Tactile Mesh Saliency

Manfred Lau[1] Kapil Dev[1] Weiqi Shi[2] Julie Dorsey[2] Holly Rushmeier[2]
[1]Lancaster University [2]Yale University

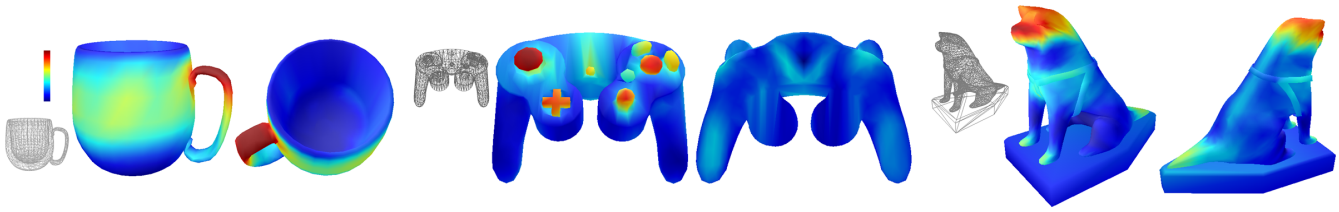

Figure 1: *Three examples of input 3D mesh and tactile saliency map (two views each) computed by our approach. Left: "Grasp" saliency map of a mug model. Middle: "Press" saliency map of a game controller model. Right: "Touch" saliency map of a statue model. The blue to red colors (jet colormap) correspond to relative saliency values where red is most salient.*

Abstract

While the concept of visual saliency has been previously explored in the areas of mesh and image processing, saliency detection also applies to other sensory stimuli. In this paper, we explore the problem of tactile mesh saliency, where we define salient points on a virtual mesh as those that a human is more likely to grasp, press, or touch if the mesh were a real-world object. We solve the problem of taking as input a 3D mesh and computing the relative tactile saliency of every mesh vertex. Since it is difficult to manually define a tactile saliency measure, we introduce a crowdsourcing and learning framework. It is typically easy for humans to provide relative rankings of saliency between vertices rather than absolute values. We thereby collect crowdsourced data of such relative rankings and take a learning-to-rank approach. We develop a new formulation to combine deep learning and learning-to-rank methods to compute a tactile saliency measure. We demonstrate our framework with a variety of 3D meshes and various applications including material suggestion for rendering and fabrication.

Keywords: saliency, deep learning, perception, crowdsourcing, fabrication material suggestion

Concepts: •**Computing methodologies** → **Shape modeling;**

1 Introduction

In recent years, the field of geometry processing has developed tools to analyze 3D shapes both in the virtual world and for fabrication into the real-world [Bächer et al. 2012; Hildebrand et al. 2013; Prévost et al. 2013; Zimmer et al. 2014]. An important aspect of a geometric shape is its saliency, which are features that are more pronounced or significant especially when comparing regions of the shape relative to their neighbors. The concept of *visual saliency* has been well studied in image processing [Itti et al. 1998; Bylinskii et al. 2015]. "Mesh Saliency" [Lee et al. 2005] is a closely related work that explores visual saliency for 3D meshes. However, other sensory stimuli have not been explored for mesh saliency. In this paper, we introduce the concept of *tactile mesh saliency*. We bring the problem of mesh saliency from the modality of visual appearances to tactile interactions. We imagine a virtual 3D model as a real-world object and consider its tactile characteristics.

There are many potential applications in graphics for mappings of tactile salience. In the virtual domain, tactile saliency can be applied to rendering appearance effects. A map of tactile salience enables the prediction of appearance that is the result of human interaction with an object. In the physical domain, tactile saliency information can be used to fabricate physical objects such that a surface may be enhanced to facilitate likely interactions.

We consider points on a virtual mesh to be tactile salient if they are likely to be grasped, pressed, or touched by a human hand. For our concept of tactile saliency, the human does not directly interact with real objects, but considers virtual meshes as if they were real objects and perceives how he/she will interact with them. We focus on a subset of three tactile interactions: grasp (*specifically for grasping to pick up an object*), press, and touch (*specifically for touching of statues*). For example, we may grasp the handle of a cup to pick it up, press the buttons on a mobile device, and touch a statue as a respectful gesture. Previous work explored the idea of touch saliency of 2D images on mobile devices [Xu et al. 2012]. The ideas of grasp synthesis for robots [Sahbani et al. 2012] and generation of robotic grasping locations [Varadarajan et al. 2012] have also been explored in previous work. However, the existing work in these areas solve different problems and have different applications. The problem we solve in this paper is to take an input 3D mesh and compute the relative tactile saliency of all vertices on the mesh.

We take a crowdsourcing and learning approach to solve our problem. This mimics a top-down or memory-dependent approach [Itti 2000] to saliency detection. The motivation for crowdsourcing is that we wish to understand how *humans* interact with a virtual shape. Hence it is natural to ask humans, collect data from them, and learn from the data. A motivation for taking a learning approach is that it is difficult to manually define a measure for tactile saliency. Moreover, if we use existing 3D shape descriptors, the algorithm may be dependent on the human-specified features. We leverage the strength of deep learning such that features do not need to be manually defined.

Computing tactile mesh saliency from geometry alone is a challenging, if not impossible, computational problem. Yet humans have great intuition at recognizing such saliency information for many

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org. © 2016 ACM.
SIGGRAPH '16 Technical Paper, July 24-28, 2016, Anaheim, CA
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925927

3D shapes even with no color or texture. While a human finds it difficult to assign absolute saliency values (e.g. vertex i has value 0.8), he/she can typically rank whether one point is more tactile salient than another (e.g. vertex i is more likely to be grasped than vertex j). Hence we do not, for example, solve the problem with a regression approach. The human-provided rankings lead us to a ranking-based learning approach. However, recent similar learning approaches in graphics [Garces et al. 2014; O'Donovan et al. 2014; Liu et al. 2015a] typically learn simple scaled Euclidean distance functions. In contrast, we combine the key concepts of deep learning and learning-to-rank methods. We do not intend to replicate the large scale of deep architectures that have been shown for image processing problems. In this paper, we combine a deep architecture (which can represent complex non-linear functions) and a learning-to-rank method (which is needed for our "ranking"-based data) to develop a deep ranking formulation for the tactile mesh saliency problem and contribute a new backpropagation as the solution.

We first collect crowdsourced data where humans compare the tactile saliency of pairs of vertices on various 3D meshes. We represent a 3D shape with multiple depth images taken from different viewpoints. We take patches from the depth images and learn a deep neural network that maps a patch to a saliency value for the patch center. The same deep neural network can be used across different depth images and 3D shapes, while different networks are needed for each tactile modality. After the learning process, we can take a new 3D mesh and compute a tactile saliency value for every mesh vertex. Since our approach is based on ranking, these are relative values and have more meaning when compared with each other.

We compute saliency maps for three tactile interactions for 3D meshes from online sources including Trimble 3D Warehouse and the Princeton Shape Benchmark [Shilane et al. 2004]. We evaluate our results with a comparison to user labeled data and a comparison to a typical learning-to-rank method with a linear function. We demonstrate our framework with the applications of material suggestion for rendering and fabrication.

The contributions of this paper are: (1) We introduce the concept of tactile mesh saliency; (2) We develop a new formulation of deep learning and learning-to-rank methods to solve this tactile saliency problem; and (3) We demonstrate applications of material suggestion for rendering and fabrication.

2 Related Work

2.1 Saliency

Saliency in Mesh Processing. The "Mesh Saliency" work of Lee et al. [2005] introduced the concept of saliency for 3D meshes. Earlier work [Watanabe and Belyaev 2001; Hisada et al. 2002] detect perceptually salient features in the form of ridges and ravines on polygon meshes. Howlett et al. [2005] study visual perception and predict the saliency for polygonal models with eye tracking. Instead of salient points or features, Shilane et al. [2007] identify distinctive regions of a mesh that distinguish a mesh's object type compared to other meshes. Kim et al. [2010] take a visual perception approach to compare the mesh saliency method [Lee et al. 2005] with human eye movements captured by eye tracking devices. Song et al. [2014] include global considerations by incorporating spectral attributes of a mesh, in contrast to previous methods based on local geometric features. While there has been much existing work on saliency and shape similarity [Gal and Cohen-Or 2006; Shtrom et al. 2013; Tao et al. 2015], their focus is on *visual saliency*. "Schelling points" provides another interpretation of saliency on mesh surfaces in terms of human coordination by "asking people to select points on meshes that they expect will be selected by other people" [Chen et al. 2012]. Liu et al. [2015b] detects the saliency of 3D shapes by studying how a human uses the object and not based on geometric features. Our work is different as we explore the concept of *tactile saliency on mesh surfaces*.

Saliency in Image Processing. Visual saliency is a well-studied topic in image processing. Previous works compute saliency maps and identify salient objects and regions in images [Itti et al. 1998; Goferman et al. 2012], and build image saliency benchmarks [Borji et al. 2012; Bylinskii et al. 2015]. Furthermore, there is work in the collection of touch saliency information for mobile devices [Xu et al. 2012], consisting of touch behaviors on the screens of mobile devices as a user browses an image. The touch behaviors can be used to generate visual saliency maps and be compared against saliency maps computed with image processing methods. Our concept of "touch" is for touching of 3D statue models.

