Landing Aircraft Attitude Estimation From Vanishing Point and Horizon Line Regression

João Paulo Lara Pinto*, Gabriel Lott*, João Pedro Klock Ferreira*, Júlia Santos Moura*, Cristiano Leite de Castro*
*Graduate Program of Electrical Engineering, Universidade Federal de Minas Gerais

***Abstract*—Vanishing points and horizon lines carry valuable perspective information about a scene. When the vanishing point location, horizon line slope and camera intrinsic matrix are known, it is possible to infer the camera pose, making these features play important roles in applications such as 3D reconstruction and autonomous navigation. Recent works have employed deep Convolutional Neural Networks (CNN) to estimate these parameters with good performance, but most are limited to ground-level camera scenarios. In this work, we propose an approach to regress the vanishing point coordinates of a runway from frontal images captured by an airplane during the landing phase, as well as to estimate the horizon line slope, using a deep CNN. Furthermore, we leverage vanishing point perspective geometry to infer three degrees of freedom of the airplane—the yaw, pitch and roll angles. The results demonstrate that our approach achieves accurate regression of both the vanishing point and horizon line slope with fast inference times. Moreover, the estimated airplane angles, despite some outliers, show promise for applications in airplane orientation within Autonomous Landing Systems.**

***Index Terms*—vanishing point, horizon line, Convolutional Neural Networks, regression, camera orientation, projective geometry.**

I. Introduction

Generally, when looking at landscape photographs, it is possible to infer whether the camera was tilted sideways or angled upward or downward based on the position and orientation of the horizon line. If the horizon appears tilted, it indicates that the camera was rotated sideways. If the horizon is positioned high in the image, the camera was likely angled downward; conversely, if it is low in the frame, the camera was probably tilted upward. Similarly, the perspective of the picture can be interpreted by observing how parallel lines in the real world behave in the image, as they typically converge toward a single point—called the vanishing point—revealing the camera's orientation relative to the scene.

Vanishing point detection and horizon line estimation both play an important role in fields that rely on understanding the camera's orientation, such as 3D scene reconstruction [1], [2] and autonomous navigation [3]. In particular, these techniques can be leveraged to estimate camera orientation in the specific context of aircraft pose estimation during vision-based autonomous landing, where reliable information about runway alignment and approach angle is critical for ensuring safe and precise maneuvers. This is especially important given that the approach and landing phases are among the most accident-prone stages of flight [4], with a significant proportion of incidents caused by human error.

Classical approaches to horizon line detection typically rely on edge detection and line fitting techniques to identify dominant linear structures corresponding to the horizon. Similarly, traditional vanishing point detection algorithms often depend on extracting and clustering line segments that intersect at a common point. However, these methods can struggle in complex or cluttered environments where clear linear features are absent, leading to unreliable estimations. Additionally, the processing time of such algorithms can be significant, making them unsuitable for real-time applications that require high frame rates.

To address these challenges, recent approaches use Deep Learning techniques to estimate the parameters of the vanishing point [5], [6] and horizon line [7], [8], achieving high accuracy with low inference times. However, most of these methods focus on estimating either the vanishing point or the horizon line independently rather than both simultaneously. Additionally, they are typically designed for ground-level images featuring prominent, easily detectable lines from which to extract the vanishing point. In [9], the authors employ a deep Convolutional Neural Network (dCNN) to estimate both the vanishing point coordinates and the horizon line slope from frontal images captured by a landing fixed-wing aircraft. However, they did not provide implementation details, as their work is proprietary, nor did they make their dataset publicly available.

In this work, we propose a regression network based on MobileNetV2 [10] to estimate both the vanishing point coordinates of a runway and the horizon line slope from frontal images captured by a landing airplane. We compare the performance of our approach with classical computer vision methods for these tasks. Furthermore, we leverage vanishing point perspective geometry to analytically compute the camera's yaw, pitch, and roll angles relative to the runway during the landing phase, demonstrating that the proposed method could serve as a cost-effective, redundant system to complement onboard orientation sensors, with potential applications in Autonomous Landing Systems.

II. Related Works

Numerous studies have focused on estimating the camera pose directly from the captured image [11]. The main approaches can be categorized into structure-based methods,

which map 2D pixel coordinates of image features to known 3D scene coordinates; end-to-end methods, where deep neural networks are trained to directly regress the camera pose from the input image [12]–[15]; and hybrid methods, which leverage deep learning to infer aspects of the scene structure before applying structure-based techniques to estimate the pose [16].

