drones

MDPI

Article

Research on Unmanned Aerial Vehicle (UAV) Visual Landing Guidance and Positioning Algorithms

Xiaoxiong Liu *, Wanhan Xue , Xinlong Xu, Minkun Zhao and Bin Qin

School of Automation, Northwestern Polytechnical University, Xi'an 710129, China; xuewanhan@mail.nwpu.edu.cn (W.X.); xuxinlong@mail.nwpu.edu.cn (X.X.); minkun_zhao@mail.nwpu.edu.cn (M.Z.); binq3638@mail.nwpu.edu.cn (B.Q.)
* Correspondence: nwpulxx@outlook.com

Abstract: Considering the weak resistance to interference and generalization ability of traditional UAV visual landing navigation algorithms, this paper proposes a deep-learning-based approach for airport runway line detection and fusion of visual information with IMU for localization. Firstly, a coarse positioning algorithm based on YOLOX is designed for airport runway localization. To meet the requirements of model accuracy and inference speed for the landing guidance system, regression loss functions, probability prediction loss functions, activation functions, and feature extraction networks are designed. Secondly, a deep-learning-based runway line detection algorithm including feature extraction, classification prediction and segmentation networks is designed. To create an effective detection network, we propose efficient loss function and network evaluation methods Finally, a visual/inertial navigation system is established based on constant deformation for visual localization. The relative positioning results are fused and optimized with Kalman filter algorithms. Simulation and flight experiments demonstrate that the proposed algorithm exhibits significant advantages in terms of localization accuracy, real-time performance, and generalization ability, and can provide accurate positioning information during UAV landing processes.

Keywords: computer vision; deep neural networks; autonomous landing; combined navigation

check for updates

Citation: Liu, X.; Xue, W.; Xu, X.; Zhao, M.; Qin, B. Research on Unmanned Aerial Vehicle (UAV) Visual Landing Guidance and Positioning Algorithms. *Drones* **2024**, *8*, 257. https://doi.org/10.3390/drones8060257

Academic Editors: Jihong Zhu, Heng Shi, Zheng Chen and Minchi Kuang

Received: 25 April 2024
Revised: 31 May 2024
Accepted: 7 June 2024
Published: 12 June 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Landing is a critical phase in unmanned aerial vehicle (UAV) flight. Currently, there are three main navigation methods used during UAV landings: instrument landing system (ILS), microwave landing system (MLS), and Global Positioning System (GPS). However, these navigation methods heavily rely on external equipment. Furthermore, they have drawbacks such as expensive equipment, poor maneuverability, difficulty in installation, susceptibility to signal interference, and vulnerability to deception. Therefore, the development of a fully autonomous, reliable, and stable autonomous landing navigation system has become an urgent problem.

With the continuous development of visual perception and navigation technologies, the application of visual navigation in the autonomous landing process of unmanned aerial vehicles (UAVs) has gained widespread attention. Visual navigation offers several advantages: (1) It does not require establishing an information link with the outside world, making it a completely autonomous navigation system that is immune to interference. (2) There is no need to set up expensive communications equipment on the ground, which is less costly. (3) It requires minimal prior information about the landing airport, allowing UAVs to land in relatively unfamiliar or temporary airfields.

UAV visual guidance landing image processing is particularly important, the current research methods on image processing can be divided into traditional methods to detect the runway line, traditional methods to detect cooperative signs and deep learning methods to detect the runway. Traditional methods are faster and easier to deploy but are more sensitive

to the environment; traditional methods to detect cooperative signs are more accurate but require certain labeling conditions; the deep learning methods used in this paper is environmentally robust and require fewer external conditions. These three methods have their own advantages and disadvantages and are used in different application scenarios

The detection of runway lines using traditional methods generally involves five steps: image preprocessing, feature extraction, feature selection, runway line fitting, and runway line detection and classification. In [1], the authors employ multi-sensor image fusion to obtain image data of the runway. They use support vector machine (SVM) for runway recognition and extract runway edges and ground lines using edge detection and the Hough transform. Finally, they obtain the aircraft's attitude data. In [2], the authors extract the horizon and runway edges using the Hough transform. They estimate the aircraft's pose separately using the horizon and runway edges and track the runway using template matching. In [3], the authors detect runway lines using the Canny edge detector and Hough transform.