2.2 Learning

Crowdsourcing and Learning. There exists previous work in applying crowdsourcing and learning techniques to solve problems related to 2D art, images, and 3D shapes. Our overall crowdsourcing and learning approach is inspired by a previous method for learning a similarity measure of styles of 2D clip art [Garces et al. 2014]. Crowdsourcing has been used to develop tools to explore font collections [O'Donovan et al. 2014]. Crowdsourcing has also been applied to solve vision problems such as extracting depth layers and image normals from a photo [Gingold et al. 2012a], and to convert low-quality inputs of drawings into high-quality outputs [Gingold et al. 2012b]. For 3D shape analysis, "Schelling points" [Chen et al. 2012] on 3D mesh surfaces can be found by first having humans select them in a coordination game and then learning them for new meshes. In our work, we take a crowdsourcing and learning framework for a different problem of tactile mesh saliency.

Deep Learning. Previous works [Wang et al. 2014; Zagoruyko and Komodakis 2015; Hu et al. 2014; Hu et al. 2015] have combined these concepts of learning for image processing problems: deep learning, ranking-based learning, metric learning, and Siamese networks (i.e. using same weights for two copies of network). One key difference in our work is in our problem formulation for our (A, B) and (C, D) data pairs and corresponding terms throughout our backpropagation (for four copies of network). Deep learning methods have also been recently applied to 3D modeling, for example for 3D shape recognition [Su et al. 2015] and human body correspondences [Wei et al. 2015]. We combine the concepts of deep architectures and learning-to-rank to solve the tactile mesh saliency problem. In particular, our solution for 3D shapes (i.e. multi-viewpoint representation, deep neural network architecture computing saliency for patch center, and combining results from viewpoints) is fundamentally different.

2.3 Grasping and Haptics

Robotic Grasping. There exists much work on finding and analyzing robot grasps for real-world objects. Goldfeder et al. [2009] build a robot grasp database and focus on generating and analyzing the grasps of robotic hands to facilitate the planning of grasping motions. Sahbani et al. [2012] provide an overview of grasp synthesis algorithms for generating 3D object grasps with autonomous multi-fingered robotic hands. Bohg et al. [2014] provide a survey of work on grasp synthesis for finding and ranking candidate grasps. The focus of previous work in this area is on grasp synthesis, while our focus is on tactile saliency based on human perception and for graphics purposes. Our output is different as for example a human can perceive the touching of a shape without physically touching it.

There is also previous work on generating grasp points from images and shapes. Saxena et al. [2007] learn a grasping point for an object in an image directly from the input image such that a robot

can grasp novel objects. Sahbani et al. [2009] first identify a graspable part of a 3D shape by segmenting the shape into distinct parts. They then generate contact points for grasping with a multi-fingered robot hand. Klank et al. [2009] match CAD models to noisy camera data and use preprocessed grasping points on the CAD models for a robot to grasp them. Varadarajan et al. [2012] take RGB-depth data from a cluttered environment, estimate 3D shapes from the data, and then generate specific grasp points and approach vectors for the purpose of planning of a robot hand. They generate specific grasp locations for robotic applications. In this paper, we solve a more general problem as we compute saliency information on the whole mesh surface for different tactile interactions according to human perception and for graphics applications.

Haptics. Haptic feedback devices allow a human to physically touch and interact with virtual objects. A previous work on haptics and perception [Plaisier et al. 2009] performs experiments where a user's hand recognizes the salient features of real objects, for example to recognize a cube among spheres. Our work takes virtual meshes as input but we do not directly touch and interact with them.

2.4 Applications

Rendering Appearances. Many techniques have been developed for modeling the appearance of weathering and aging [Mérillou and Ghazanfarpour 2008]. Modeling the appearance requires the representation of a local effect, such as the development of patina [Dorsey and Hanrahan 1996] and the spatial distribution of that local effect. For some types of aging, the spatial distribution can be determined by means of simulating natural phenomena such as flow [Liu et al. 2005]. However, for spatial distribution of material aging due to human interaction, such a simulation is not feasible. Our tactile saliency map can be used as a predictor of the spatial distribution of appearance effects due to human interaction.

Fabrication and Geometry Modeling. Recent work has considered physical properties of virtual shapes for the purpose of fabrication. For example, there is work in analyzing the strength of a 3D printed object [Zhou et al. 2013] and in learning the material parameters including color, specularity, gloss, and transparency of 3D meshes [Jain et al. 2012]. In addition, there has been work in fabricating objects based on virtual shapes. Lau et al. [2011] build real-world furniture by generating parts and connectors that can be fabricated from an input 3D mesh. Bacher et al. [2012] fabricate articulated characters from skinned meshes. Hildebrand et al. [2013] decompose a 3D shape into parts that are fabricated in an optimal direction. Schwartzburg et al. [2013] and Cignoni et al. [2014] generate interlocking planar pieces that can be laser cut and slotted together to resemble the original 3D shape. In this growing field of fabrication, this paper makes a contribution by computing tactile saliency on a 3D mesh surface from its geometry, which can be useful for suggesting materials to fabricate the shape.

There have been many developments in the area of geometry processing on analyzing virtual 3D meshes for various purposes. Some of these relates to our work as a general understanding of meshes can help to identify tactile saliency information. In particular, there are many methods for segmenting and labeling 3D meshes [Chen et al. 2009; Kalogerakis et al. 2010]. Given a segmentation, we may be able to extract some information about tactile saliency. However, computing our saliency directly without an intermediate segmentation step is more general and can avoid potential errors in the intermediate step. Also, segmentation gives discrete parts whereas we generate continuous values over a mesh surface. Given our saliency information, we may be able to segment a mesh into distinct parts but this is not our focus. To demonstrate our application of fabrication material suggestion, we do separate a mesh into distinct parts if each part were to be fabricated with different materials.

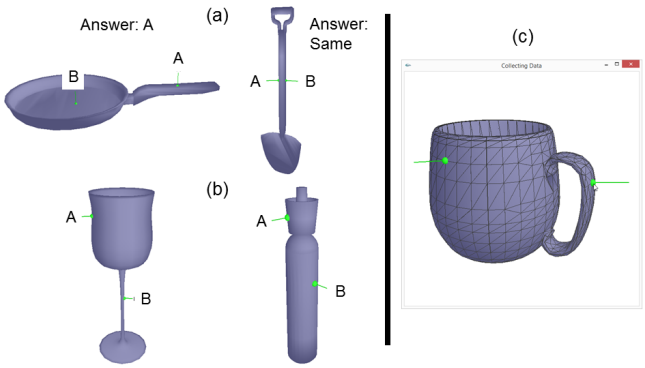

Figure 2: *(a) Two examples of images with correct answers given as part of the instructions for Amazon Mechanical Turk HITs. Text instructions were given to users: they are specifically asked to imagine the virtual shape as if it were a real-world object, and to choose which point is more salient (i.e. grasp to pick up, press, or touch for statue) compared to the other or that they have the same saliency. (b) Two examples of images of HITs we used. (c) Screenshot of software where user directly selects pairs of vertices and specify which is more salient (or same).*

3 Collecting Saliency Data

Our framework collects saliency data from humans and learns a saliency measure from the data. This section describes the process of collecting data from humans about the tactile saliency of 3D mesh points. The data for each tactile interaction is collected separately. Throughout the data collection process, the users perceive how they may interact with virtual meshes and are not given any real objects. We collected 150 3D meshes representing various types of objects from online datasets such as Trimble 3D Warehouse and the Princeton Shape Benchmark [Shilane et al. 2004].

We ask humans to label saliency data. However, it is difficult for humans to provide absolute saliency values (for example, to provide a real number value to a mesh vertex). The key to our data collection is that humans can compare saliency between pairs of vertices more easily, similar to [Garces et al. 2014] where humans can compare relative styles of 2D clip art more easily. Hence we ask humans to compare between pairs of vertices of a mesh and decide which vertex is more salient (or that they have the same saliency).

We used two methods for collecting data. First, we generated images of pairs of vertices on virtual 3D meshes and asked humans to label them on Amazon Mechanical Turk. A human user is initially given instructions and example images with correct answers (Figure 2a). Each HIT (a set of tests on Amazon Mechanical Turk) then consists of 24 images (see Figure 2b for some examples). For each image, the user selects either "A" or "B" if one of the labeled vertices is more salient, or "same" if he/she thinks that both vertices have equal saliency. For the interaction of grasping, we specify that we do not intend the human to grasp an object with one point, but the user should think of grasping the object as a whole to decide which point is more likely included in the grasping. For meshes where the size is important, we also give the user information about the size on the image (e.g. toy car of length 5 centimeters). We paid $0.10 for each HIT. A user typically takes a few seconds for each image and about one to two minutes for each HIT. We had 118 users and 4200 samples of data (2600 for grasp, 1100 for press, and 500 for touch) where each sample is one image. The crowdsourced data may be unreliable. Before a user can work on the HITs, he/she needs to pass a "qualification" test by correctly answering at least four of five images. For each HIT, we have four control images and the user must correctly answer three of them for us to accept the

data. We rejected 8.6% of HITs.

