Vision-based Landing Guidance through Tracking and Orientation Estimation

João P. K. Ferreira[1], João P. Pinto[1], Júlia Moura[1], Yi Li[2], Cristiano L. Castro[1,2], Plamen Angelov[2]
[1] Graduate Program in Electrical Engineering, Universidade Federal de Minas Gerais
[2] School of Computing and Communications, Lancaster University

Abstract

Fixed-wing aerial vehicles are equipped with functionalities such as ILS (instrument landing system), PAR (precision approach radar) and, DGPS (differential global positioning system), enabling fully automated landings. However, these systems impose significant costs on airport operations due to high installation and maintenance requirements. Moreover, since these navigation parameters come from ground or satellite signals, they are vulnerable to interference. A more cost-effective and independent alternative for guiding landing is a vision-based system that detects the runway and aligns the aircraft, reducing the pilot's cognitive load. This paper proposes a novel framework that addresses three key challenges in developing autonomous vision-based landing systems. Firstly, to overcome the lack of aerial front-view video data, we created high-quality videos simulating landing approaches through the generator code available in the LARD (landing approach runway detection dataset) repository. Secondly, in contrast to former studies focusing on object detection for finding the runway, we chose the state-of-the-art model LoRAT to track runways within bounding boxes in each video frame. Thirdly, to align the aircraft with the designated landing runway, we extract runway keypoints from the resulting LoRAT frames and estimate the camera relative pose via the Perspective-n-Point algorithm. Our experimental results over a dataset of generated videos and original images from the LARD dataset consistently demonstrate the proposed framework's highly accurate tracking and alignment capabilities. Our approach source code and the LoRAT model pre-trained with LARD videos are available at https://github.com/jpklock2/vision-based-landing-guidance

1. Introduction

An autonomous landing system (ALS) is a technology designed to enable aircraft or unmanned aerial vehicles (UAVs) to land safely and accurately without human intervention [20]. This system typically integrates sensors, navigation systems, and control algorithms to detect the landing site, approach it with the correct trajectory, and manage the descent and touchdown phases. ALS is particularly essential in situations where manual control is challenging or impossible, such as in limited visibility during adverse weather or when operating drones and other unmanned systems. Its applications span various fields, including photography, research, surveillance, defence, and space exploration [17]. Enhancing aircraft autonomy reduces pilots' cognitive load, thereby improving safety in civil aviation [19].

Nowadays, fixed-wing aircraft have functionalities that allow for complete automatic landings. Instrument landing system (ILS) and precision approach radar (PAR) provide radio signals for horizontal and vertical guidance in the final landing approach. DGPS provides the differential global position to guide aircraft through a virtual path. However, these systems come with substantial costs for airport operations due to their high installation and maintenance demands. Moreover, since these navigation parameters come from ground or satellite signals, they are vulnerable to intentional or unintentional interferences such as radiofrequency emitters, solar flares, and spoofing. As a result, they have been primarily used under poor weather conditions, which can affect the landing rate. In good visibility, pilots are still required to visually confirm the runway from a certain distance during the final approach.

A more cost-effective alternative for landing guidance of fixed-wing aerial vehicles is a computer vision-based system equipped with a front-looking camera for detecting the runway and aligning the aircraft. Visual aspects of the landing phase can provide information about the aircraft's relative pose to the designated runway [4]. In contrast to conventional landing functionalities, a Vision-based Autonomous Landing System has two advantages. Firstly, it helps alleviate the cognitive burden on pilots by automating the complex task of landing. This reduces the need for continuous manual control and decision-making. By minimizing the pilot's workload, the system allows them to focus on monitoring and managing other vital aspects of flight, thereby reducing fatigue and the potential for human error. Secondly, it relies primarily on the aircraft's onboard sen-

sors. This reduces the need for costly ground-based installations and maintenance, allowing airports to operate more efficiently. Moreover, the system's ability to function independently of satellite or ground-based equipment means it can be used further in airports that may not have advanced landing systems.

The problem of relative localization for autonomous landing aims at obtaining the airplane camera orientation (yaw, pitch, and roll angles) and the position concerning the designated runway. Solving this problem involves estimating a function that receives as input a sequence of frames representing the landing approach and outputs the camera pose parameters for each frame. A common approach in literature to solve the problem is to divide it into two sequential stages, in which the first is responsible for detecting the runway and the second for solving a well-known problem in photogrammetry called perspective-n-point (PnP) [9], given the 3D reference coordinate system established on the detected runway. If we can precisely detect the runway, we might get enough point correspondences (3D to 2D) to find the camera pose parameters mathematically.

The task of runway detection is particularly challenging since the detection model must be invariant to significant scale differences, aspect ratio changes, weather conditions, and the complexity of distinct airport landscapes. Several studies have addressed the runway detection problem as an object detection task, trying to find the runway corner points/lines or instance semantic segmentation (e.g., aiming and threshold markings) at each frame [4, 8, 15, 21, 22]. In contrast to these former studies, this paper proposes to track the runway through the sequence of frames and then extract the runway corner points for each cropped image resulting from the tracking model. We chose tracking as our framework's main component because of the nature of the detection problem. In a landing video, the runway is the only object of interest and, ideally, remains in the image continuously once it enters the airplane's field of vision. It means that once the model detects the runway for the first time, it no longer needs to search the entire image for the object. Instead, it only needs to focus on the area around the runway, reducing the search cost and improving detection accuracy.