The detection of cooperative markers using traditional methods is similar to the process of detecting runway lines. However, adjustments need to be made based on the characteristics and requirements of the cooperative markers. Generally, it involves five steps: image preprocessing, feature extraction, feature selection, marker detection and localization, and marker classification and recognition. In [4], a monocular single-frame vision measurement algorithm for autonomous landing of unmanned helicopters is derived. By detecting square cooperative targets, the position and attitude of the unmanned helicopter are calculated, totaling six parameters. In [5], the landing guidance process for unmanned aerial vehicles (UAVs) is divided into two stages. In the initial stage of landing, runway corner features are used as guidance markers, and in the later stage of landing, Apriltag labels are recognized to guide the UAV's landing. In [6], a fixed-wing UAV autonomous landing method based on binocular vision is proposed.

Using deep learning for runway line detection involves several steps, including data preparation and preprocessing, building the deep learning model, model training, test evaluation, model optimization, and model inference. Deep learning methods enable end-to-end training using a large amount of image data, allowing for the learning of more complex feature representations and improving the performance and robustness of runway line detection. Additionally, deep learning methods can adapt to different lighting conditions, complex backgrounds, and exhibit good generalization capability. In [7], a deep learning approach is used for UAV detection and tracking. By performing triangulation and filtering calculations on the detected objects in a binocular vision system, the spatial position of the UAV is estimated. In the positioning stage, a Kalman filter is used to smooth the spatial trajectory, approximating the area where the target is likely to appear in the current frame. This improves the accuracy of estimation while reducing the difficulty of tracking. In [8], an onboard-YOLO algorithm suitable for lightweight and efficient usage on UAV onboard systems is proposed. It utilizes separable convolutions instead of conventional convolutional kernels, effectively improving the detection speed.

However, single visual navigation alone may not satisfy the requirements for precision and reliability in autonomous UAV landing. It is necessary to complement the limitations of visual navigation by leveraging measurement information from other sensors. Traditional methods of combined navigation include GPS/INS(combined navigation), INS/visual combined navigation, GPS/INS/visual combined navigation, and so on. As the number of sensors increases, the accuracy and robustness of the navigation system continuously improve, enhancing the performance of autonomous UAV landing. In [9], a visual–inertial navigation fusion algorithm is proposed, where position and attitude alignment are achieved using Kalman filtering. The position alignment estimates velocity errors and accelerometer biases, while the attitude alignment estimates attitude errors and gyroscope drift. The estimated alignment errors and the attitude information output by the visual navigation system are used to correct the inertial navigation attitude. In [10], YOLOv3 is used to detect the runway region of interest (ROI), and an RDLines algorithm is employed to extract the left

and right runway lines from the ROI. A visual/inertial combined navigation model is then designed within the framework of square-root unscented Kalman filtering.

In a visual navigation system, accurate detection of the runway and runway lines is crucial for the system's performance [11]. Traditional methods for runway line detection, such as those based on the Hough transform and LSD line detection, offer good real-time performance. However, their generalization to different scenarios is poor and they heavily rely on manually designed features and parameters. Although researchers have proposed more comprehensive feature description methods (e.g., SIFT, ORB), their robustness and accuracy still cannot fully meet the demands of practical applications. Therefore, traditional line detection methods are not suitable for use in visual navigation systems. With the advancement of parallel computing capabilities brought about by hardware such as GPUs, deep neural networks have greatly improved detection accuracy and robustness. This has led to a new stage in object detection and the development of a series of deeper, faster training, and more accurate deep neural networks. Utilizing deep learning for runway line detection is a promising choice. Furthermore, considering the limitations of single visual navigation, integrating IMU (inertial measurement unit) information and visual localization results can be a good solution. This combination can provide complementary information and enhance the overall performance of the navigation system.

This paper proposes a deep-learning-based UAV localization method to address the navigation problem in autonomous UAV landing. The simulation and experimental results demonstrate that the proposed algorithm exhibits good robustness, accuracy, and real-time performance. These findings suggest that the algorithm can be used effectively for autonomous UAV landing.

The main contributions of this paper are as follows:

(1) A runway line detection and visual positioning system during visual guidance landing is constructed. The system is divided into four parts: runway ROI selection, runway line detection, visual positioning and combined navigation, thereby providing an end-to-end navigation solution for UAV visual guidance landing.