Second, we provide a software tool for users to select pairs of vertices. The user visualizes a mesh in 3D space and directly clicks on a vertex with the mouse to select it (Figure 2c). The user then provides the label (i.e. which vertex is more salient or same) for each pair of vertices with keyboard presses. We asked users to try to select vertices over the whole mesh. This method provides more reliable data as we can give more guidance to the users from the start, and hence we do not reject any data collected with this method. The tradeoff is that this method may not be able to collect data on a large scale if needed. A user can label hundreds of samples each hour and we paid $12 per hour. For this method, we had 30 users and collected 13200 samples (7700 for grasp, 4100 for press, and 1400 for touch).

From the data collection, we have the "original" data sets $\mathcal{I}_{orig}$ and $\mathcal{E}_{orig}$. $\mathcal{I}_{orig}$ contains pairs of vertices (v_A, v_B) where vertex A is labeled as more salient than vertex B. $\mathcal{E}_{orig}$ contains pairs of vertices (v_C, v_D) where vertices C and D are labeled as having the same saliency. Each data sample from these sets has two different vertices, and some vertices are repeated across samples. The set $\mathcal{V} = \{v_1, \ldots, v_h\}$ contains all the vertices, where h is the total number of vertices on all meshes that were labeled. We have h = 23517 vertices (13473 for grasp, 7523 for press, and 2521 for touch). The total number of labeled vertices is much smaller than the total number of vertices in all meshes.

4 Multi-View Deep Ranking

In this section, we describe our framework for learning a tactile saliency measure with the collected data in Section 3. We learn a measure that maps from a vertex to a saliency value. The problem is challenging as we need to develop the appropriate data representation, problem formulation, and network architecture. As our collected data is ranking-based (i.e. some vertices are ranked to be more salient than others), we take a learning-to-rank approach which is commonly used in information retrieval and web page ranking to rank the vertices of a mesh according to their saliencies. We leverage the strength of deep learning to learn complex non-linear functions by using the fundamental concept of learning multiple layers in a neural network architecture. We contribute a deep ranking method: a formulation of learning-to-rank that works with backpropagation in a deep neural network that can be used to solve our tactile saliency problem.

We first describe the processing of the collected data into a multiple-view representation. We then describe the deep ranking formulation, including the overall loss function and the backpropagation in the neural network that takes into account the concept of learning-to-rank. After the measure is learned, we can use it to compute saliency values for all vertices of a mesh.

4.1 Multiple-View Data Representation

Inspired by approaches that take multi-view representations of 3D shapes [Chen et al. 2003; Su et al. 2015] for other geometry processing problems, we represent a 3D mesh with multiple depth images from various viewpoints. We scale each mesh to fit within each depth image. The collected "original" data sets $\mathcal{I}_{orig}$ and $\mathcal{E}_{orig}$ are converted to training data sets $\mathcal{I}_{train} = (\mathbf{x}_A^{(view_i)}, \mathbf{x}_B^{(view_j)})$ and $\mathcal{E}_{train} = (\mathbf{x}_C^{(view_i)}, \mathbf{x}_D^{(view_j)})$. Each pair in the original sets (v_A, v_B) becomes various pairs of $(\mathbf{x}_A^{(view_i)}, \mathbf{x}_B^{(view_j)})$. $\mathbf{x}_A^{(view_i)}$ is a smaller and subsampled patch of the depth image from view i for vertex v_A. To convert from v to $\mathbf{x}$ for each viewpoint or depth image, the vertex v that is visible from that viewpoint is projected to coordinates in the depth image, and a patch with the projected coordinates as its center is extracted as $\mathbf{x}$. Each pair (v_A, v_B) can have a different number of views (typically between six and fourteen). The two vertices in the same pair can have two different viewpoints as long as the corresponding vertices are visible.

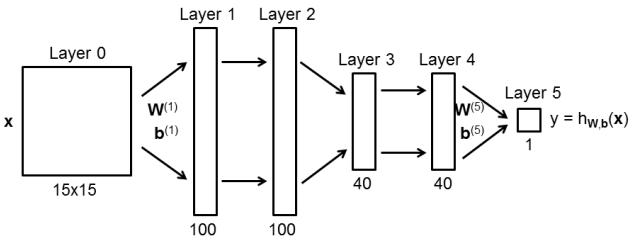

Figure 3: *Our deep neural network with 6 layers.* $\mathbf{x}$ *is a smaller and subsampled patch of a depth image and* y *is the patch center's saliency value. The size of each depth image is 300x300. We take smaller patches of size 75x75 which are then subsampled by 5 to get patches* $(\mathbf{x})$ *of size 15x15. This patch size corresponds to real-world sizes of about 4-50 cm. The number of nodes is indicated for each layer. The network is fully connected. For example,* $\mathbf{W}^{(1)}$ *has 100x225 values and* $\mathbf{b}^{(1)}$ *has 100x1 values. The network is only for each view or each depth image and we compute the saliency for multiple views and combine them to compute the saliency of each vertex. Note that we also need four copies of this network to compute the partial derivatives for the batch gradient descent.*

4.2 Deep Ranking Formulation and Backpropagation

Our algorithm takes as input the sets $\mathcal{I}_{train}$ and $\mathcal{E}_{train}$ and learns a deep neural network that maps a patch $\mathbf{x}$ to the patch center's saliency value $y = h_{\mathbf{W},\mathbf{b}}(\mathbf{x})$ (Figure 3). We experimented with different network architectures for our problem and found that it can be difficult to represent the position of the pixel that we are computing the saliency for. Our problem formulation was the most effective among the architectures we tested. The neural network is fully-connected. We learn $\mathbf{W}$ which is the set of all weights $(\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(5)})$ where $\mathbf{W}^{(l)}$ is the matrix of weights for the connections between layers $l-1$ and l, and $\mathbf{b}$ which is the set of all biases $(\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(5)})$ where $\mathbf{b}^{(l)}$ is the vector of biases for the connections to layer l. The same neural network can be used across different depth images and 3D shapes. Each tactile interaction is learned separately and needs a different network.

In contrast to typical supervised learning frameworks, we do not directly have the target values y that we are trying to compute. Our data provides rankings of pairs of vertices. Hence we take a learning-to-rank formulation and learn $\mathbf{W}$ and $\mathbf{b}$ to minimize the following ranking loss function:

$$\mathcal{L}(\mathbf{W},\mathbf{b}) = \frac{1}{2}\|\mathbf{W}\|_2^2 + \frac{C_{param}}{|\mathcal{I}_{train}|} \sum_{(\mathbf{x}_A,\mathbf{x}_B)\in\mathcal{I}_{train}} l_1(y_A - y_B) + \frac{C_{param}}{|\mathcal{E}_{train}|} \sum_{(\mathbf{x}_C,\mathbf{x}_D)\in\mathcal{E}_{train}} l_2(y_C - y_D) \quad (1)$$

where $\|\mathbf{W}\|_2^2$ is the L^2 regularizer (2-norm for matrix) to prevent over-fitting, C_{param} is a hyper-parameter, $|\mathcal{I}_{train}|$ is the number of elements in $\mathcal{I}_{train}$, $l_1(t)$ and $l_2(t)$ are suitable loss functions for the inequality and equality constraints, and $y_A = h_{\mathbf{W},\mathbf{b}}(\mathbf{x}_A)$. We use these loss functions:

$$l_1(t) = \max(0, 1-t)^2 \quad (2)$$

$$l_2(t) = t^2 \quad (3)$$

The two training sets $\mathcal{I}_{train}$ and $\mathcal{E}_{train}$ contain inequality and equality constraints respectively. If $(\mathbf{x}_A, \mathbf{x}_B) \in \mathcal{I}_{train}$, vertex A should be more salient than vertex B and $h(\mathbf{x}_A)$ should be greater than $h(\mathbf{x}_B)$. Similarly $(\mathbf{x}_C, \mathbf{x}_D) \in \mathcal{E}_{train}$ implies equal saliency: $h(\mathbf{x}_C)$ should be equal to $h(\mathbf{x}_D)$. The loss function $l_1(t)$ enforces prescribed inequalities in $\mathcal{I}_{train}$ with a standard margin of 1, while the equality loss function $l_2(t)$ measures the standard squared deviations from the equality constraints in $\mathcal{E}_{train}$.

To minimize $\mathcal{L}(\mathbf{W}, \mathbf{b})$, we perform an end-to-end neural network backpropagation with batch gradient descent, but we have a new formulation that is compatible with learning-to-rank and with our ranking-based data. First, we have a forward propagation step that takes each pair $(\mathbf{x}_A, \mathbf{x}_B) \in \mathcal{I}_{train}$ and propagates $\mathbf{x}_A$ and $\mathbf{x}_B$ through the network with the current $(\mathbf{W}, \mathbf{b})$ to get y_A and y_B respectively. Similarly, $\mathbf{x}_C$ and $\mathbf{x}_D$ from each pair $(\mathbf{x}_C, \mathbf{x}_D) \in \mathcal{E}_{train}$ are propagated. Hence there are four copies of the network for each of the four cases A, B, C, and D.