One of the requirements for runway tracking is to have a cohesive and timely sequence of frames representing the approach and landing phases. A recent introduction of the landing approach runway detection dataset (LARD) aims to provide a collection of high-quality aerial images specifically designed for the task of runway detection [7]. Although the original images available in LARD do not form sequences of frames that can be synthesized into a video, a major benefit of LARD is the availability of the generator code that enables the creation of high-quality synthetic front-looking images with automatic annotation (runway border keypoints). So, we have used the LARD generator to create the training set of simulated landing videos for our tracking deep model.

As for the contribution of our study, we propose a framework for autonomous landing with the following features: **(i)** an efficient state-of-the-art tracking algorithm LoRAT [16] to track the runway over the LARD generated videos; the resulting frames from LoRAT are then automatically cropped in a region surrounding the detected runway to facilitate the subsequent task of runway keypoint from Yolov8n-pose. [11]; **(ii)** a collection of videos simulating landing approaches for nearly all airports in the LARD dataset that we will make available in our git repository.

2. Related Works

2.1. Vision-based Runway Detection

Detecting a runway from a front-looking camera during the landing phase of a fixed-wing aircraft presents challenges that need to be addressed to achieve robustness and precision for autonomous landing [1, 13, 14]. In particular, the detection algorithm must handle scale variations, being able to discriminate features from distant to nearby scenes and accommodate potential directional variations. To address these challenges, some recent studies in the literature [4, 5, 15, 21] have approached this problem through modified versions of deep learning models for semantic segmentation of runway instances: runway area, runway aiming point, threshold marks, contour lines. The main point of these studies is that precise delineation of the runway area can be achieved by performing pixel-level classification of runway markings.

In [4], the authors proposed a real-time runway detection model based on an adapted version of MobilnetV3 ConvNet to generate feature lines probability maps in parallel with the semantic output (runway area). The method achieved accurate results for the proposed FS2020 runway dataset. However, some problems were observed in low visibility and near-ground conditions. In [21], the authors chose Yolov8-seg as the essential VALNet component, enriched with additional modules to deal with scale and aspect ratio variations. The proposed deep model presented better results than standard deep models (Mark R-CNN, Yolov8-seg) on their proposed runway landing dataset (RLD). Nevertheless, despite promising results, these studies focused only on runway instance segmentation and, thus, did not infer relative position and orientation from the segmented images.

In contrast to instance segmentation, other studies have approached the problem by modelling bounding boxes and feature point extraction [3, 8, 22] of runways. [8] divided the problem into two stages: runway detection based on ROI (region of interest) plus a sparse coding spatial pyramid

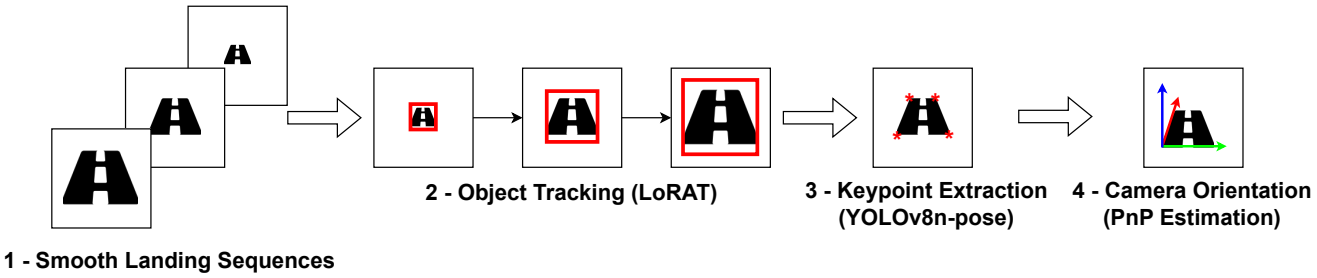

Figure 1. Proposed vision-based landing guidance framework.

matching feature extractor; then, the key points are selected on the detected runway surface, and an orthogonal iteration algorithm is applied to estimate aircraft relative pose. The downside is that the handcrafted feature extractor is sensitive to the pre-set parameters and might suffer under more complex scenes. Moreover, the test bed in [8] considered only a few random frames, which is insufficient for robustness evaluation.

Finally, it is worth noting that private aviation companies such as Airbus [2] and Daedalean [6] are actively researching deep-learning solutions for runway detection using front-view optical sensors. Airbus revealed that it has a deep network that can calculate runway distances and virtual axes for airplane lateral and vertical guidance, simulating the information provided by ILS. Deadelan presented a landing system that can track the runway (showing bounding boxes), and when the aircraft gets closer, the model estimates the parallel lines comprising the runway. Due to confidentiality, the reports provided by these companies do not include detailed technical information.

