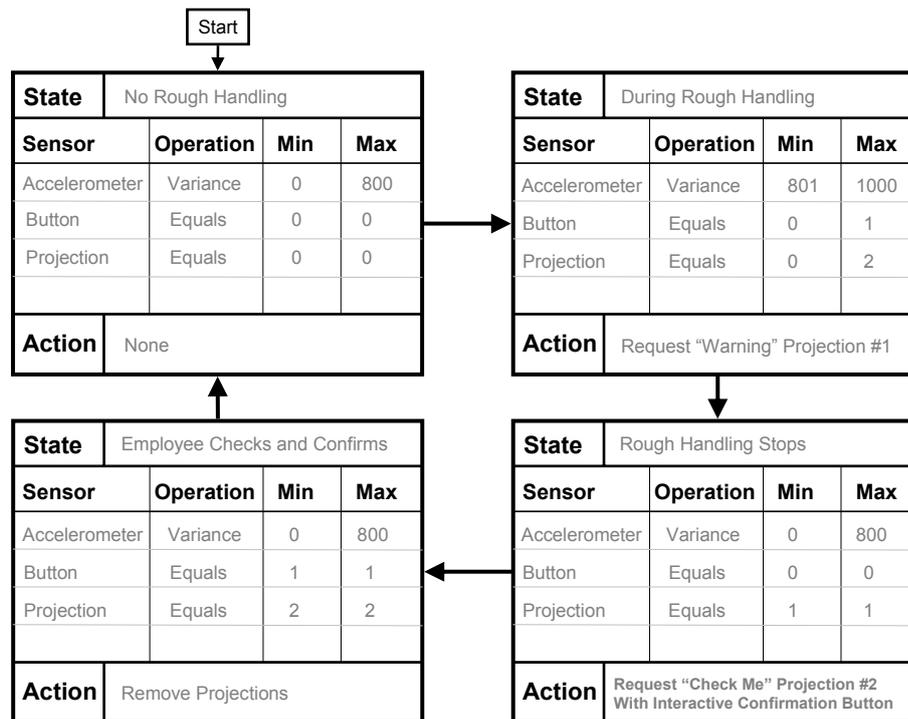


Figure B.1 State Machine Program for Detecting Rough Handling of a smart object

B.2 Smart Book

A smart book object allows mixing of traditional fixed printed media with dynamic projected content augmentations at runtime. The book itself is articulated and can be physically opened. Either force sensors or simple binary contact sensors (modelled as a switch) on each page sense when it is opened and which page is open. In this case, when the book is opened, it changes in both appearance and geometry. Tracking while object geometries change is a significant challenge for traditional vision based detection systems. However, using the 2nd input modality (manipulating the geometry) the object detects each page opening event and automatically updates the projector-camera system with a new appearance and geometry, so that tracking can continue uninterrupted.

Figure B.2 shows a diagrammatical representation of this simple program. In this case force sensors are used on each page, and a simple boolean AND combines the sensor information and decides which state to enter. For a book the typical state transitions are either being opened or closed at a random page, or flipping forwards or backwards through the pages. These typical transitions are again shown by the arrows.

This type of smart book program is typical of many other objects which either have a single sensor, or large number of the same sensor. For example, the smart cup [Gellersen, Beigl et al. 1999] contains a single temperature sensor. If we want to display this temperature on the cup itself we could use a similar program, where each value range of temperature (for example, each degree) is a separate state.