We then perform a backward propagation step for each of the four copies of the network and compute these delta (δ) values:

$$\delta_i^{(n_l)} = y(1-y) \qquad \text{for output layer} \quad (4)$$

$$\delta_i^{(l)} = (\sum_{k=1}^{s_{l+1}} \delta_k^{(l+1)} w_{ki}^{(l+1)}) \ (1 - (a_i^{(l)})^2) \qquad \text{for inner layers} \quad (5)$$

where the δ and y values are indexed as δ_{Ai} and y_A in the case for A. The index i in δ is the neuron in the corresponding layer and there is only one node in our output layers. n_l is the number of layers, s_{l+1} is the number of neurons in layer $l+1$, $w_{ki}^{(l+1)}$ is the weight for the connection between neuron i in layer l and neuron k in layer $(l+1)$, and $a_i^{(l)}$ is the output after the activation function for neuron i in layer l. We use the $tanh$ activation function which leads to these δ formulas. Note that due to the learning-to-rank aspect, we define these δ to be different from the usual δ in the standard neural network backpropagation.

We can now compute the partial derivatives for the gradient descent. For $\frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}}$, we split this into a $\frac{\partial \mathcal{L}}{\partial \|\mathbf{W}\|_2} \frac{\partial \|\mathbf{W}\|_2}{\partial w_{ij}^{(l)}}$ term and $\frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial w_{ij}^{(l)}}$ terms (a term for each y_A and each y_B computed from each $(\mathbf{x}_A, \mathbf{x}_B)$ pair and a term for each y_C and each y_D computed from each $(\mathbf{x}_C, \mathbf{x}_D)$ pair). The $\frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial w_{ij}^{(l)}}$ term is expanded for the A case for example to $\frac{\partial \mathcal{L}}{\partial y_A} \frac{\partial y_A}{\partial a_i} \frac{\partial a_i}{\partial z_i} \frac{\partial z_i}{\partial w_{ij}^{(l)}}$ where the last three partial derivatives are computed with the copy of the network for the A case. z_i is the value of a neuron before the activation function. The entire partial derivative is:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}} &= w_{ij}^{(l)} \\
&+ \frac{2C_{param}}{|\mathcal{I}_{train}|} \sum_{(A,B)} max(0, 1 - y_A + y_B)\, chk(y_A - y_B)\, \delta_{Ai}^{(l+1)} a_{Aj}^{(l)} \\
&- \frac{2C_{param}}{|\mathcal{I}_{train}|} \sum_{(A,B)} max(0, 1 - y_A + y_B)\, chk(y_A - y_B)\, \delta_{Bi}^{(l+1)} a_{Bj}^{(l)} \\
&+ \frac{2C_{param}}{|\mathcal{E}_{train}|} \sum_{(C,D)} (y_C - y_D)\, \delta_{Ci}^{(l+1)} a_{Cj}^{(l)} \\
&- \frac{2C_{param}}{|\mathcal{E}_{train}|} \sum_{(C,D)} (y_C - y_D)\, \delta_{Di}^{(l+1)} a_{Dj}^{(l)}
\end{aligned} \quad (6)$$

There is one term for each of the A, B, C, and D cases. (A, B) represents $(\mathbf{x}_A, \mathbf{x}_B) \in \mathcal{I}_{train}$ and all terms in the summation can be computed with the corresponding $(\mathbf{x}_A, \mathbf{x}_B)$ pair. The $chk()$ function is:

$$chk(t) = 0 \qquad \text{if } t \geq 1 \quad (7)$$

$$= -1 \qquad \text{if } t < 1 \quad (8)$$

For each (A, B) pair, we can check the value of $chk(y_A - y_B)$ before doing the backpropagation. If it is zero, we do not have to perform the backpropagation for that pair as the term in the summation is zero. The partial derivative for the biases is similar:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial b_i^{(l)}} &= \frac{2C_{param}}{|\mathcal{I}_{train}|} \sum_{(A,B)} max(0, 1 - y_A + y_B)\, chk(y_A - y_B)\, \delta_{Ai}^{(l+1)} \\
&- \frac{2C_{param}}{|\mathcal{I}_{train}|} \sum_{(A,B)} max(0, 1 - y_A + y_B)\, chk(y_A - y_B)\, \delta_{Bi}^{(l+1)} \\
&+ \frac{2C_{param}}{|\mathcal{E}_{train}|} \sum_{(C,D)} (y_C - y_D)\, \delta_{Ci}^{(l+1)} \\
&- \frac{2C_{param}}{|\mathcal{E}_{train}|} \sum_{(C,D)} (y_C - y_D)\, \delta_{Di}^{(l+1)}
\end{aligned} \quad (9)$$

The batch gradient descent starts by initializing $\mathbf{W}$ and $\mathbf{b}$ randomly. We then go through the images for a fixed number of iterations, where each iteration involves taking a set of data samples and performing the forward and backward propagation steps and computing the partial derivatives. Each iteration of batch gradient descent sums the partial derivatives from a set of data samples and updates $\mathbf{W}$ and $\mathbf{b}$ with a learning rate α as follows:

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}} \quad (10)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial b_i^{(l)}} \quad (11)$$

4.3 Using Learned Saliency Measure

After learning $\mathbf{W}$ and $\mathbf{b}$, we can use them to compute a saliency value for all vertices of a mesh. The learned measure gives a relative saliency value where the saliency of a vertex is with respect to the other vertices of the mesh. For each vertex v_i, we choose a set of views $view_j$ where v_i is visible and compute the subsampled patches $\mathbf{x}_i^{(view_j)}$. The views can in theory be random but in practice we pick a small set of views from the set used in the training process. If a vertex is not directly visible from any viewpoint, we can take a set of views even if the vertex is occluded. We compute $h_{\mathbf{W},\mathbf{b}}(\mathbf{x}_i^{(view_j)})$ for each j with the learned $\mathbf{W}$ and $\mathbf{b}$, and take the average of these values to get the saliency value for v_i.

5 Results: Tactile Saliency Maps

We demonstrate our approach with three tactile interactions. The saliency maps show the results of the *crowdsourced consensus* as they combine the data from various people. Note that the human users only provided data for a very small number of vertices on the training meshes, and it would be tedious for a human to label them all. We generate the saliency maps by computing the saliency values for each vertex, and then mapping these values (while maintaining the ranking) to $[0, 1]$ such that each vertex can be assigned a color for visualization purposes. We also blend the saliency values by blending each vertex's value with those of its neighbors. The saliency results should be interpreted as follows (as this is how the data was collected): we should think of the virtual shape to be a real object and perceive how likely we are to grasp, press, or touch each point. Since our results provide a relative ranking, a single vertex labeled red for example may not necessarily be salient on its own.

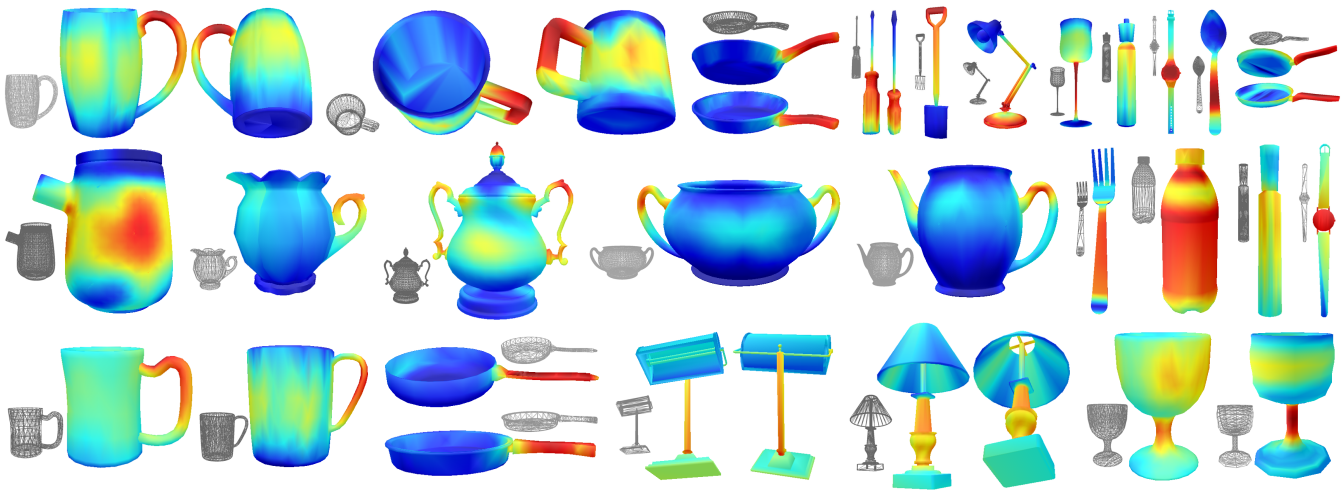

Figure 4: *"Grasp" saliency maps (grasp to pick up objects). Each example has the input mesh and corresponding result (some with two views). The top row shows meshes used in the training data while the bottom two rows show new meshes.*

For the parameters of our network, we set the hyper-parameter C_{param} to 1000. We initialize each weight and bias in $\mathbf{W}$ and $\mathbf{b}$ by sampling from a normal distribution with mean 0 and standard deviation 0.1. We go through all images at least 100 times or more for the network to produce reasonable results. For each iteration of the batch gradient descent, we typically choose between 100 and 200 data samples for $\mathcal{I}_{train}$ and $\mathcal{E}_{train}$. The learning rate α is set to 0.0001. The learning process can be done offline. For example, 100 iterations of batch gradient descent for one 3D mesh with about 10 viewpoints and 100 data samples takes about 20 seconds in MATLAB. This runtime scales linearly as the number of 3D meshes increases. After the weights and biases have been trained, computing the saliency of each vertex requires straightforward forward propagations and the runtime is interactive.