2.2. Object Tracking

As a deep learning-based computer vision task, object tracking involves leveraging algorithms to automatically detect, monitor, and follow objects within video sequences, enabling real-time analysis and decision-making. Recently, **lo**w-**r**ank **a**daptation (LoRA) [10] freezes the pre-trained model weights while integrating trainable low-rank decomposition matrices into each layer of the Transformer model, significantly cutting down the number of trainable parameters needed for downstream tasks. Inspired by the parameter-efficient fine-tuning (PEFT) in large language models, Lin et al. proposed LoRAT to apply larger vision transformers (ViT) for tracking using laboratory-level resources. In transformer-based trackers, position embeddings are separated into shared spatial embeddings and independent type embeddings. The shared embeddings, which convey the absolute coordinates of multi-resolution images (such as the template and search images), are derived from the pre-trained backbones. Meanwhile, the independent embeddings, which identify the source of each token, are learned from scratch. Furthermore, an anchor-free head is introduced on a multilayer perceptron (MLP) to optimize PEFT, resulting in improved performance with reduced computational cost.

3. Proposed Framework

Our proposed solution is built on a progression of novel contributions that combine the landing guidance framework, depicted in Fig. 1. The framework receives the smooth landing sequence of frames created from the LARD dataset generator (detailed in Section 3.1) as input. Next, a LoRAT model is proposed to track landing runways within bounding boxes in each frame (Section 3.2). As an initial condition, the LoRAT model requires the detected runway bounding box for the first frame to go through the remaining sequence, predicting the following bounding boxes. The first bounding box is detected via Yolov8, and this detection does not need to be highly accurate for LoRAT to work well, as will be later shown in Section 5.1. Afterwards, the key points are extracted via Yolov8n-pose, which is facilitated by cropping every resulting frame from the LoRAT model, as detailed in Section 3.3. Finally, camera orientation is estimated using the runway extracted keypoints and the PnP algorithm (Section 3.4).

3.1. Data Generation

The recently released landing approach runway detection dataset (LARD) [7] aims to provide high-quality aerial images during the approach and landing phases. The dataset primarily comprises images generated using conventional landing trajectories, where the possible positions and orientation of the aircraft during landing are defined within a geometric landing approach cone. The training set consists of 14,433 synthetic images with a resolution of 2448×2648 (or 12,212 images when the domain adaptation sequences are excluded). They are produced from 32 runways in 15 different airports using *Google Earth Studio*. The Test set is composed of 2,212 synthetic images taken from 79 runways in 40 different airports and 103 hand-labeled pictures from real landing footage on 38 runways in 36 different airports.

Although this dataset allows for the training of an object detection system, the images were augmented (e.g., through horizontal flips) and randomly sampled within the geometric landing cone. As a result, it is not possible to properly organize them into a sequence of frames that simulate a real plane landing, which is necessary to train the object tracking component of the framework.

Following the LARD dataset protocol to divide the airports' runways into training and test subsets, we generated smooth landing sequences for each runway: training sequences of around 500 frames and testing sequences of around 35 frames. For the training dataset, we discarded the domain adaptation runways and two training runways with missing information, leading to a dataset of 25 runways in 11 airports, with 11,519 images. For the test sequences, we discarded the real landing footage and used the whole dataset of synthetic images, leading to 79 runways in 40 different airports, with a total of 2,484 images. We used the configuration parameters of the LARD image generator to generate the sequences, as will be shown later in Section 4.1. Our sequences simulate a straight line from the middle of the cone (defined by the LARD authors) up to the runway. As we propose a proof-of-concept for the methodology, we do not include shakiness or lateral plane movement in the sequences.

3.2. Landing Runway Tracking

Although object detection can accurately locate objects in images, the nature of our problem allows for a more cost-effective and straightforward solution. As mentioned earlier, we opt for single-object (runway) tracking instead of object detection. By doing so, we reduce the search area once we only need to focus on the region around the runway. This can also lead to better detection accuracy, which is necessary for the pipeline's later steps that depend upon the whole runway present in the detected bounding box.

Among several tracking models, we use LoRAT [16], which is a single object tracking model that combines a transformer architecture with LoRA [10] training, leading to a fast training model that is accurate and capable of running in a single commercial GPU. The tracking algorithm receives a pair of images along with the bounding box of the target object in the first image and then identifies the same object in the second image. Envisioning a portable application that could run in an embedded system, we use the smallest model, LoRAT-B-224, with 99M parameters and 30 GMACs (roughly 60 GFLOPs).

The model training process can be described as follows: given several image sequences, a random sequence and a random anchor frame are selected. A pair of preceding and following frames are randomly chosen, up to 100 frames away from the anchor. Both frames are cropped around the object's bounding box in the prior frame, sized 2 and 4 times bigger than the bounding box size for the prior and posterior frames, respectively. The crops are feedforwarded in the network, generating a final bounding box, whose error is backpropagated to the network. We use the original training pipeline, with the primary modification of reducing the maximum distance between the frame pairs and the anchor frame to 30 frames instead of 100. We did this reduction to accommodate scale variation. In the final frames of a landing sequence, the runway gradually enlarges in the images. Using a large distance between frames (e.g., 100) can result in the bounding box from the earlier image being much smaller than in the later image, causing the crop to miss part of the object and ultimately reducing the algorithm's performance.