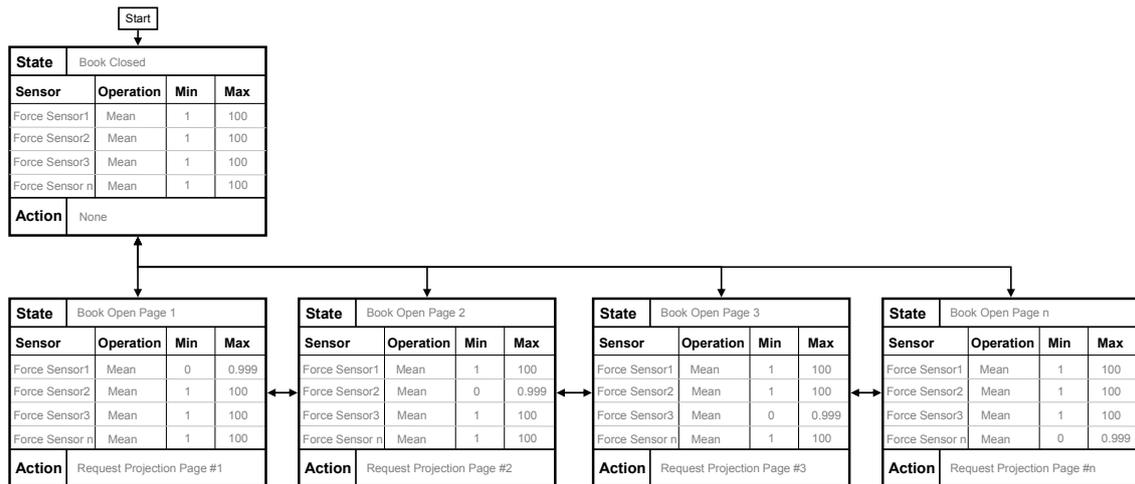


Figure B.2 State Machine Program for Articulated Smart Book with Force Sensors on Each Page

B.3 Smart Furniture Assembly

The smart furniture described in section 2.2.2 has embedded sensing to guide the purchaser with the task of assembly [Antifakos, Michahelles et al. 2002; Holmquist, Gellersen et al. 2004]. For this example we replace the signalling LEDs with projected displays to notify the user of the next task in the assembly sequence. A message about how to assemble each piece is projected in the right order when they are moved together into the same location. Binary contact sensors (modelled as a switch) detect when each part is correctly assembled. In this example each object uses the 1st (location and orientation) 4th (other sensors on an object) and 7th input modalities (requesting information about other objects) to calculate its location and orientation relative to other objects and determine with its sensors whether it is correctly assembled. Relative location detection is possible as all objects share the common world coordinate system.

The objects request projections to guide the user in sequential assembly based on the pre-defined series of states shown in Figure B.3. The program requires that the later assembled objects have explicit knowledge of the identity and sensor values of earlier objects, to ensure correct assembly sequence by monitoring their sensors. For example, object 3 can monitor the switch states on object 1 and 2 to know when they are assembled and hence, when to project its display. However, in constrained scenarios such as this the identity and capabilities of each part are fixed and known a-priori by a designer; hence this requirement is not a limitation.

In this case our furniture consists of three separate pieces which need to be assembled in sequential order. These separate pieces are each a smart object and use a simple boolean AND to combine the sensor information when deciding which state to enter. The typical state transitions are again shown by the arrows.