(2) In view of the requirements of navigation accuracy and real-time performance in this application scenario, the image processing end algorithm is optimized and designed, including optimizing the loss function, optimizing the feature extraction network and feature fusion network, adding an attention mechanism, and optimizing the network structure.

(3) In order to further improve the visual positioning accuracy, the Kalman filter algorithm is used to fuse the IMU information and the visual positioning information. The simulation results show that the combined navigation algorithm can effectively improve the positioning accuracy.

The rest of this paper is organized as follows: The Section 2 of the paper presents the framework of the visual-guided landing localization algorithm. The Section 3 focuses on the runway ROI selection algorithm in the visual landing detection, while the Section 4 explores the runway line detection algorithm for airport runways within the visual landing detection. The Section 5 discusses the visual localization and combined navigation algorithm. Finally, the Section 6 describes the deployment of the algorithm on edge computing devices and presents the results of simulation experiments.

2. Vision-Guided Landing Positioning Algorithm Framework

This paper focuses on visual-guided landing for small fixed-wing unmanned aerial vehicles (UAVs). The main research objectives are image processing algorithms and visual localization algorithms for UAV landing. The specific research content includes runway ROI selection networks, runway line detection networks, UAV position estimation algorithms, and combined navigation algorithms. The effectiveness of these algorithms is then validated in a constructed simulation system. The overall algorithm framework is illustrated in Figure 1.

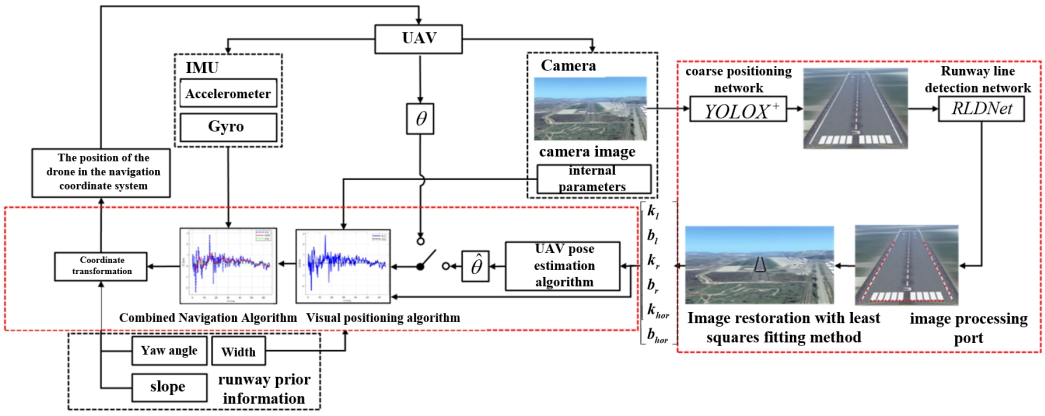


Figure 1. Overall flowchart for visually guided landing.

The algorithm system designed in this paper consists of two main components: the image processing side and the pose estimation combined navigation side. In the image processing side, the images captured by the camera are processed. The YOLOX runway ROI selection network is used to perform rough positioning of the runway lines, which helps exclude interfering objects and ensures that the runway is evenly distributed in the image. The detected bounding boxes are then input into the runway line detection network (RLDNet) for line detection. In this stage, instead of using image segmentation techniques, a specific row (or column) classification method is employed, reducing computational complexity and improving real-time inference. The line detection outputs information such as the slope and intercept of the runway lines. The pose estimation and combined navigation side mainly consist of two algorithms: visual localization and combined navigation. The visual localization algorithm utilizes prior information about the runway, camera intrinsic parameters, UAV attitude information, and information obtained from the detected runway lines to estimate the UAV's position. The visual localization algorithm in this study utilizes a vision-based localization algorithm based on the concept of homogeneous transformation. The position information derived from visual localization is fused with the position information obtained from the IMU using Kalman filtering, resulting in accurate and reliable localization results.