3.3. Keypoint Extraction

When estimating the pose of an aircraft through images, accurate landmarks are essential for achieving robust results that reflect the plane's position and orientation. Since we use object tracking rather than semantic segmentation to achieve real-time results, we lack precise information about the runway's exact position in the images, aside from the bounding box, which may not necessarily encompass all the runway corners. To address this issue, we propose using a keypoint extraction network to identify the actual corners of the runway within a detected bounding box. Keypoint extraction is relatively fast, making it a suitable addition to our real-time solution.

Since landing images simulate the cockpit view, where the pilot must adjust the airplane's direction from a significant distance, the runway often appears small, surrounded by elements that can complicate keypoint detection. Because highly accurate keypoints are critical for the next pipeline stage, we address this challenge by cropping the images around the bounding box identified during runway tracking, adding a margin of 0.2 times the bounding box size. This approach improves accuracy and allows for faster inference. Fig. 2 shows an example of a cropped image.

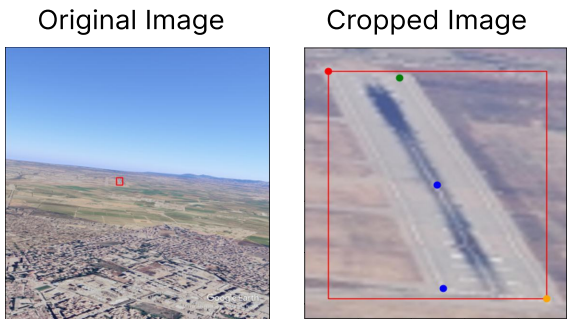

Figure 2. Original image (left) and cropped image (right) based on the runway bounding box predicted from the tracking model.

Deep learning models for keypoint detection are commonly trained for human pose estimation tasks, so they need to be retrained with our dataset to identify the designated keypoints accurately. This study uses the YOLOv8n-pose model, described by Ultralytics [11], designed for pose estimation tasks. This model generates heatmaps to predict the locations of potential keypoints, with the keypoint having the highest probability being selected as the final prediction. Notably, YOLOv8n-pose employs a unified architecture with a single 'head' that simultaneously handles both bounding box and keypoint detection, contrasting with models that utilize separate heads for these distinct tasks.

3.4. Orientation

In the orientation estimation stage of our framework, the detected keypoints are processed by a perspective-n-point (PnP) algorithm [9], which maps a set of 3-dimensional coordinates in a reference object coordinate system to 2-dimensional coordinates projected on the normalized image plane. The reference coordinate system, with axes denoted by X^W, Y^W, and Z^W, is shown in Fig. 3.

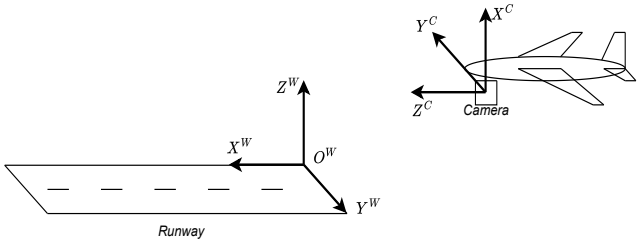

Figure 3. Reference and camera coordinate systems definitions.

To increase the number of correspondence points (3D to 2D) used in the algorithm's calculations, we performed a simple interpolation on the detected keypoints to determine the runway's midpoint across its width. This results in 6 image points and their corresponding 3D object coordinates, which are used as input to the algorithm. These points are based on the runway's width and length, as illustrated in Fig. 4 and listed in Table 1. The runway measurements were obtained using the Google Earth measuring tool.

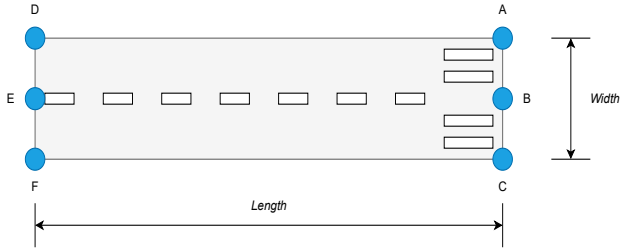

Figure 4. Reference runway points given to the PnP algorithm.

Table 1. Point coordinates from Fig. 4 given to the PnP algorithm.

	A	*B*	*C*	*D*	*E*	*F*
X^W	0	0	0	*Length*	*Length*	*Length*
Y^W	0	*Width*/2	*Width*	0	*Width*/2	*Width*
Z^W	0	0	0	0	0	0

Camera calibration is performed using the ground truth runway corner points and their object coordinates (X^W, Y^W, Z^W) from the training set to obtain the camera's intrinsic parameters, represented by matrix K:

$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3407.6 & 0 & 1291.1 \\ 0 & 3506.6 & 1321.8 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where f_x and f_y are the focal lengths in the x- and y-axis, and u_0 and v_0 are the camera's principal points. With matrix K, the PnP algorithm uses each detected keypoint pixel coordinates $P_i^C = [u, v]^T$, along with its corresponding 3D reference coordinates $P_i^W = [x, y, z]^T \in \{A, B, C, D, E, F\}$, to compute the camera's extrinsic parameters $R_{3\times3}$ and $t_{3\times1}$ through non-linear optimization, to minimize the error of Eq. 2 for all specified points in the image. Matrix R provides estimated values for the camera's yaw, pitch, and roll angles, while the norm of t gives its slant distance to the runway.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R \mid t \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2)$$

4. Experimental Setup

This section further details our proposed tracking dataset and the specific parameters of the models used in this work.