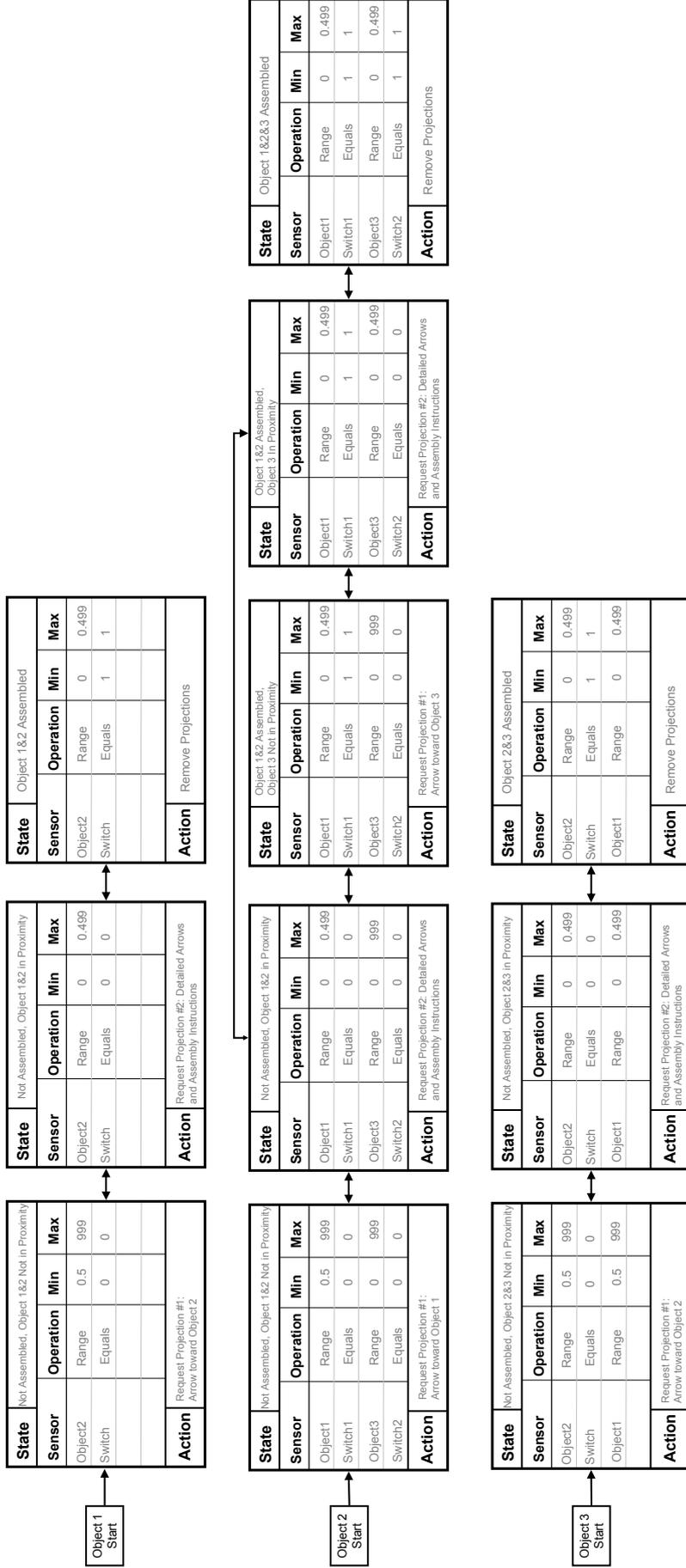


Figure B.3 Smart Furniture Assembly Program for 3 Individual Pieces

B.4 Smart Cooking Object Model

This section accompanies the Smart Cooking demonstration presented in Chapter 7. Here we present the content model states for each of the objects (egg box, pan, salt and stove) grouped by the recipe state variable and arranged in recipe state order 0-12.

0				
Egg Box				
				
State	1 - Closed			
Sensor	Operation	Min	Max	
Recipe	Value	0	0	
Light	Value	0	2	
Combination Method		AND		
Action	Project "Add egg to pan"			
Pan				
				
State	1 – Empty			
Sensor	Operation	Min	Max	
Recipe	Value	0	0	
Force	Value	2	29g	
Combination Method		AND		
Action	Project "Add egg to pan"			
State	2 - Open			
Sensor	Operation	Min	Max	
Recipe	Value	0	0	
Light	Value	2	9999	
Combination Method		AND		
Action	Project "Add egg to pan"			
State	2 – Egg Added			
Sensor	Operation	Min	Max	
Recipe	Value	0	0	
Force	Value	30g	9999	
Combination Method		AND		
Action	Remove Projection, Send 1			
1				
State	3 – Finished with Eggs			
Sensor	Operation	Min	Max	
Recipe	Value	1	1	
Combination Method		AND		
Action	Remove Projection			
State	3 – Add water to pan			
Sensor	Operation	Min	Max	
Recipe	Value	1	1	
Force	Value	30g	9999	
Combination Method		AND		
Action	Project "Add water to pan"			
State	4 – Place on stove			
Sensor	Operation	Min	Max	
Recipe	Value	1	1	
Force	Value	0	1	
Temp	Value	0	15°	
Combination Method		AND		
Action	Project place on stove, Send 2			
2				
State	5 – On Stove			
Sensor	Operation	Min	Max	
Recipe	Value	2	2	
Force	Value	30g	9999	
Object Stove	Distance	0	0.25m	
Combination Method		AND		
Action	Remove Projection, Send 3			

Figure B.4 Hard-Boiled Egg Recipe States 0-2

Pan



Salt



3

State	6 – Add Salt to Pan			
Sensor	Operation	Min	Max	
Recipe	Value	3	3	
Force	Value	30g	9999	
Object Stove	Distance	0	0.25m	
Combination Method		AND		
Action	Project “Add Pinch of Salt”			

State	1 – Add Salt, Salt Static			
Sensor	Operation	Min	Max	
Recipe	Value	3	3	
Force	Value	10	250g	
Object Pan	Distance	0.25m	9999	
Combination Method		AND		
Action	Project “Add Pinch of Salt”			

State	2 – Add Salt, Salt Mobile			
Sensor	Operation	Min	Max	
Recipe	Value	3	3	
Force	Value	0	1	
Object Pan	Distance	0	0.25m	
Combination Method		AND		
Action	Project “Add Pinch of Salt”, Send 4			

4

State	3 – Salt Added, Salt Static			
Sensor	Operation	Min	Max	
Recipe	Value	4	4	
Force	Value	10	250g	
Combination Method		AND		
Action	Remove Projection, Send 5			