In structure-based methods—which generally achieve the highest accuracies—the typical approach first involves identifying image descriptors that correspond to known 3D object-space coordinates. Subsequently, a Perspective-n-Point (PnP) algorithm is employed to compute camera pose hypotheses based on these correspondences [17]. A key challenge with such methods is that, when only a few known scene points are available, relying solely on the resulting correspondences can make the estimation process more sensitive to outliers.

A robust alternative approach for this task is to combine vanishing point coordinates with the horizon line equation to recover the camera pose through projective geometry. This method is somewhat analogous to structure-based approaches, although it allows for the recovery of only the camera's orientation, not its position. Common methods for obtaining the vanishing point include detecting dominant lines in the image, computing their intersections, and performing soft voting to identify the intersection most likely to correspond to the vanishing point [18], [19]. More recent approaches leverage deep Convolutional Neural Networks (dCNNs) to regress the vanishing point coordinates [5], [6], [9], achieving superior results compared to classical computer vision methods. In [9], the network also regresses the horizon line slope along with other geometric parameters of the runway in perspective, which are then used to compute the airplane's orientation and position. However, since it is proprietary work, the implementation details of the regressor are not disclosed.

Horizon line estimation has been extensively studied in maritime navigation [20]. However, due to the significant visual differences between open sea environments and more complex settings such as aerial images or urban landscapes, direct methods developed for maritime applications often fail to perform effectively in these scenarios. An alternative approach involves applying a minimal path algorithm [21], followed by filtering techniques to eliminate vertical deviations and outliers, along with RANSAC [22] for robust model fitting. Nevertheless, for complex landscapes, the use of convolutional networks to regress the horizon line slope [9] [7] remains the state of the art for this task.

III. Perspective Geometry of Vanishing Points

Assuming a pinhole camera model and considering a line lying on the ground plane of the world coordinate system, let us define a set of equally spaced points along this line as X_i, with $i \in 1, \ldots, 5$. Although these points are uniformly distributed throughout the world, their projections onto the image plane—determined by the rays passing through each point and converging at the camera's optical center O_c—become progressively closer together, decreasing monotonically in spacing [23]. As illustrated in Figure 1, when the distance of a point along this line tends to infinity, the vanishing point of the line is obtained by intersecting the image plane with the family of rays parallel to the line that converge at O_c.

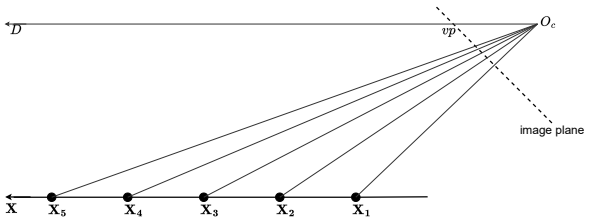

Fig. 1. Perspective geometry of vanishing points. The projections of equally spaced points along a line, X_i, $i \in 1, \ldots, 5$, onto the image plane monotonically decrease in spacing. The vanishing point vp is defined by the intersection of the image plane with rays of direction D, parallel to the line, that converge at the camera's optical center O_c.

Algebraically, points along a line in the 3D object space, with direction $\mathbf{D} = [\mathbf{d}^\top, 0]^\top$, can be expressed as $\mathbf{X}(\lambda) = \mathbf{X}_0 + \lambda\mathbf{D}$, where $\mathbf{X}_0$ is a point on the line, and $\lambda \in \mathbb{R}$ (Figure 2). Under a projective camera model $\mathbf{P} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}]$, where $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the intrinsic calibration matrix, and $[\mathbf{I} \mid \mathbf{0}]$ consists of the 3×3 identity matrix (representing no rotation) augmented with a 3×1 zero vector (representing no translation), the projection of $\mathbf{X}(\lambda)$ onto the image plane can be written as

$$\mathbf{x}(\lambda) = \mathbf{P}\mathbf{X}(\lambda) = \mathbf{P}\mathbf{X_0} + \lambda\mathbf{P}\mathbf{D} = \mathbf{x_0} + \lambda\mathbf{K}\mathbf{d} \qquad (1)$$

where $\mathbf{x_0}$ is the projection of the point $\mathbf{X_0}$. The vanishing point is obtained as $\lambda \to \infty$, resulting in

$$vp = \lim_{\lambda\to\infty} \mathbf{x}(\lambda) = \mathbf{K}\mathbf{d} \qquad (2)$$

This result shows that the vanishing point depends solely on the direction of the line in world coordinates, and not on its specific position given by $\mathbf{X_0}$.