4.1. Dataset

The geometry of a landing example in the LARD dataset is depicted in Fig. 5.

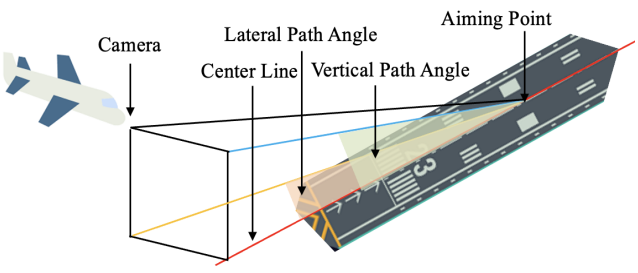

Figure 5. Geometry of a landing.

Three parameters define the aircraft's relative pose to the runway. First, the along-track (slant) distance refers to the distance, measured in nautical miles (NM), between the projection of the aircraft's nose onto the runway's centerline and the Aiming Point. Second, the lateral path angle is the angle formed between the runway's centerline, the line from the Aiming Point, and the projection of the aircraft's nose on the ground. Similarly, the vertical path angle represents the angle formed between the runway's centerline and a plane orthogonal to the ground that intersects the centerline. On the other hand, the aircraft's attitude is defined by its rotation angles: pitch, roll, and yaw. The yaw angle is relative to the runway heading, while pitch and roll are relative to the horizontal plane.

The LARD dataset uses a set of all pairs ⟨positions, attitude⟩ within the ranges of the six ground truths in Table 2. Our tracking dataset consists of generated smooth video sequences, with most parameters set to fixed values equal to the center of the LARD ranges, as shown in Table 2.

Table 2. Ground truths of the generic landing approach cone.

Ground truths	LARD Range	Tracking Range
Along track distance	[0.08, 3] NM	[0.08, 3] NM
Lateral path angle	$[-4, 4]^\circ$	0.0°
Vertical path angle	$[-2.2, -3.8]^\circ$	-3.0°
Yaw	$[-10, 10]^\circ$	0.0°
Pitch	$[-8, 0]^\circ$	-4.0°
Roll	$[-10, 10]^\circ$	0.0°

4.2. Model Configuration

We reproduce or fine-tune different models on a single Nvidia RTX-4090 GPU for different parts of our pipeline:

- LoRAT for tracking: We train a LoRAT-B-224 [1] model using the same parameters as the original authors [16], except for modifying the distance from the anchor frame for random sampling pairs to 30.
- YOLOv8 for keypoint extraction: we train a YOLOv8n-pose [2] with an image size of 640 pixels and a batch size of 4. We trained each network until convergence for a varying number of epochs, as explained in Sections 5.1 and 5.2.

5. Results

5.1. Landing Runway Tracking

Using our proposed LARD-based video dataset for training and testing the tracking model, we trained for 21 epochs using all images in the training dataset. We used 11,519 pairs of images for each epoch, which is the total number of images in the dataset.

Since the tracking algorithm receives an initial input corresponding to the runway detected in the first frame, we perform additional experiments to evaluate the sensitivity of our tracking solution to this first bounding box detection. By varying the position of the bounding box of the first frame, we dislocate it diagonally by 0 (ground truth), 75, 100, 125, and 150 pixels. The mean diagonal size of the first frame bounding boxes for the test dataset is 75 ± 11 pixels, so pixels are a slight overlap with the ground truth is expected at 75, while larger displacements should not overlap. We dislocate all bounding boxes to the upper right corner to keep consistency. Finally, we evaluate our model using the standard One-Pass Evaluation (OPE) metrics [18]: Success Score (SUC), Precision (P), Normalized Precision (P_{Norm}), Success Rate at IoU >= 0.5 ($SR_{0.5}$) and Success Rate at IoU >= 0.75 ($SR_{0.75}$). The mean results for every runway of each experiment in the test dataset can be seen in Table 3, where PD indicates the diagonal Pixel Displacement of the bounding box towards the upper left corner.

Table 3. Landing tracking accuracy (in %) for different overlaps of the first frame with the ground truth.

PD	SUC	P	P_{Norm}	$SR_{0.5}$	$SR_{0.75}$
0	**80.12**	**88.11**	**93.64**	**92.84**	**85.77**
75	79.79	**88.11**	93.41	92.77	85.73
100	78.50	87.10	91.25	90.67	82.85
125	67.56	73.95	79.25	77.52	71.65
150	31.07	34.17	40.47	34.44	30.82

The results of the ground truth case indicate that the network can track runways effectively, achieving an SUC score of 80.1% and a Precision of 88.1%. When the bounding box was displaced by 75 pixels diagonally, the results remained very close to the ground truth, meaning the network could track the runway despite the displacement. Similar behaviour was observed for smaller displacements, so we did not report results below 75. Performance began to decline at around 100 pixels, with significant drops occurring at 125 and 150 pixels. This highlights the approach's limitations: the object detector must be accurate enough to locate the runway within 100 pixels (diagonally) of its actual position. More significant deviations may result in decreased algorithm performance.