The algorithm designed in this study is applicable to UAV visual landing guidance scenarios. However, most currently available datasets that include airport runways are composed of aerial images, such as the runway images in NWPU-RESISC45, which are all remote sensing aerial images. These datasets cannot meet the design requirements of the algorithm proposed in this study. Therefore, this study combines runway images from the landing perspective in the virtual simulation system Vega Prime, real airport runway images, affine-transformed NWPU-RESISC45 runway images, and simplified runway images. After manual processing and annotation, these images form the dataset used in this study, which includes a total of 2500 landing perspective runway images. Additionally, for the purpose of simulating and validating the visual positioning algorithm and combined navigation algorithm, the dataset collected from the virtual simulation system Vega Prime includes UAV's real poses and IMU data corresponding to the images. The four types of airport runway images included in this dataset are shown in Figure 2.

Furthermore, in order to save annotation time, this study initially annotates the left and right runway lines, as well as the starting runway line. Then, by calculating the position of the rough runway localization box based on the annotated pixel coordinates of the runway line endpoints, the airport runway rough localization dataset is automatically generated. After cropping the original image using the rough localization box, the resolution is adjusted to generate the airport runway line dataset. This allows for the simultaneous generation of datasets for both the airport runway rough localization and the airport runway line detection tasks through a single annotation process.

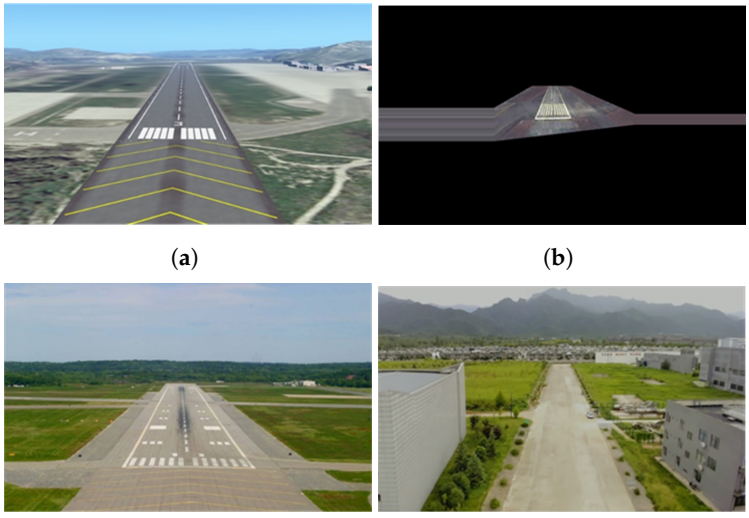


Figure 2. Construction of the visual landing guidance datasets. (**a**) Runway in Vega Prime; (**b**) Remote sensing image; (**c**) Real runway; (**d**) Airstrip runway.

In order to effectively utilize the dataset and validate the algorithm's performance, this study divided the constructed runway rough localization dataset and runway line detection dataset into training and testing sets using a ratio of 4:1. The training set is utilized to train the runway rough localization network and runway line detection network. The testing set is then used to evaluate the prediction performance of the rough localization network and runway line detection network, as well as to calculate their performance metrics. This division helps with the optimization and adjustment of the networks.

3. Airport Runway Rough Localization Algorithm

As a general-purpose object detection framework, YOLOX has high detection accuracy and speed for most object detection applications. However, due to the higher requirements for image processing accuracy and real-time performance in the scenario of this study, further optimization and design of YOLOX are needed.

3.1. Design of Probability Prediction Loss Function

The probability prediction loss of YOLOX, L_{obj}, is calculated using the binary cross-entropy loss [12]. For a given sample, the binary cross-entropy loss is computed as Equation (1).

$$l(y_i, \hat{y}_i) = \sum_{i=0}^{C} -(y_i \log(\hat{y}_i) + y_i \log(1 - \hat{y}_i)) \tag{1}$$

where y_i represents the ground truth and $\hat{y}_i$ represents the predicted value. For all samples, the binary cross-entropy loss function value is the average of the loss function values for all positive and negative samples. The calculation method is as Equation (2).

$$BCELoss = \frac{1}{N} \sum_{i=1}^{N} l(y_i, \hat{y}_i) \tag{2}$$

where N represents the total number of positive and negative samples.