5

Stove



State	1 – Turn to 100% setting			
Sensor	Operation	Min	Max	
Recipe	Value	5	5	
Gas	Value	0	99%	
Combination Method		AND		
Action	Project Turn on, 100% setting			

State	7 – Salt Added			
Sensor	Operation	Min	Max	
Recipe	Value	5	5	
Combination Method		AND		
Action	Remove Projection			

State	2 – At 100% setting			
Sensor	Operation	Min	Max	
Recipe	Value	5	5	
Gas	Value	100%	100%	
Combination Method		AND		
Action	Remove Projection, Send 6			

Figure B.5 Hard-Boiled Egg Recipe States 3-5

6

Pan



State	8 – Boil Water		
Sensor	Operation	Min	Max
Recipe	Value	6	6
Force	Value	30g	9999
Object Stove	Distance	0	0.25m
Combination Method		AND	
Action	Project Temp X, Wait for Boil		

Stove



State	9 – Boiling Water		
Sensor	Operation	Min	Max
Recipe	Value	6	6
Force	Value	30g	9999
Object Stove	Distance	0	0.25m
Temp	Value	100	100°
Combination Method		AND	
Action	Project Temp X, Send 7		

7

State	3 – Turn to 40% setting		
Sensor	Operation	Min	Max
Recipe	Value	7	7
Gas	Value	0	30%
Gas	Value	50	100%
Combination Method		(1 AND 2) OR 3	
Action	Project Turn to simmer (40%)		

State	4 – Around 40% setting		
Sensor	Operation	Min	Max
Recipe	Value	7	7
Gas	Value	31	49%
Combination Method		AND	
Action	Remove Projection, Send 8		

8

State	5 – Pan Too Cold		
Sensor	Operation	Min	Max
Recipe	Value	8	8
Object Pan	Temp	0	90°
Combination Method		AND	
Action	Project Pan too cold, adjust to simmer		

State	10 – Simmering 7 mins		
Sensor	Operation	Min	Max
Recipe	Value	8	8
Force	Value	30g	9999
Object Stove	Distance	0	0.25m
Time	Value	0	7min
Combination Method		AND	
Action	Project countdown timer		

Figure B.6 Hard-Boiled Egg Recipe States 6-8

8(continued)

Stove



State	6 – Pan Too Hot		
Sensor	Operation	Min	Max
Recipe	Value	8	8
Object Pan	Temp	99	9999
Combination Method		AND	
Action	Project Pan too hot, adjust to simmer		

State	7 – Pan Correct Temp		
Sensor	Operation	Min	Max
Recipe	Value	8	8
Object Pan	Temp	91	98°
Combination Method		AND	
Action	Remove Projection		

Pan



State	11 – Simmering 7 mins		
Sensor	Operation	Min	Max
Recipe	Value	8	8
Force	Value	30g	9999
Object Stove	Distance	0	0.25m
Time	Value	7min	7min
Combination Method		AND	
Action	Remove Projection, Send 9		

9

State	8 – Turn Off Stove		
Sensor	Operation	Min	Max
Recipe	Value	9	9
Gas	Value	1	100%
Combination Method		AND	
Action	Project Turn Off Stove		

State	9 – Stove Off		
Sensor	Operation	Min	Max
Recipe	Value	9	9
Gas	Value	0	0%
Combination Method		AND	
Action	Remove Projection, Send 10		

10

State	12 – Empty Water In Sink		
Sensor	Operation	Min	Max
Recipe	Value	10	10
Force	Value	30g	9999
Object Stove	Distance	0	0.25m
Combination Method		AND	
Action	Project Empty Water In Sink		

Figure B.7 Hard-Boiled Egg Recipe States 8-10

10 (continued)

Pan



State	13 – Emptying Water		
Sensor	Operation	Min	Max
Recipe	Value	10	10
Force	Value	0	1
Object Stove	Distance	0.25m	9999
Object Sink	Distance	0	0.25m
Combination Method		AND	
Action	Project Empty Water In Sink, Send 11		

11

State	14 – Water Emptied		
Sensor	Operation	Min	Max
Recipe	Value	11	11
Force	Value	0	1
Object Sink	Distance	0	0.25m
Temp	Value	0	70°
Combination Method		AND	
Action	Project "Enjoy Your Meal", Send 12		

12

State	15 – End		
Sensor	Operation	Min	Max
Recipe	Value	12	12
Force	Value	30	9999
Combination Method		AND	
Action	Remove Projection		

Figure B.8 Hard-Boiled Egg Recipe States 10-12