To validate our network choice, we performed an additional experiment where we trained YOLOv8 for object detection (without keypoints) for 338 epochs, until convergence, using the default parameters specified in Section 4.2 and our proposed tracking dataset. We then used the model as a substitute for our tracking network. We then extracted the same tracking metrics for the YOLO detections over

[1] https://github.com/LitingLin/LoRAT

[2] https://github.com/ultralytics/ultralytics/issues/1915

time and listed them in Table 4.

Table 4. Comparison of different tracking algorithms.

Network	SUC	P	P_{Norm}	$SR_{0.5}$	$SR_{0.75}$
LoRAT	**80.12**	**88.11**	**93.64**	**92.84**	**85.77**
YOLOv8	74.51	87.82	91.62	91.91	62.88

The results show that LoRAT obtained a better result in every metric. Since we need an accurate tracker for improved keypoint detection results, this demonstrates that a model proper for the tracking task suits our pipeline better than an object detection model.

5.2. Keypoint Extraction

To assess the impact of the image cropping in our solution, we trained the Yolov8n-pose model on the LARD train dataset twice, with images cropped around the bounding box for 160 epochs, and with the same dataset with full-sized images for 229 epochs. This experimental setup involved the evaluation of the model performance using precision (P), recall (R), and mean Average Precision metrics (mAP^{50}/$mAP^{50\text{-}95}$), following the validation procedure outlined in [12]. The are presented in Table 5.

Table 5. Comparison of yolov8n-pose results in cropped and non-cropped images

Crop	P	R	mAP^{50}	$mAP^{50\text{-}95}$
✗	95.45%	92.95%	95.62%	88.15%
✓	**98.92%**	**98.96%**	**98.49%**	**94.59%**

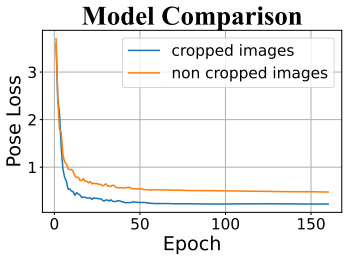

Figure 6. Validation losses for keypoint detection model trained with cropped and full-size images.

The results obtained from training on the cropped dataset demonstrated superior performance across all evaluated metrics, and, as shown in Figure 6, both models converged within the number of epochs we allowed them to execute. This suggests that cropping the image around the bounding box identified during the tracking phase is critical for ensuring precise outcomes. An illustrative example of these results is presented in Fig. 7, where the keypoints have been scaled to the original image dimensions for enhanced visualization.

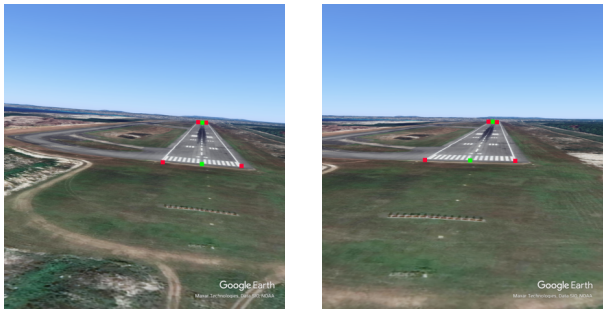

Figure 7. Keypoints detection results with red bullets.

5.3. Orientation

We obtained results for estimating the camera orientation and the slant distance to the runway across four distinct scenarios, using the keypoints detected by the YOLO network and the ground truth points of runway corners. The evaluation was conducted on 29 images corresponding to runway 35 of the MDSD airport from the original LARD test dataset, on 31 frames of a smooth landing video sequence generated for the same runway, on 31 images of runway 34R from the RJAA airport from the original LARD test dataset; and on 30 frames of a smooth landing video sequence generated for the same runway. Evaluation metrics include the error between the aircraft's estimated and ground truth parameters.

Figs. 8 to 11 present the orientation and slant distance estimation errors for each image across different scenarios, comparing ground-truth-as-input estimations with detected-keypoints-as-input estimations. The corresponding numerical results, expressed as mean and standard deviation errors across all images and scenarios, are detailed in Table 6.

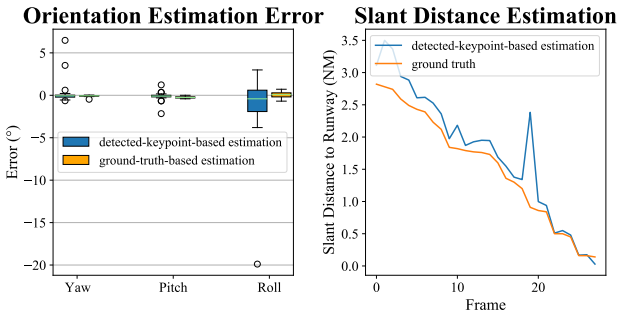

Figure 8. Results of Runway 35 - MDSD Airport (LARD test set).