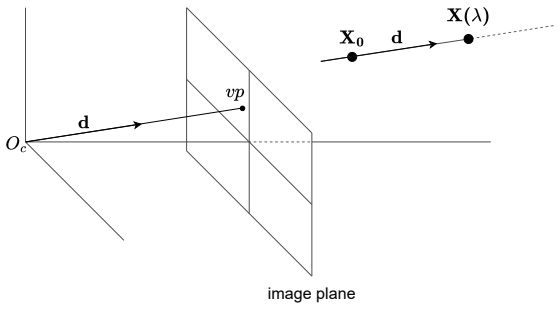

Fig. 2. Projection of 3D space onto the camera's image plane. The vanishing point vp is at the intersection of the image plane with rays of direction $\mathbf{d}$ through O_c. The line in 3D world coordinates can be parametrized as $\mathbf{X}(\lambda) = \mathbf{X_0} + \lambda\mathbf{D}$, where $\mathbf{X}_0$ is a point on the line and $\mathbf{D} = (\mathbf{d}^\top, 0)$. Adapted from [23].

IV. METHODOLOGY

The proposed approach for estimating the vanishing point of the landing runway side lines and the slope of the horizon line employs a MobileNetV2-based feature extractor [10], followed by a regression module. MobileNetV2 was chosen due to its compact architecture, which offers a good trade-off between performance and inference time, making it well-suited for real-time embedded applications, as required by this method. In addition, its proven effectiveness in similar tasks [9] further motivated its choice. Furthermore, MobileNetV2 contains fewer pooling layers compared to other common convolutional neural network (CNN) architectures. As noted in [6], excessive pooling can potentially degrade features relevant to vanishing point estimation. Moreover, [6] highlights that fully connected layers can compromise the preservation of geometric information from the input data. To mitigate this issue, the regression head built on top of the MobileNetV2 feature extractor employs only a single fully connected layer, which is the only such layer in the entire network. Figure 3 illustrates the proposed network architecture.

The MobileNetV2 feature extractor requires a square input of size 224×224. To meet this requirement, the non-square dataset images are scaled, and the corresponding ground truth vanishing point coordinates and horizon line slope are adjusted accordingly during training, resulting in $[X, Y]$ and α respectively. The feature extractor comprises all layers up to and including the pointwise (1×1) convolution layer, following the sequence of bottleneck layers in MobileNetV2, producing a feature map of shape 7×7×1280. Subsequently, an adaptive global average pooling layer is applied, reducing the feature map to a 1×1×1280 tensor. This output is then flattened and passed through a fully connected layer, producing the regressed vector $[\hat{X}, \hat{Y}]$ representing the vanishing point coordinates in the scaled image, along with $\hat{\alpha}$, the estimated horizon line slope. Finally, the coordinates are rescaled to match the original image dimensions, yielding $[\hat{vp}_x, \hat{vp}_y]$ as the estimated vanishing point coordinates in the original image space.

To train the regression network to minimize both the distance between the predicted and true vanishing point coordinates, as well as the error in the predicted horizon line slope, the loss function was formulated as the sum of the squared errors of each output variable. However, since the α values are a few orders of magnitude smaller than the vanishing point coordinates (which correspond to pixel positions), a scalar weight β was introduced to balance their relative contributions to the overall loss. The resulting loss function is defined as

$$\mathcal{L}_{\text{total}} = \frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{X}_i)^2 + (Y_i - \hat{Y}_i)^2 + \beta(\alpha_i - \hat{\alpha}_i)^2 \quad (3)$$

where N is the batch size and the subscript i represents the i-th sample in the batch. Empirical experiments were conducted to tune the value of β, and it was found that setting $\beta = 10$ yielded good performance, so this value was used in all training experiments reported in this paper.

The vanishing point coordinates and the horizon line slope estimated by the trained network are subsequently used to infer three degrees of freedom of the camera mounted on the aircraft: the yaw, pitch and roll angles. The roll angle ϕ, expressed in degrees, is computed directly from the estimated slope

$$\phi = -\frac{180}{\pi}\arctan(\hat{\alpha}) \quad (4)$$

The yaw angle ψ and pitch angle θ are then estimated using the vanishing point perspective geometry relations described in Section III. As previously discussed, the vanishing point vp of a set of parallel lines in world coordinates is defined by the intersection between the image plane and the family of light rays parallel to those lines that converge at the camera's optical center O_c.

Since pitch estimation depends only on the vertical displacement of the vanishing point, and yaw estimation depends exclusively on its horizontal displacement, we treat each axis of the image plane separately when calculating these angles. The geometric relationships used for this purpose are illustrated in Figure 4. For pitch estimation, vp denotes the y-coordinate of the regressed vanishing point, c is the y-coordinate of the image principal point, f is the focal length along the y-axis, and a corresponds to the pitch angle θ to be estimated. Similarly, for yaw estimation, vp and c represent the x-coordinates of the vanishing point and the principal point, respectively, f is the focal length along the x-axis, and a is the yaw angle ψ to be estimated. Both the focal lengths and the principal point coordinates are obtained from the calibrated camera intrinsic matrix $\mathbf{K}$.