5.1 Grasp Saliency Maps

Figures 1 (left) and 4 show the results for grasp saliency. These are specifically for grasping to pick up objects as there can be other types of grasping. Our method generalizes well when it is applied to new data. For example, our method learns the parts in the 3D shapes that should be grasped such as handles in the teapots and trophy (Figure 4, 2^{nd} row), and these overall shapes are new testing models that are very different from those in the training data. The results for the desklamps (Figure 4, 3^{rd} row) are also interesting, since these are new testing models that do not appear in the training data and the graspable parts are successfully learned. Furthermore, the results for the cup handles (Figure 4) may seem counter-intuitive, as they are often computed to be more likely grasped at the top part than the bottom part. However, these results are explained by the user data which gives the crowdsourced consensus, as the users ranked points near the top part of the handle as more likely to be grasped than points near the middle and bottom parts of the handle.

Objects of Different Sizes. Figure 5 (left) shows grasping results that consider objects of different sizes. For each case in the figure, we told the user whether it is a real size car or a toy size car (e.g. telling user during data collection that car is of length 5cm). We scale them according to their sizes in the depth images. Users prefer to "grasp" a real size car on the door handles of the car. On the other hand, users prefer to grasp a toy size car around the middle more than at the front and back ends of the car. Our examples show that we can obtain different results for objects of different sizes.

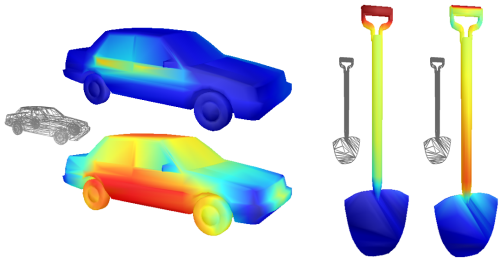

Figure 5: *Left (Objects of Different Sizes): For the same car mesh, the top image shows the grasp saliency for a real size car and the bottom image shows a different grasp saliency for a toy size car. Right (Grasping Sub-Types): For the same shovel mesh, the left shovel is for grasping to pick up and the right shovel is for grasping to use. The region near the blade in the right shovel is more likely to be grasped than for the left shovel.*

Grasping Sub-Types. Figure 5 (right) shows an example for two sub-types of grasping: grasping to pick up an object and grasping to use an object. These are considered to be different grasping modalities and we collect the data and learn the saliency measures separately. For the "grasping to use" case, a human typically grasp the shovel's handle with one hand and use the other hand to grasp at the region near the blade. Our shovel example shows that, for the same mesh, different modalities can lead to different results.

5.2 Press Saliency Maps

Figures 1 (middle) and 6 show examples of press saliency maps. Our method learns to identify the parts of 3D shapes that can be pressed such as buttons and touch screens. An interesting result is in the perceived relative likelihood of pressing buttons on the game controllers: some buttons are more likely to be pressed than others. However, this is not the case for the microwave as there is less consensus on which microwave buttons are more likely to be pressed, since the buttons on different microwaves may be different.

Multiple Tactile Modalities for Same Object. We can learn multiple tactile saliency measures for the same object, as an object may be grasped, pressed, touched, or interacted with in different ways. An example is the watch models. Figure 4 shows the "grasping" of watches where users prefer to grasp near the middle of the watch and then progressively less towards the top and bottom ends. Figure 6 shows the "pressing" of watches where users prefer to press

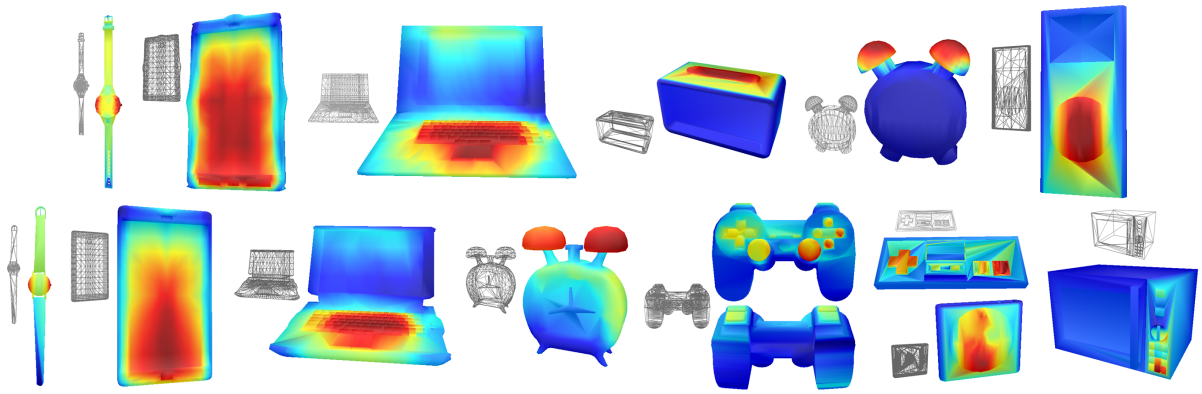

Figure 6: *"Press" tactile saliency maps. Each example has the input mesh and corresponding result (some with two views). The top row shows meshes used in the training data while the bottom row shows new meshes.*

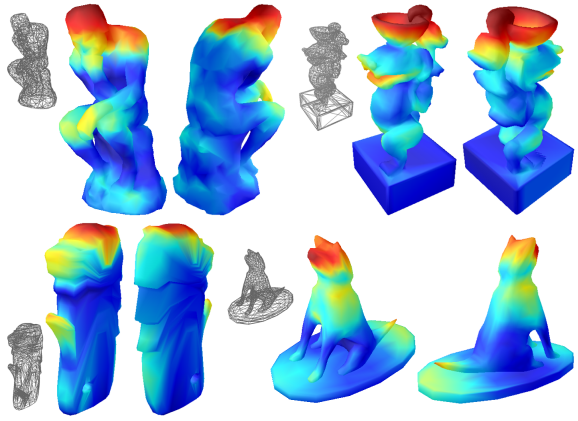

Figure 7: *"Touch" saliency maps are specifically for touching statues. Each example shows the input mesh and the saliency map (two views). The top row shows meshes used in the training data while the bottom row shows new meshes.*

the buttons on the sides of the watch more than any other parts.

5.3 Touch Saliency Maps

We demonstrate touch saliency specifically for the touching of statues and not for "touching" 3D shapes in general. We show examples of results in Figures 1 (right) and 7. The results show that humans tend to touch the top part or the head regions of the statues, and then also significant parts such as hands, mouth, and tail. The algorithm learns to assign higher saliency values to these protruding and/or significant parts.

6 Evaluation

6.1 Network Parameters and Robustness

There is typically a wide range of parameters for the learning to find a solution for the 3D models that we have tested. The number of iterations of batch gradient descent, the learning rate α, and the initialization of the weights $\mathbf{W}$ and biases $\mathbf{b}$ are the parameters that we adjust most often (in this order). We initially set the parameters based on cross-validation. For example, the hyper-parameter C_{param} is chosen from $\{0.01, 0.1, 1, 10, 10^2, 10^3, 10^4\}$. For validation, we used only inequality constraints since the equality constraints will not be precisely met in practice. The optimal C_{param} is the one that minimizes the validation error.

We use a patch size of 15x15 (a smaller and subsampled patch of a depth image). The disadvantage of this size is that we only have

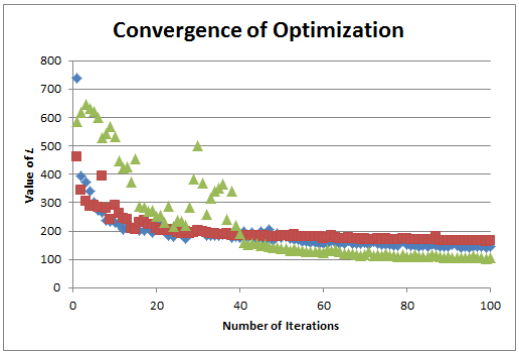

Figure 8: *Example plots (three colors for three cases) of the overall loss function $\mathcal{L}$ versus number of iterations in the batch gradient descent. They show the convergence in our optimization.*

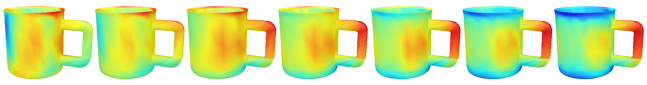

Figure 9: *Progression of results (grasp saliency) for a mug model as the number of iterations (images show iteration number $10, 20, ..., 70$) in the batch gradient descent increases.*

local information. However, our result is that local information is enough to predict tactile saliency. This patch size is a parameter. Increasing this size can lead to more global information until we get the original depth image with the pixel to be predicted at its center, but this can also lead to a longer learning time. We take a relatively small patch size as it already works well and is efficient.