Focal Loss is a solution for addressing the issue of sample imbalance [13]. Its calculation method is as Equation (3).

$$Focal\ Loss = \begin{cases} -(1-\hat{p})^{\gamma} \log(\hat{p}) & if\ y_i = 1 \\ -\hat{p}^{\gamma} \log(1-\hat{p}) & if\ y_i = 0 \end{cases} \tag{3}$$

Let

$$p_t = \begin{cases} \hat{p} & if\ y_i = 1 \\ 1 - \hat{p} & else \end{cases} \tag{4}$$

The expression for Focal Loss can be uniformly represented as Equation (5).

$$Focal\ Loss = -(1 - p_t)^{\gamma} \log(p_t) \tag{5}$$

where p_t reflects the degree of proximity between the predicted value and the ground truth. The larger the value of p_t, the closer the predicted value is to the ground truth, indicating a more accurate classification. Where $\gamma > 0$ is an adjustable factor. Similarly, the expression for the binary cross-entropy loss function can be uniformly represented as Equation (6).

$$L_{ce} = -\log(p_t) \tag{6}$$

Compared to the binary cross-entropy loss function, Focal Loss does not modify the loss function value for inaccurately classified samples, while reducing the weight of the loss function value for accurately classified samples. This ultimately increases the weight of inaccurately classified samples in the overall loss function.

The calculation method of the Focal Loss loss function used in the training process is as Equation (7).

$$Loss = -\alpha(1 - p_t)^{\gamma} \log(p_t) \tag{7}$$

That is, in the traditional Focal Loss, a coefficient is introduced, and $\alpha = 0.25$, $\gamma = 2$. At this point, the model accuracy will slightly improve. In all subsequent experiments in this study, the binary cross-entropy loss is replaced with Focal Loss by default.

3.2. Design of Regression Loss Function

YOLOX calculates the position regression loss for predicting bounding boxes and ground truth boxes using the IoU loss. When the predicted box and the ground truth box do not intersect, the IoU loss function cannot reflect the distance between the predicted box and the ground truth box. In this case, the loss function is non-differentiable, making it unable to optimize the scenario where the two boxes do not intersect. Therefore, this paper replaces the calculation method of the position regression loss with EIoU. The EIoU loss, which reduces the contribution of a large number of anchor frames with less overlap area with the target frame to the predictor frame regression, makes the regression of the predictor frame more focused on high-quality anchor frames. The EIoU is calculated as Equation (8).

$$L_{EIoU} = L_{IoU} + L_{dis} + L_{asp} = 1 - IoU + \frac{\rho^2(c, c^{gt})}{d^2} + \frac{\rho^2(w, w^{gt})}{C_w^2} + \frac{\rho^2(h, h^{gt})}{C_h^2} \tag{8}$$

where $\frac{\rho^2(c,c^{gt})}{d^2}$ denotes the centroid loss, $\frac{\rho^2(w,w^{gt})}{C_w^2}$ is the width loss, $\frac{\rho^2(h,h^{gt})}{C_h^2}$ is the height loss, and C_w and C_h are the widths and heights of the smallest outer bounding box containing the prediction box and the target box.

3.3. Design of Feature Extraction Network

The feature extraction network in YOLOX is a multi-branch residual structure called CSPDarknet53. Since the algorithm in this paper needs to be deployed on edge devices, in order to further compress the model parameter size while improving model accuracy, the feature extraction network of YOLOX is replaced with EfficientRe [14], GhostNet [15], MobileNetV3-Large, and MobileNetV3-Smallin [16] in separate experiments. The performance of different feature extraction networks is tested, and the experimental results are shown in Table 1.

Table 1. The ablation experiments on different feature extraction networks.

Feature Extraction Network	Evaluation Metrics					
	$AP_{0.75}$	F1	*Recall* (%)	Precision	FLOPS (G)	Param (M)
EfficientRep	93.42	0.86	90.08	82.09	50.229	17.05
GhostNet	94.95	0.86	90.6	82.22	20.309	9.06
MobileNetV3-Small	93.82	0.85	89.11	81.98	**18.473**	**6.849**
MobileNetV3-Large	**95.01**	**0.87**	**91.37**	**82.35**	31.703	25.974