With these relations established, one could easily solve for θ and ψ using basic trigonometry. However, the geometric relations depicted in Figure 4 do not account for the roll angle, which can influence the x and y coordinates of the vanishing point in the image. To address this, we first apply a rotation matrix $\mathbf{R}(-\phi)$ to the vanishing point, rotating it around the camera's principal point to effectively compensate for the roll, resulting in a scenario with zero roll.

The rotated vanishing point can then be used to solve for θ and ψ with Equations 5 to 8

$$\mathbf{R}(-\phi) = \begin{bmatrix} cos(-\phi) & -sin(-\phi) \\ sin(-\phi) & cos(-\phi) \end{bmatrix} = \begin{bmatrix} cos\phi & sin\phi \\ -sin\phi & cos\phi \end{bmatrix} \quad (5)$$

$$\begin{bmatrix} \hat{vp}'_x \\ \hat{vp}'_y \end{bmatrix} = \mathbf{R}(-\phi)\begin{bmatrix} \hat{vp}_x - c_x \\ \hat{vp}_y - c_y \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \quad (6)$$

$$\theta = arctan\left(\frac{\hat{vp}'_y - c_y}{f_y}\right) \quad (7)$$

$$\psi = arctan\left(\frac{\hat{vp}'_x - c_x}{f_x}\right) \quad (8)$$

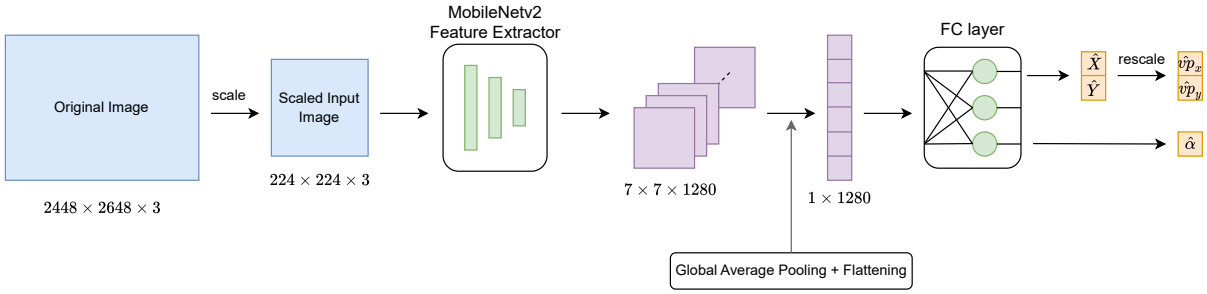

Fig. 3. Proposed regression network architecture.

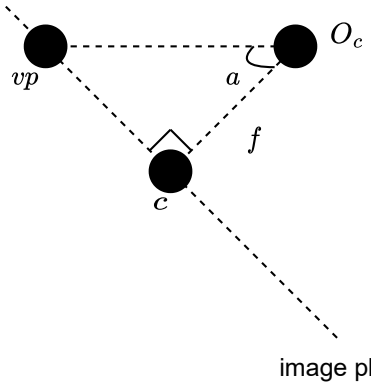

Fig. 4. Geometric relationships based on vanishing point perspective geometry, illustrating the camera optical center O_c, focal length f, principal point c, vanishing point vp, and camera angle a. For pitch calculation ($a = \theta$), the y-axis components of the relevant coordinates and focal length are used, whereas for yaw calculation ($a = \psi$), the x-axis components are considered.

V. Experimental Setup

This section describes the experiments conducted, the dataset employed, and the procedure used to obtain the target ground truths for training.

A. Dataset

Although there are some publicly available vanishing point datasets [6], it is difficult to find one that specifically covers the scenario of a front-facing view from a landing aircraft—let alone one that also includes horizon line annotations. The most common types of landing runway datasets [24]–[26] typically provide runway semantic masks or runway corner keypoints, primarily intended for tasks such as runway segmentation or detection. Moreover, manually annotating vanishing points and horizon lines across tens of thousands of images is highly labor-intensive and time-consuming. Therefore, in this work, the required labels were automatically derived from the existing annotations provided by the dataset used in this work.

TABLE I
Ground Truths of the Generic Landing Approach Cone.