Figure 8 shows plots of the overall loss function $\mathcal{L}$ versus the number of iterations. The value of $\mathcal{L}$ gradually converges. We can see from the figure that it is intuitive to set the number of iterations after visualizing such plots. Figure 9 shows the progression of results of a mug model during the optimization. The results are not accurate near the start and gradually moves towards a good solution.

We give some idea of what the neural network computes with the images in Figure 10. We use the learned measures to compute the saliency for each pixel in the depth images. These images already show preliminary results and note that we combine multiple viewpoints for each vertex to compute the final saliency value.

6.2 Quantitative Evaluation

We evaluate whether our learned measure can predict new examples by comparing with ground truth data. We take the human labeled

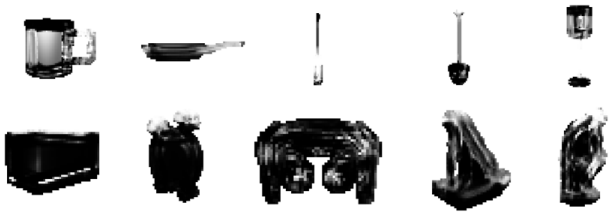

Figure 10: *We show the results for individual depth images for various 3D models and viewpoints. These are intermediate results and we combine them from different viewpoints to get our saliency measure. The* $[0, 1]$ *grayscale colors indicate least to most salient. Top row: for grasping. Bottom row: for pressing (first three) and touching (last two).*

data itself to be the ground truth. We perform a 5-fold cross validation of the collected data, where the training data is used to learn a saliency measure and we report the percentage error for the left-out validation data (Table 1, Deep Ranking column). We take only the data in the inequality set $\mathcal{I}_{orig}$, as the equality set $\mathcal{E}_{orig}$ contains vertex pairs with the "same" saliency and it is difficult to numerically determine if two saliency values are exactly equal. For the pairs of vertices in $\mathcal{I}_{orig}$, the prediction from the learned measure is incorrect if the collected data says v_A is more salient than v_B, but the computed saliency of v_A is less than that of v_B.

We also compare between our deep ranking method and an existing learning-to-rank method that has an underlying linear representation (Table 1, RankSVM column). For "RankSVM", we compute features manually, use the same saliency data we already collected, and learn with the RankSVM method [Chapelle and Keerthi 2010]. We explicitly compute a feature vector of 3D shape descriptors for each mesh vertex, except that we use a variant of the commonly used version of some descriptors as we compute features for a vertex relative to the whole model rather than for the whole model. The features include: D2 Shape Distribution [Osada et al. 2001], Gaussian Image [Horn 1984], Light Field Descriptors [Chen et al. 2003; Shilane et al. 2004], and Gaussian and Mean curvatures [Surazhsky et al. 2003]. We then use RankSVM which computes a weight vector with the same dimensions as our feature vector. The saliency measure is a linear function and is the dot product of the learned weight vector and a feature vector. We use the same overall loss function as in Equation 1 except with the linear function and weights. We minimize this loss function using the primal Newton method as originally developed by Chapelle [Chapelle and Keerthi 2010] for inequality constraints and subsequently adapted by Parikh and Grauman [Parikh and Grauman 2011] for equality constraints. The results show that a deep multiple layer architecture makes a significant difference compared to a linear saliency measure.

6.3 User Study

We performed a user study to evaluate our learned saliency measures. The idea is to evaluate our measures by comparing them with data perceived by real-world users for virtual meshes and physical objects. The user experiment started by questions about each user's previous 3D modeling experiences followed by tasks with four objects. For the first object, we ask the user to take a real mug (Figure 11 left) and choose ten pairs of points on it. For each pair, they should select which point is more likely to be grasped. They were told to pick points evenly on the object's surface. We recorded the approximate location of each point on the real mug as the vertex on the corresponding virtual mesh that we modeled. For the second object, they were given a real laptop (Figure 11 right) and to choose ten pairs of points on it and tell us for each pair which point is more likely to be pressed. For the third object, they were given a virtual mesh of a cooking pan. The users can visualize and manipulate (i.e. rotate, pan, zoom) the virtual shape with an interactive tool. In this case, we have already selected ten points on the shape and we asked them to rank the ten points in terms of how likely they will grasp them. Points that are similar in ranking are allowed. For the fourth object, they were given a virtual mesh of a mobile phone. Similar to the third object, they were asked to rank ten already selected points in order of how likely they will press them.

3D Model	No. of Samples	RankSVM (% error)	Deep Ranking (% error)
Mug	114	10.5	1.8
Cooking Pan	181	9.4	3.3
Screwdriver	64	7.8	1.6
Shovel	88	26.1	2.3
Cell Phone	76	27.6	2.6
Laptop	23	4.3	4.3
Alarm Clock	48	12.5	2.1
Game Controller	262	3.4	1.5
Statue of Dog	95	3.2	1.1
Statue of Human	49	10.2	4.1

Table 1: *Comparison between a learning-to-rank method with a typical linear function (RankSVM) and our deep learning-to-rank method. "No. of Samples" is the number of* (v_A, v_B) *pairs from the inequality set* $\mathcal{I}_{orig}$*. "% error" is the percentage of samples that are incorrectly predicted based on cross validation. There are 3 groups of models for the grasp, press, and touch modalities.*

Figure 11: *For real objects: we took a real mug and laptop, created 3D models of them, and computed the grasp saliency map for the mug and the press saliency map for the laptop.*

We had 10 users (2 female). Each user was paid $6 and each session lasted approximately 30 minutes. Two users have previous experiences with Inventor and two users have experiences with Blender.

We took the data that users gave for the real mug as ground truth and compared them with our grasp saliency measure. Our predictions have an error rate of 2.4%, where 16 responses were pairs of vertices perceived to have the same saliency and we did not use these responses. For the data of the real laptop, our press saliency predictions have an error rate of 3.2%, where 7 responses were pairs of vertices perceived to have the same saliency. For the ranking of each set of ten points for the virtual objects, we compared the user rankings with our corresponding saliency measures. We used the NDCG ranking score which is used in information retrieval [Järvelin and Kekäläinen 2002] to give an indication of accuracy. We first use our saliency measure to rank each set of ten points of each object. We then compare this ranking and the user rankings with the NDCG score. The mean NDCG score for the "grasp" object is 0.92 and for the "press" object is 0.90. The results show that our saliency measures correspond to users' perception of saliency.

6.4 Comparison with Real-World Objects

For a real mug and laptop, we created corresponding 3D virtual models of them, and computed their saliency maps (Figure 11). The saliency maps visually correspond to our perception of grasping and pressing. Users prefer to grasp the handle and middle parts of the mug, and users prefer to press the keys and mouse pad of the laptop.

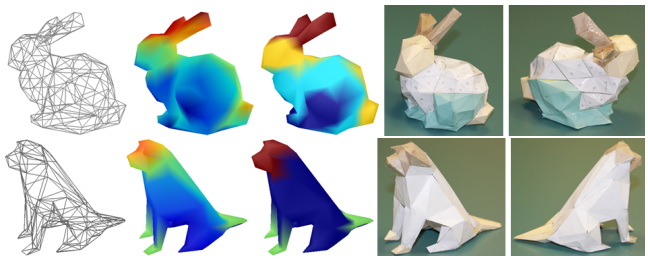

Figure 12: *Fabrication Material Suggestion: Papercraft. The more likely it is to grasp or touch, the more sturdy the material. Top row: input bunny mesh, grasp saliency map, saliencies discretized into 4 clusters, and fabricated paper model (two views). The materials are softer paper (blue in figure), normal paper (white), thicker card (light brown), and cardboard-like paper (brown). Bottom row: input dog statue mesh, touch saliency map, saliencies discretized into 3 clusters, and fabricated paper model (two views).*

6.5 Failure Cases

An example failure case is the knife model in the left figure. For this knife, the handle and blade parts are very similar in geometric shape and hence it is difficult to differentiate between them. Moreover, another category of failure cases is meshes of object types that we have no training data for. As our framework is data-driven, it relies on the available training data.

7 Applications

7.1 Fabrication Material Suggestion: Papercraft

We apply our computed saliency information to fabricate papercraft models. The key concept is that the more likely a surface point of the mesh will be grasped or touched, the more sturdy or stronger the paper material can be. The resulting papercraft model will be more likely to stay in shape and/or not break.