The results show that, in all cases, the average errors are close to zero degrees and tend to align with estimates using the ground truth corners of the runways, which are the highest standard baselines for this method. In all scenarios, the roll angle exhibited the highest standard deviation, highlighting the sensitivity of this parameter to minor fluctuations in the detected keypoint positions. The estimated

Table 6. Orientation and slant distance estimation errors across different scenarios, using the ground truth (gt) and detected keypoints (dkp) as input; symbols I and V refer to images from the LARD test set and our smooth video generated sequence, respectively.

Scenery	Yaw Error (°)		Pitch Error (°)		Roll Error (°)		Slant Dist. Error (NM)	
	gt	dkp	gt	dkp	gt	dkp	gt	dkp
I_{MDSD}	**-0.07 ± 0.09**	0.25 ± 1.40	-0.25 ± 0.10	**-0.12 ± 0.51**	**-0.03 ± 0.36**	-1.03 ± 4.04	**0.03 ± 0.02**	0.25 ± 0.29
V_{MDSD}	**-0.06 ± 0.01**	-0.07 ± 0.21	-0.27 ± 0.02	**-0.04 ± 0.20**	**0.03 ± 0.08**	-0.83 ± 2.76	**0.03 ± 0.01**	0.32 ± 0.29
I_{RJAA}	**-0.04 ± 0.06**	-0.38 ± 1.48	**-0.15 ± 0.11**	0.21 ± 0.68	**0.22 ± 0.43**	0.27 ± 2.17	**0.05 ± 0.04**	0.52 ± 0.84
V_{RJAA}	**-0.02 ± 0.01**	-0.14 ± 0.23	-0.18 ± 0.02	**0.09 ± 0.31**	**0.07 ± 0.07**	2.02 ± 1.61	**0.07 ± 0.04**	0.40 ± 0.29

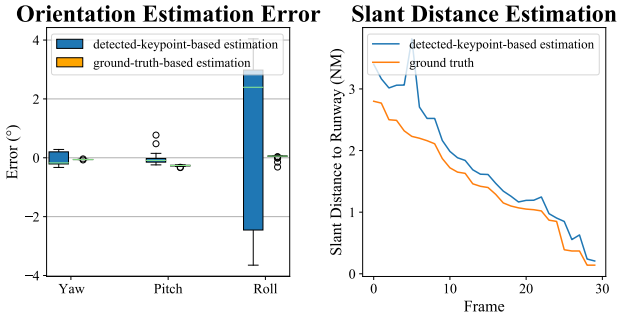

Figure 9. Results of Runway 35 - MDSD Airport (generated video sequence).

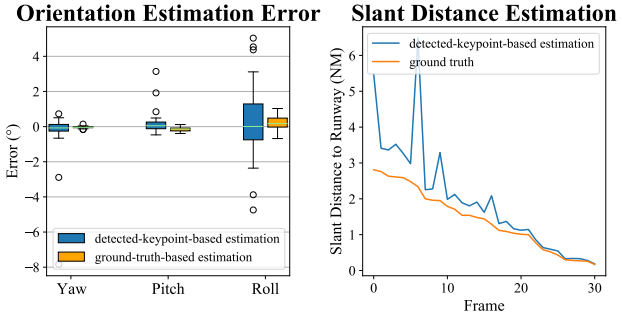

Figure 10. Results of Runway 34R - RJAA Airport (LARD test set).

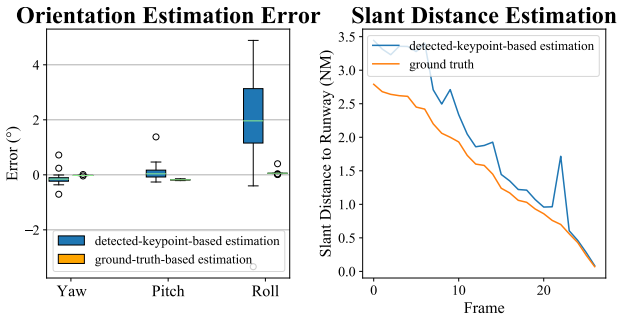

Figure 11. Results of Runway 34R - RJAA Airport (generated video sequence).

slant distances per frame are relatively close to the ground truth despite a few outliers, and the estimates generally improve as the aircraft approaches the runway. This improvement is attributed to better runway visibility and more accurate keypoint detection as proximity increases.

6. Conclusion

This paper proposes a vision-based landing guidance framework that mainly relies on the LoRAT transformer model to track runways within bounding boxes in each frame. We created smooth video sequences simulating landing approaches through the image generator code available in the repository of the LARD dataset. This was a requirement for LoRAT deep model training. We have also proposed cropping the area surrounding the resulting runway bounding boxes predicted by LoRAT, which leads to performance gains of the Yolov8n-pose keypoint extractor. As for the pose estimation via the PnP algorithm, the proposed framework achieved accurate results despite some outliers.

Although we have obtained very high metrics (all above 90%) with the keypoint extraction model (Yolov8n-pose), we have observed that the PnP algorithm for relative pose estimation is susceptible to minimal deviations in the position of the extracted keypoints, which leads us to consider future alternatives. A possible solution is to increase the number of uncorrelated keypoints given to PnP. Moreover, we could enrich the keypoint estimates by either modifying the LoRAT tracking model to predict keypoint displacements frame by frame or by performing sensor fusion (our proposed visual estimator + onboard INS) through a filtering technique, like the extended Kalman filter.