Ground truths	LARD Range
Along track distance	[0.08, 3] NM
Lateral path angle	[-4, 4]°
Vertical path angle	[-2.2, -3.8]°
Yaw	[-10, 10]°
Pitch	[-8, 0]°
Roll	[-10, 10]°

The Landing Approach Runway Detection (LARD) dataset [24] was selected for the experiments. It consists of high-quality synthetic frontal-perspective images of an aircraft during landing. The images were generated following conventional landing trajectories, with the aircraft's orientation and position ranges defined according to a typical geometric landing cone, as shown in Table I. The dataset comprises 14,433 images at a resolution of 2448×2648 pixels, divided into a training set of 12,212 images captured from 32 runways across 15 different airports, and a test set of 2,212 images captured from 79 runways in 40 different airports. Regarding the labels, the dataset provides the image coordinates of the four runway corner points, as well as the aircraft's orientation—expressed in yaw, pitch, and roll angles—and its position along the runway aiming point.

The ground truth horizon line slope was approximated by calculating the slope of the line connecting the two foremost runway corners. While this slope does not always exactly correspond to the true horizon line angle, for the range of airplane poses present in the dataset, it provides a sufficiently accurate approximation. Additionally, minor noise in the labels can have a beneficial regularization effect in model learning. The ground truth vanishing points were determined by extending the runway side lines and identifying their intersection point within the image. Examples of the generated ground truth labels are presented alongside the results in Figure 5.

B. Experiments

The experiments were conducted on Google Colab, with Python 3.11, PyTorch 2.6, and CUDA 12.4. The hardware environment included access to a cloud-based Tesla T4 GPU with 12GB of RAM. The regression network was initially trained to convergence on a validation set—composed by a

subset of 1,817 images taken from the training set—using a learning rate of $\eta = 10^{-4}$, followed by a second training phase with $\eta = 10^{-5}$ until further convergence. The batch size was set to 64, and the Adam optimizer was used for all experiments.

From the estimated vanishing point and horizon line slope, we infer the aircraft's pitch and roll angles, as detailed in Section IV. We compare the results and inference times of our model with those obtained using classical computer vision algorithms for vanishing point and horizon line detection. For vanishing point detection, we adapt the method from [19], which detects line segments in the image, extends them, and computes their intersections, selecting the point with the most intersections as the vanishing point. For horizon line estimation, we adopted the approach from [21], which uses a minimum path algorithm to estimate the line that separates the ground and the sky. In Section VI, we refer to these methods as 'baseline' for their respective tasks. The vanishing point position error is measured as the Euclidean distance to the ground truth in pixels. The horizon line slope error is quantified as the absolute difference in degrees. Similarly, yaw, pitch, and roll errors are reported in degrees, computed as the simple difference between the estimated and ground truth values.

VI. Results

Figure 5 presents a qualitative comparison between the results obtained by the proposed regression network and the baselines across various airport scenarios in the test set. The dot represents the vanishing point, and the line represents the horizon line. As shown, in most cases—regardless of the aircraft's distance from the runway—the vanishing point estimated by our method closely aligns with the ground truth. Moreover, the estimated horizon line is almost perfectly aligned with the ground truth horizon, indicating that the network effectively regressed both the vanishing point coordinates and the horizon line slope. One scenario that notably challenges the method involves horizons filled with mountains of irregular heights, which can create the illusion that the horizon is more inclined than it actually is. In such cases, the regression network exhibited more noticeable errors. Nevertheless, even in images where the runway is barely visible, the network was able to produce satisfactory results.

Regarding the baseline methods, it is evident that they only perform well in situations where the aircraft is very close to the runway. This outcome is expected, as these methods were originally designed for the constrained scenario of near-ground-level cameras capturing images with clearly defined lines, such as roads. In some cases, the baseline vanishing point method failed to detect the vanishing point within the image boundaries, as illustrated by the absence of the yellow dot in certain examples in Figure 5. Although the baseline horizon line estimation method was able to approximate the horizon slope reasonably well in some instances (as seen by the slope of the line, not its position), it did not perform as accurately as the proposed method.

TABLE II
95th percentile of estimation errors for each parameter across the different methods.

Method	95th percentile of estimation errors	
	Vanishing Point (pixels)	Horizon Line slope (°)
Baseline	1,649	29.59
Ours	**150**	**3.27**

TABLE III
Comparison of inference times.