We fabricate papercraft models as follows. An input mesh is simplified to a smaller number of faces while maintaining the overall shape. We compute the saliency map for the simplified shape. The saliency values on all vertices are then discretized into a fixed number of clusters such that each cluster can be made with one material. For each cluster, we unfold the faces into a set of 2D patterns with Pepekura Designer. We print or cut each pattern with a material based on the average saliency of the vertices in the cluster. The patterns are then folded and taped together. Figure 12 shows a bunny paper model and a dog statue paper model. The thickest cardboard-like paper makes it easy to grasp the paper bunny by its ears and makes the head of the dog statue more durable even if that part is touched more.

7.2 Fabrication Material Suggestion: 3D Printing

We can also apply our computed saliency information to suggest different materials for different parts of a mesh depending on how likely the surface points are grasped. The key concept is that the more likely a surface point will be grasped, the more soft the 3D printed material can be. The resulting object will then be more comfortable to grasp. This is motivated by real-world objects such as screwdrivers and shovels where the parts that are grasped are sometimes made with softer or rubber materials.

We fabricate a mesh as follows. We compute the grasp saliency map with the input mesh. The saliency values are separated into a fixed number of clusters. The whole shape is then separated into different volumetric parts by first converting it into voxel space. Each voxel is assigned to the cluster of its closest surface point.

Figure 13: *Fabrication Material Suggestion: 3D Printing. The more likely it is to grasp, the more soft the material to make it more comfortable to grasp. Input screwdriver mesh, grasp saliency map, saliencies discretized and blended into 4 clusters of volumetric parts, and screwdriver with 6 discrete parts and 4 suggested materials fabricated with an Objet Connex multi-material 3D printer.*

These voxel clusters can be blended with their neighbors to make the result more smooth. Each volumetric cluster is converted back to a mesh with the Marching Cubes algorithm. Each part can then be assigned a different material based on the saliency of the cluster. The parts may be 3D printed into a real object with different materials. Figure 13 shows an example of the above process for a screwdriver input mesh. The 3D printed screwdriver is more comfortable to grasp near the middle. The softer material in the middle also inherently suggests to users that they should grasp it there.

7.3 Rendering Properties Suggestion

We can apply our computed saliency information to suggest various colors, material properties, and textures for 3D models. The motivation is to apply the potential effects of human interactions to render a 3D model with only its geometry with realistic and interesting appearances. There are many possible ways to create these effects. We can modulate the color and material properties (such as shininess and ambience properties) of 3D shapes based on the computed saliency values. We can also map different textures to different parts of a mesh based on the saliencies. Figure 14 shows examples of such renderings. We cluster the computed saliencies in different ways to modulate the rendered properties, textures, and to simulate a dirt effect for the mug. We map different textures (e.g. grip textures) to the mug, cooking pan, shovel, screwdriver, and alarm clock to indicate the parts that are more graspable or pressable. We modulate the color, shininess, and map different textures to the dog and human statues to indicate the parts that are more likely to be touched and to make them look more interesting.

8 Discussion

We have introduced the concept of computing tactile saliency for 3D meshes and presented a solution based on combining the concepts of deep learning and learning-to-rank methods. For future work, we will experiment with other tactile modalities and other possible types of human interactions with virtual and real objects. We collected data on user perceptions of interactions with virtual 3D meshes in this paper. In the future, we can also collect data where humans interact with real-world objects, although this may be difficult to scale to a large amount of data.

We have leveraged two fundamental strengths of deep learning by having an architecture with multiple layers and by not using handcrafted 3D shape descriptors. However, there is more to deep learning that we can explore. One assumption we have made is that local information and a small patch size in our learning is enough. Even though we already achieve good results, it would be worthwhile to explore higher resolution depth images and patch sizes to account for more global information, experiment with a larger number of 3D models, and incorporate convolutional methods to handle a larger network architecture.

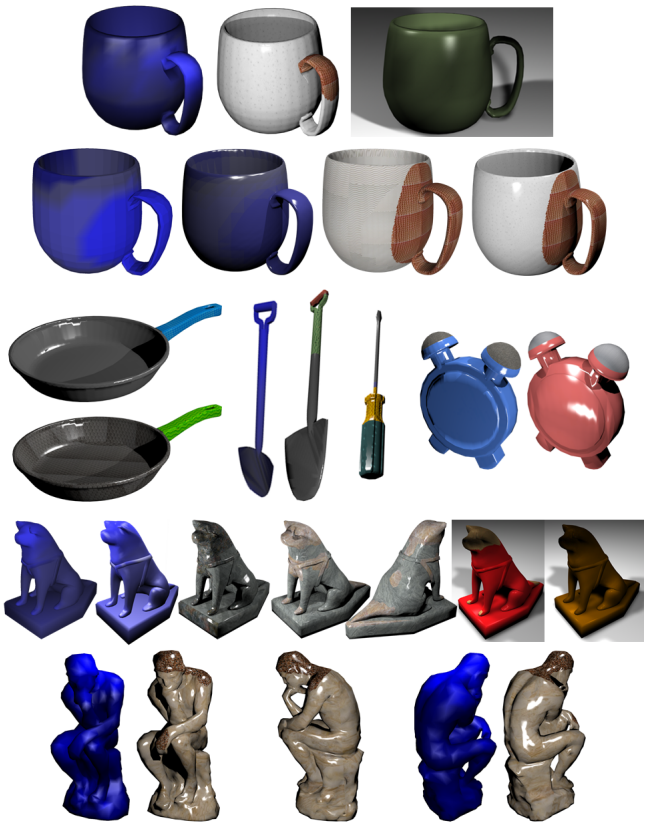

Figure 14: *Rendering Properties Suggestion. Our computed saliency information can be used to suggest different ways to render the 3D shapes to make them look more realistic and interesting.*

A limitation of our method is that it may not work without existing training data for some types of shapes, unless there are other shapes with similar parts in the data. However, this makes sense for a data-driven framework. If a human has never seen an object before, it may not be clear what the important points to grasp the object are. To resolve this limitation, it may be helpful in the future to have some way to indicate how confident the saliency measure is.

There are other potential applications of our work beyond the saliency idea. In robotics, our work can be applied to computing how a robot arm can grasp and/or manipulate an object. In functionality analysis, understanding the saliency of a virtual shape can help to understand its functionality as if it were a real object.

If we can segment and label [Kalogerakis et al. 2010] a 3D mesh first, we may have a better understanding of the shape before computing saliency values. In addition, there is work on assigning materials to 3D models [Jain et al. 2012]. Combining these ideas with our method can be a direction for future work.

Acknowledgements

We thank the reviewers for their suggestions for improving the paper. We thank Kwang In Kim for discussions about machine learning and Nicolas Villar for the multi-material 3D printed object. This work was funded in part by NSF grants IIS-1064412 and IIS-1218515. Kapil Dev was funded by a Microsoft Research PhD scholarship and Manfred Lau was on a sabbatical leave at Yale during this project.

References

BÄCHER, M., BICKEL, B., JAMES, D. L., AND PFISTER, H. 2012. Fabricating Articulated Characters from Skinned Meshes. *ACM Trans. Graph. 31*, 4 (July), 47:1–47:9.

BOHG, J., MORALES, A., ASFOUR, T., AND KRAGIC, D. 2014. Data-Driven Grasp Synthesis – A Survey. *IEEE Transactions on Robotics 30*, 2, 289–309.

BORJI, A., SIHITE, D. N., AND ITTI, L. 2012. Salient Object Detection: A Benchmark. *ECCV*, 414–429.

BYLINSKII, Z., JUDD, T., BORJI, A., ITTI, L., DURAND, F., OLIVA, A., AND TORRALBA, A., 2015. MIT Saliency Benchmark. http://saliency.mit.edu/.

CHAPELLE, O., AND KEERTHI, S. S. 2010. Efficient Algorithms for Ranking with SVMs. *Information Retrieval 13*, 3, 201–215.

CHEN, D. Y., TIAN, X.-P., SHEN, Y.-T., AND OUHYOUNG, M. 2003. On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum 22*, 3, 223–232.

CHEN, X., GOLOVINSKIY, A., AND FUNKHOUSER, T. 2009. A Benchmark for 3D Mesh Segmentation. *ACM Trans. Graph. 28*, 3 (July), 73:1–73:12.

CHEN, X., SAPAROV, A., PANG, B., AND FUNKHOUSER, T. 2012. Schelling Points on 3D Surface Meshes. *ACM Trans. Graph. 31*, 4 (July), 29:1–29:12.

CIGNONI, P., PIETRONI, N., MALOMO, L., AND SCOPIGNO, R. 2014. Field-aligned Mesh Joinery. *ACM Trans. Graph. 33*, 1 (Feb.), 11:1–11:12.

DORSEY, J., AND HANRAHAN, P. 1996. Modeling and Rendering of Metallic Patinas. In *Proceedings of SIGGRAPH 96*, Annual Conference Series, 387–396.

GAL, R., AND COHEN-OR, D. 2006. Salient Geometric Features for Partial Shape Matching and Similarity. *ACM Trans. Graph. 25*, 1 (Jan.), 130–150.

GARCES, E., AGARWALA, A., GUTIERREZ, D., AND HERTZMANN, A. 2014. A Similarity Measure for Illustration Style. *ACM Trans. Graph. 33*, 4 (July), 93:1–93:9.