Finally, we intend to extend the experiments to test the robustness of our pipeline over more challenging landing sequences involving airplane manoeuvres, shakiness, and real footage. We also plan to test our approach on other publicly available datasets, such as FS2020 [4] and RLD (runway landing dataset) [21], to compare it with recent state-of-the-art methods from the literature.

Acknowledgment

This work is supported by ELSA – European Lighthouse on Secure and Safe AI funded by the European Union under grant agreement No. 101070617 and by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES).

References

[1] B. Ajith, Sudam D. Adlinge, Sudin Dinesh, U. P. Rajeev, and E. S. Padmakumar. Robust Method to Detect and Track the Runway during Aircraft Landing Using Colour segmentation and Runway features. *Proceedings of the International Conference on Trends in Electronics and Informatics, ICOEI 2019*, 2019-April:751–757, 2019. 2

[2] J. Au, D. Reid, and A. Bill. Challenges and opportunities of computer vision applications in aircraft landing gear. In *2022 IEEE Aerospace Conference (AERO)*, pages 1–10. IEEE, 2022. 3

[3] B. Cai, Z. Jiang, H. Zhang, D. Zhao, and Y. Yao. Airport detection using end-to-end convolutional neural network with hard example mining. *Remote Sensing*, 9(11), 2017. 2

[4] M. Chen and Y. Hu. An image-based runway detection method for fixed-wing aircraft based on deep neural network. *IET Image Processing*, 18(8):1939–1949, 2024. 1, 2, 8

[5] W. Chen, Z. Zhang, L. Yu, and Y. Tai. Bars: a benchmark for airport runway segmentation. *Applied Intelligence*, 53(17):20485–20498, apr 2023. 2

[6] Daedalean. Visual landing guidance. https://daedalean.ai/tpost/inxg7y7671-land-safely-daedaleans-visual-landing-sy. Accessed: 2024-09-03. 3

[7] M. Ducoffe, M. Carrere, L. Féliers, V. Mussot A. Gauffriau, C. Pagetti, and T. Sammour. LARD - landing approach runway detection - dataset for vision based landing. *arXiv preprint arXiv: 2304.09938*, 2023. 2, 3

[8] Y. Fan, M. Ding, and Y. Cao. Vision algorithms for fixed-wing unmanned aerial vehicle landing system. *Science China Technological Sciences*, 60, 02 2017. 2, 3

[9] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Readings in Computer Vision*, pages 726–740, 1987. 2, 5

[10] E. J Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *Proceedings of International Conference on Learning Representations (ICLR)*, 2022. 3, 4

[11] G. Jocher, A. Chaurasia, and J. Qiu. Ultralytics yolov8, 2023. 2, 5

[12] T. D. Kim, N. Dao, D. Tran, A. L. Tran, and D. Pham. Surgical tool detection and pose estimation using yolov8-pose model: A study on clipper tool. In *2024 9th International Conference on Integrated Circuits, Design, and Verification (ICDV)*, pages 225–229, 2024. 7

[13] D. Kordos, P. Krzaczkowski, and E. Zesławska. Vision system measuring the position of an aircraft in relation to the runway during landing approach. *Mathematical Problems in Engineering*, 23(3):1560, 2023. 2

[14] Y. Li, P. Angelov, and N. Suri. Self-supervised representation learning for adversarial attack detection. *Proceedings of European Conference on Computer Vision (ECCV)*, 2024. 2

[15] Y. Li, P. Angelov, Z. Yu, A. Pellicer Alvaro, and N. Suri. Federated adversarial learning for robust autonomous landing runway detection. *Proceedings of International Conference on Artificial Neural Networks (ICANN)*, 2024. 2

[16] L. Lin, H. Fan, Z. Zhang, Y. Wang, Yong Xu, and H. Ling. Tracking meets LoRA: Faster training, larger model, stronger performance. *Proceedings of European Conference on Computer Vision (ECCV)*, 2024. 2, 4, 6

[17] U. Martinez-Hernandez, V. Cedeno-Campos, and A. Rubio-Solis. Active visual object exploration and recognition with an unmanned aerial vehicle. *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2019. 1

[18] M. Müller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem. TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11205 LNCS:310–327, 2018. 6

[19] X. Pang, N. Zhao, J. Tang, C. Wu, D. Niyato, and K. Wong. IRS-assisted secure UAV transmission via joint trajectory and beamforming design. *IEEE Transactions on Communications*, 70(2):1140 – 1152, 2022. 1

[20] P. Wang, Z. Yang, X. Chen, and H. Xu. A Transformer-based method for UAV-view geo-localization. *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2023. 1

[21] Q. Wang, W. Feng, H. Zhao, B. Liu, and S. Lyu. Valnet: Vision-based autonomous landing with airport runway instance segmentation. *Remote Sensing*, 16(12), 2024. 2, 8

[22] S. Yin, H. Li, and L. Teng. Airport Detection Based on Improved Faster RCNN in Large Scale Remote Sensing Images. *Sensing and Imaging*, 21(1):49, Oct. 2020. 2