Method	Inference time (ms)
Baseline	3,080.61 ± 2,012.86
Ours	**9.56 ± 4.23**

Figure 6 compares the error distribution over the test set images. From the histograms, it is evident that most of the errors produced by our method are concentrated at low values, whereas the baseline method's errors are more widely distributed, extending to higher values. Table II presents a comparison of the 95th percentile of the errors for each method, indicating the thresholds below which 95% of the errors fall. The baseline vanishing point estimation method exhibits a 95th percentile error of 1,649 pixels, corresponding to 67.36% of the image width, which is substantially higher than the 150 pixels (6.17% of the image width) achieved by our method, representing a considerable improvement. Regarding the horizon line slope, the baseline error's 95th percentile exceeds 29°, whereas our method achieves a much lower limit of 3.27°, demonstrating significantly better performance.

Furthermore, to highlight one of the key advantages of using a neural network over classical computer vision methods, Table III compares the inference time of the proposed regression network with that of the baseline methods for estimating the desired parameters. It is evident that our network is significantly faster, performing a single inference in 9.56 ms on average. This efficiency makes it highly suitable for real-time applications, such as estimating the attitude of a camera installed on an aircraft operating at a high frame rate.

Finally, as a proof of concept for the proposed camera orientation estimation method, Figure 7 shows the errors in yaw, pitch and roll computed from the vanishing point and horizon line slope predicted by the regression network. Although several outliers are present (14% of the samples for roll, 18% for pitch, and 21% for yaw), it is evident that the mean errors are very close to zero, with most values falling below 5° for yaw and roll, and below 2° for pitch—indicating strong performance. The corresponding numerical results are presented in Table IV. Potential causes for the outliers include label noise due to how it was generated, complex backgrounds such as mountainous regions, or scenarios where the runway is so distant that the network struggles to accurately identify the vanishing point of its sidelines.

VII. Conclusion

This paper proposes a regression approach based on MobileNetV2 to estimate the vanishing point coordinates of a