GINGOLD, Y., SHAMIR, A., AND COHEN-OR, D. 2012. Micro Perceptual Human Computation for Visual Tasks. *ACM Trans. Graph. 31*, 5 (Sept.), 119:1–119:12.

GINGOLD, Y., VOUGA, E., GRINSPUN, E., AND HIRSH, H. 2012. Diamonds from the Rough: Improving Drawing, Painting, and Singing via Crowdsourcing. *Proceedings of the AAAI Workshop on Human Computation (HCOMP)*.

GOFERMAN, S., ZELNIK-MANOR, L., AND TAL, A. 2012. Context-Aware Saliency Detection. *PAMI 34*, 10, 1915–1926.

GOLDFEDER, C., CIOCARLIE, M., DANG, H., AND ALLEN, P. K. 2009. The Columbia Grasp Database. *ICRA*, 3343–3349.

HILDEBRAND, K., BICKEL, B., AND ALEXA, M. 2013. Orthogonal Slicing for Additive Manufacturing. *SMI 37*, 6, 669–675.

HISADA, M., BELYAEV, A. G., AND KUNII, T. L. 2002. A Skeleton-based Approach for Detection of Perceptually Salient Features on Polygonal Surfaces. *CGF 21*, 4, 689–700.

HORN, B. 1984. Extended Gaussian Images. *Proceedings of the IEEE 72*, 12, 1671–1686.

HOWLETT, S., HAMILL, J., AND O'SULLIVAN, C. 2005. Predicting and Evaluating Saliency for Simplified Polygonal Models. *ACM Trans. on Applied Perception 2*, 3 (July), 286–308.

HU, J., LU, J., AND TAN, Y. P. 2014. Discriminative Deep Metric Learning for Face Verification in the Wild. *CVPR*, 1875–1882.

HU, J., LU, J., AND TAN, Y. P. 2015. Deep Transfer Metric Learning. *CVPR*, 325–333.

ITTI, L., KOCH, C., AND NIEBUR, E. 1998. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *PAMI 20*, 11, 1254–1259.

ITTI, L., 2000. Models of Bottom-Up and Top-Down Visual Attention. PhD Thesis, California Institute of Technology Pasadena.

JAIN, A., THORMÄHLEN, T., RITSCHEL, T., AND SEIDEL, H.-P. 2012. Material Memex: Automatic Material Suggestions for 3D Objects. *ACM Trans. Graph. 31*, 6 (Nov.), 143:1–143:8.

JÄRVELIN, K., AND KEKÄLÄINEN, J. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. on Information Systems 20*, 4, 422–446.

KALOGERAKIS, E., HERTZMANN, A., AND SINGH, K. 2010. Learning 3D Mesh Segmentation and Labeling. *ACM Trans. Graph. 29*, 4 (July), 102:1–102:12.

KIM, Y., VARSHNEY, A., JACOBS, D. W., AND GUIMBRETIÈRE, F. 2010. Mesh Saliency and Human Eye Fixations. *ACM Trans. on Applied Perception 7*, 2 (Feb.), 12:1–12:13.

KLANK, U., PANGERCIC, D., RUSU, R., AND BEETZ, M. 2009. Real-time CAD Model Matching for Mobile Manipulation and Grasping. *IEEE-RAS Int'l Conf. on Humanoid Robots*, 290–296.

LAU, M., OHGAWARA, A., MITANI, J., AND IGARASHI, T. 2011. Converting 3D Furniture Models to Fabricatable Parts and Connectors. *ACM Trans. Graph. 30*, 4 (July), 85:1–85:6.

LEE, C. H., VARSHNEY, A., AND JACOBS, D. W. 2005. Mesh Saliency. *ACM Trans. Graph. 24*, 3 (July), 659–666.

LIU, Y., ZHU, H., LIU, X., AND WU, E. 2005. Real-time Simulation of Physically based On-Surface Flow. *The Visual Computer 21*, 8-10, 727–734.

LIU, T., HERTZMANN, A., LI, W., AND FUNKHOUSER, T. 2015. Style Compatibility for 3D Furniture Models. *ACM Trans. Graph. 34*, 4 (July), 85:1–85:9.

LIU, Z., WANG, X., AND BU, S. 2015. Human-Centered Saliency Detection. *IEEE Trans. on Neural Networks and Learning Systems*.

MÉRILLOU, S., AND GHAZANFARPOUR, D. 2008. A Survey of Aging and Weathering Phenomena in Computer Graphics. *Computers & Graphics 32*, 2, 159–174.

O'DONOVAN, P., LIBEKS, J., AGARWALA, A., AND HERTZMANN, A. 2014. Exploratory Font Selection Using Crowdsourced Attributes. *ACM Trans. Graph. 33*, 4 (July), 92:1–92:9.

OSADA, R., FUNKHOUSER, T., CHAZELLE, B., AND DOBKIN, D. 2001. Matching 3D Models with Shape Distributions. *Shape Modeling International*, 154–166.

PARIKH, D., AND GRAUMAN, K. 2011. Relative Attributes. *International Conference on Computer Vision (ICCV)*, 503–510.

PLAISIER, M. A., TIEST, W. M. B., AND KAPPERS, A. M. L. 2009. Salient Features in 3-D Haptic Shape Perception. *Attention, Perception, & Psychophysics 71*, 2, 421–430.

PRÉVOST, R., WHITING, E., LEFEBVRE, S., AND SORKINE-HORNUNG, O. 2013. Make It Stand: Balancing Shapes for 3D Fabrication. *ACM Trans. Graph. 32*, 4 (July), 81:1–81:10.

SAHBANI, A., AND EL-KHOURY, S. 2009. A Hybrid Approach for Grasping 3D Objects. *IROS*, 1272–1277.

SAHBANI, A., EL-KHOURY, S., AND BIDAUD, P. 2012. An Overview of 3D Object Grasp Synthesis Algorithms. *Robotics and Autonomous Systems 60*, 3, 326–336.

SAXENA, A., DRIEMEYER, J., KEARNS, J., AND NG, A. Y. 2007. Robotic Grasping of Novel Objects. *NIPS*, 1209–1216.

SCHWARTZBURG, Y., AND PAULY, M. 2013. Fabrication-aware Design with Intersecting Planar Pieces. *CGF 32*, 2, 317–326.

SHILANE, P., AND FUNKHOUSER, T. 2007. Distinctive Regions of 3D Surfaces. *ACM Trans. Graph. 26*, 2 (June), 7.

SHILANE, P., MIN, P., KAZHDAN, M., AND FUNKHOUSER, T. 2004. The Princeton Shape Benchmark. *SMI*, 167–178.

SHTROM, E., LEIFMAN, G., AND TAL, A. 2013. Saliency Detection in Large Point Sets. *ICCV*, 3591–3598.

SONG, R., LIU, Y., MARTIN, R. R., AND ROSIN, P. L. 2014. Mesh Saliency via Spectral Processing. *ACM Trans. Graph. 33*, 1 (Feb.), 6:1–6:17.

SU, H., MAJI, S., KALOGERAKIS, E., AND LEARNED-MILLER, E. 2015. Multi-view Convolutional Neural Networks for 3D Shape Recognition. *ICCV*.

SURAZHSKY, T., MAGID, E., SOLDEA, O., ELBER, G., AND RIVLIN, E. 2003. A Comparison of Gaussian and Mean Curvatures Estimation Methods on Triangular Meshes. *International Conference on Robotics and Automation*, 1021–1026.

TAO, P., CAO, J., LI, S., LIU, X., AND LIU, L. 2015. Mesh Saliency via Ranking Unsalient Patches in a Descriptor Space. *Computers and Graphics 46*, C, 264–274.

VARADARAJAN, K., POTAPOVA, E., AND VINCZE, M. 2012. Attention driven Grasping for Clearing a Heap of Objects. *IROS*, 2035–2042.

WANG, J., SONG, Y., LEUNG, T., ROSENBERG, C., WANG, J., PHILBIN, J., CHEN, B., AND WU, Y. 2014. Learning Fine-Grained Image Similarity with Deep Ranking. *CVPR*, 1386–1393.

WATANABE, K., AND BELYAEV, A. G. 2001. Detection of Salient Curvature Features on Polygonal Surfaces. *CGF 20*, 3, 385–392.

WEI, L., HUANG, Q., CEYLAN, D., VOUGA, E., AND LI, H. 2015. Dense Human Body Correspondences Using Convolutional Networks. *CVPR*.

XU, M., NI, B., DONG, J., HUANG, Z., WANG, M., AND YAN, S. 2012. Touch Saliency. *ACM International Conference on Multimedia*, 1041–1044.

ZAGORUYKO, S., AND KOMODAKIS, N. 2015. Learning to Compare Image Patches via Convolutional Neural Networks. *CVPR*.

ZHOU, Q., PANETTA, J., AND ZORIN, D. 2013. Worst-case Structural Analysis. *ACM Trans. Graph. 32*, 4 (July), 137:1–137:12.

ZIMMER, H., LAFARGE, F., ALLIEZ, P., AND KOBBELT, L. 2014. Zometool Shape Approximation. *Graphical Models 76*, 5, 390–401.