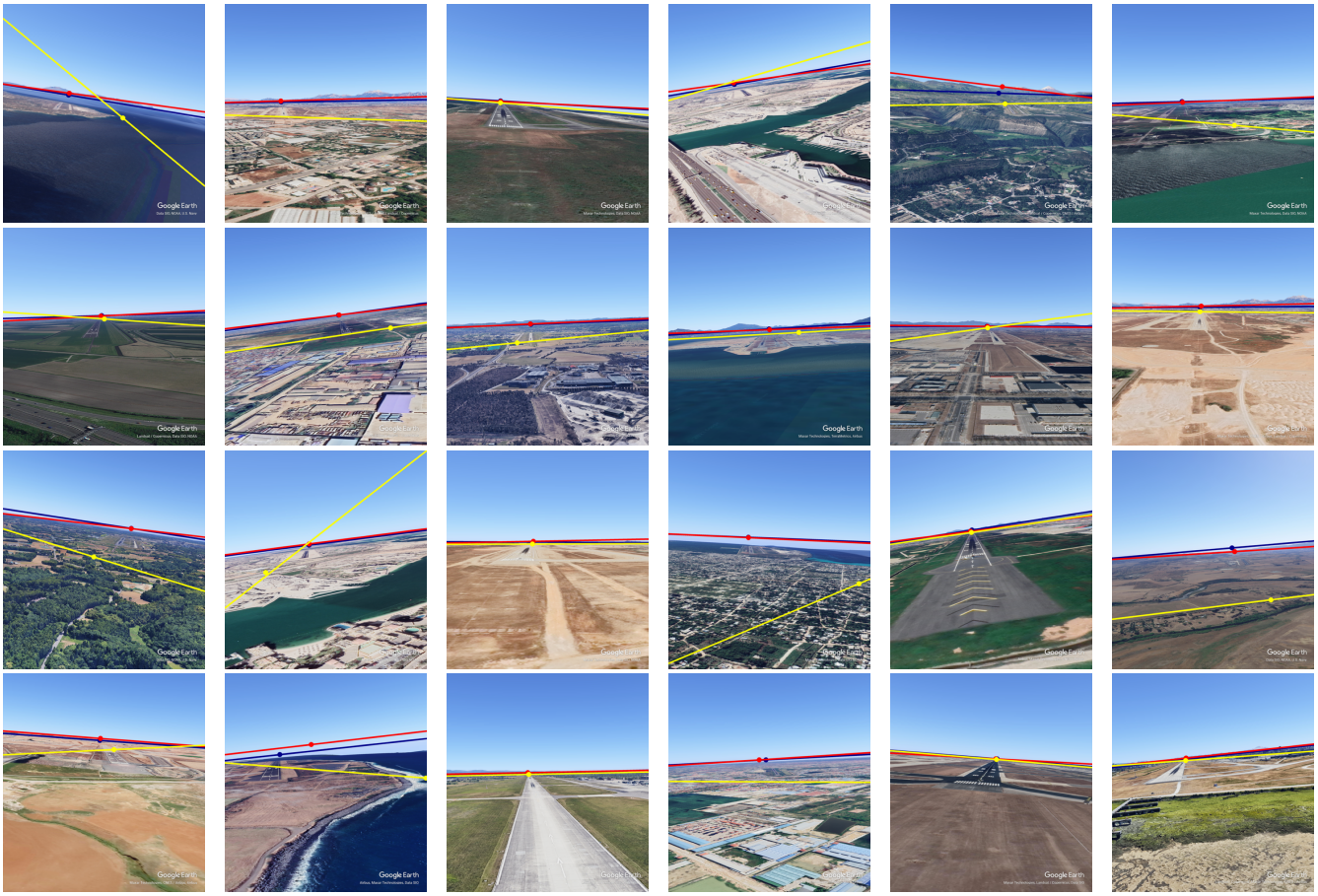

Fig. 5. Vanishing point and horizon line estimation results. In dark blue, the ground truth; in red, the predictions from our method; and in yellow, estimates from the baseline method. The dot represents the vanishing point and the line represents the horizon line.

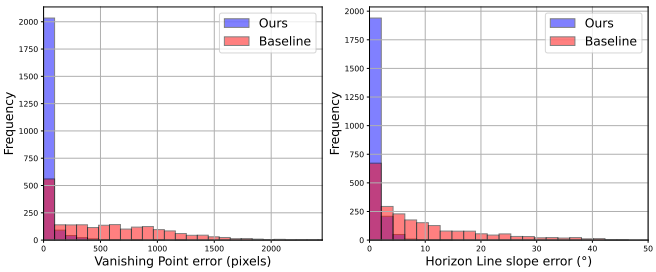

Fig. 6. Histograms of estimation errors for vanishing point locations (left) and horizon line slopes (right).

TABLE IV
PITCH AND ROLL ESTIMATION ERRORS.

Estimated parameter	Error (°)
Roll	0.02 ± 3.21
Pitch	0.00 ± 1.30
Yaw	1.25 ± 3.37

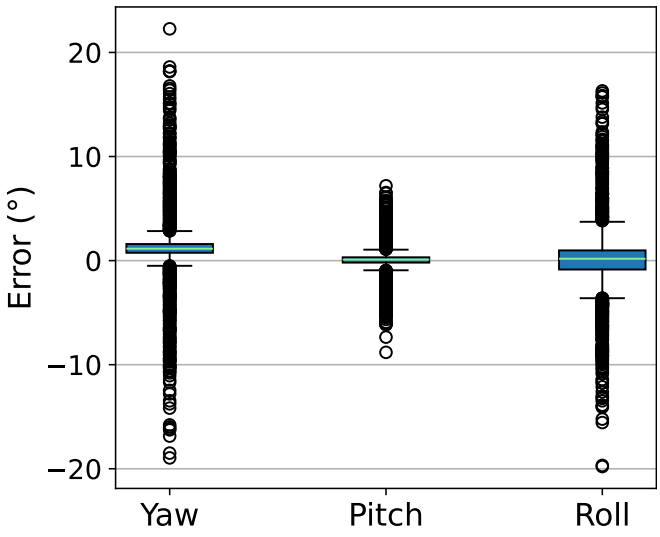

Fig. 7. Yaw, pitch and roll estimation errors.

landing runway and the slope of the horizon line. These estimated parameters are then used to infer the camera's—and consequently the airplane's— yaw, pitch and roll angles during the landing phase, leveraging vanishing point perspective geometry.

We achieve strong performance in vanishing point position and horizon line slope estimation on the LARD dataset using

the proposed model, significantly outperforming classical computer vision baseline methods in both accuracy—evidenced by lower errors—and inference speed. The accurate estimation of these parameters enables reliable inference of the camera's yaw, pitch and roll, with mostly low error rates. However, a notable number of outliers persist, likely due to challenging visual scenarios and label noise present within the dataset. Nonetheless, the results are promising for use as a cost-effective and redundant system to complement onboard aircraft orientation sensors, with potential applications in autonomous landing systems.

Future work could further enhance the results by exploring more robust feature extractors for the regression network. Additionally, multitask networks could be employed to improve performance by leveraging auxiliary tasks, such as runway instance segmentation, to assist in the regression of the vanishing point and horizon line slope. Finally, extending the experiments to datasets that include variations in daytime, weather, and lighting conditions would allow a more comprehensive assessment of the method's overall robustness.

ACKNOWLEDGMENT

This work is supported by FAPEMIG - Fundação de Amparo à Pesquisa do Estado de Minas Gerais and by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES).

REFERENCES

[1] Y.-M. Tsai, Y.-L. Chang, and L.-G. Chen, "Block-based vanishing line and vanishing point detection for 3d scene reconstruction," in *2006 International Symposium on Intelligent Signal Processing and Communications*, 2006, pp. 586–589.

[2] D. Chrysostomou, N. Kyriakoulis, and A. Gasteratos, "Multi-camera 3d scene reconstruction from vanishing points," in *2010 IEEE International Conference on Imaging Systems and Techniques*, 2010, pp. 343–348.

[3] J. Sarmento, A. S. Aguiar, F. N. d. Santos, and A. J. Sousa, "Robot navigation in vineyards based on the visual vanish point concept," in *2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA)*, 2021, pp. 406–413.

[4] Airbus, "A statistical analysis of commercial aviation accidents 1958-2023," Brochure, 2023.

[5] C.-K. Chang, J. Zhao, and L. Itti, "Deepvp: Deep learning for vanishing point detection on 1 million street view images," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4496–4503.

[6] H.-S. Choi, K. An, and M. Kang, "Regression with residual neural network for vanishing point detection," *Image and Vision Computing*, vol. 91, p. 103797, 2019.

[7] F. Kluger, H. Ackermann, M. Y. Yang, and B. Rosenhahn, "Temporally consistent horizon lines," 2020. [Online]. Available: https://arxiv.org/abs/1907.10014

[8] S. Workman, M. Zhai, and N. Jacobs, "Horizon lines in the wild," *arXiv preprint arXiv:1604.02129*, 2016.

[9] G. Balduzzi, M. Ferrari Bravo, A. Chernova, C. Cruceru, L. van Dijk, P. de Lange, J. Jerez, N. Koehler, M. Koerner, C. Perret-Gentil *et al.*, "Neural network based runway landing guidance for general aviation autoland," United States. Department of Transportation. Federal Aviation Administration . . . , Tech. Rep., 2021.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[11] Y. Shavit and R. Ferens, "Introduction to camera pose estimation with deep learning," *arXiv preprint arXiv:1907.05272*, 2019.

[12] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[13] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu, "Image-based localization using hourglass networks," in *Proceedings of the IEEE international conference on computer vision workshops*, 2017, pp. 879–886.

[14] N. Radwan, A. Valada, and W. Burgard, "Vlocnet++: Deep multitask learning for semantic visual localization and odometry," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4407–4414, 2018.

[15] Y. Lin, Z. Liu, J. Huang, C. Wang, G. Du, J. Bai, and S. Lian, "Deep global-relative networks for end-to-end 6-dof visual localization and odometry," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2019, pp. 454–467.

[16] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 716–12 725.

[17] J. a. P. K. Ferreira, J. a. P. Pinto, J. Moura, Y. Li, C. L. Castro, and P. Angelov, "Vision-based landing guidance through tracking and orientation estimation," in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, February 2025, pp. 9663–9671.

[18] H. Kong, J.-Y. Audibert, and J. Ponce, "General road detection from a single image," *IEEE Transactions on Image Processing*, vol. 19, no. 8, pp. 2211–2220, 2010.

[19] J. Lezama, R. Grompone von Gioi, G. Randall, and J.-M. Morel, "Finding vanishing points via point alignments in image primal and dual domains," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 509–515.

[20] C. Jeong, H. Yang, K.-D. Moon, and N. Pfeifer, "Fast horizon detection in maritime images using region-of-interest," *Sensors*, vol. 21, no. 13, 2021.

[21] S. Mikolka-Flöry and N. Pfeifer, "Horizon line detection in historical terrestrial images in mountainous terrain based on the region covariance," *Remote Sensing*, vol. 13, no. 9, p. 1705, 2021.

[22] M. FISCHLER AND, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[23] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[24] M. Ducoffe, M. Carrere, L. Féliers, A. Gauffriau, V. Mussot, C. Pagetti, and T. Sammour, "Lard–landing approach runway detection–dataset for vision based landing," *arXiv preprint arXiv:2304.09938*, 2023.

[25] M. Chen and Y. Hu, "An image-based runway detection method for fixed-wing aircraft based on deep neural network," *IET Image Processing*, vol. 18, no. 8, pp. 1939–1949, 2024.

[26] Q. Wang, W. Feng, H. Zhao, B. Liu, and S. Lyu, "Valnet: Vision-based autonomous landing with airport runway instance segmentation," *Remote Sensing*, vol. 16, no. 12, p. 2161, 2024.