From the experimental results, it can be seen that for the dataset used in this paper, introducing EfficientRep does not significantly improve the model's performance. On the contrary, the introduction of EfficientRep leads to a significant increase in model size and computational complexity. When using MobileNetV3-Large as the feature extraction network, the model's performance is significantly improved; however, the trade-off is a substantial increase in both model size and computational complexity. As a comparison, this paper uses MobileNetV3-Small as the feature extraction network, which significantly reduces the model size and computational complexity. Although $AP_{0.75}$ is slightly higher than EfficientRep, it is much lower than MobileNetV3-Large. When using GhostNet as the feature extraction network, there is a significant improvement in $AP_{0.75}$, recall, and precision. In terms of parameter size, the model is comparable to using CSPDarknet53 as the feature extraction network, but the computational complexity is substantially reduced. Given the requirements of the application scenario in this paper regarding model performance and computational complexity, from this point forward, the experiments in this paper default to using GhostNet as the feature extraction network.

In addition, the feature extraction network in YOLOX includes the SiLU activation function. As the network deepens, models that use the SiLU activation function tend to experience a noticeable decrease in classification accuracy. In this paper, on the basis of the YOLOX network structure, the SiLU activation function is replaced with the Mish activation function. With the deepening of the network, the Mish activation function can still maintain a higher classification accuracy. Equation (9) is the expression of the Mish activation function [17].

$$Mish = x \times \tanh(\ln(1 + e^x)) \tag{9}$$

3.4. Feature Fusion Networks and Channel Attention Mechanisms

In this paper, the ordinary convolutions in the YOLOX feature fusion network are replaced with group shuffle convolution (GSConv). GSConv reduces the model's parameter count while preserving the connections between channels in the feature layers, ensuring that the model's accuracy is not compromised [18,19]. After the feature layers go through ordinary convolutions, GSConv applies depth-wise separable convolutions, and then, concatenates the feature layers before the depth-wise separable convolutions in the channel direction. Finally, the shuffle structure is used to fuse the feature layers from both ordinary convolutions and depth-wise separable convolutions. Additionally, if GSConv is used throughout the entire model, the model will become deeper and may have an impact on real-time performance. Therefore, in this paper, only the ordinary convolutions in the YOLOX feature fusion network are replaced with GSConv, specifically, replacing the BottleNeck in CSPLayer with GSBottleNeck.

Attention mechanism can allocate computational resources to more important tasks without increasing computational complexity significantly, especially when resources are limited.

The efficient channel attention (ECA) mechanism builds upon the SE channel attention mechanism [20] by replacing the fully connected layer with a (1×1) convolutional layer. This allows for learning the weight information between channels without reducing the channel dimension, and it also helps reduce the number of parameters [21]. The ECA mech-

anism first applies global average pooling to the input feature layer to obtain $(1 \times 1 \times C)$-dimensional feature maps. Then, through $(1 \times 1 \times k)$ convolutional operations, it learns the importance of different channels.

The size of the convolutional kernel affects the receptive field, and larger convolutional kernels are needed for feature layers with a larger number of channels. Therefore, the kernel size can be dynamically adjusted using a function. The calculation method for the convolutional kernel is as Equation (10).

$$k = \psi(C) = \left| \frac{\log_2(C)}{2} + \frac{1}{2} \right|_{odd} \tag{10}$$

In this context, k represents the number of channels in the convolutional kernel, C represents the number of channels in the input convolutional layer, and $||_{odd}$ indicates that the size of the convolutional kernel must be an odd number.

In this paper, we added the channel attention mechanism ECA in the middle of YOLOX's feature extraction network and PAFPN.

4. Airport Runway Line Detection Algorithm

4.1. Detection Principle

In this paper, the idea behind designing the runway line detection algorithm is to select the correct location of the left and right runway lines in a predefined row anchor box and the start location of the runway line in a predefined column anchor box using global features. Therefore, the first step is to partition the input image into row anchor boxes and column anchor boxes. Then, each row and column anchor box is further divided into grid cells. In this way, runway line detection can be defined as selecting specific cells within the predefined row/column anchor boxes to represent the positions of the left and right runway lines and the starting runway line.

Assume the maximum number of runways is C, the number of row anchor boxes is h, and the number of grid cells in each row/column anchor box is w, and let X denote the global features of the image. Let f^{ij} represent the classifier for the runway line positions on the ith row/column anchor box of the jth runway. Then, the prediction of the runway line can be expressed as Equation (11).

$$P_{i,j,:} = f^{ij}(X) \tag{11}$$

where $i \in [1, C]$, $j \in [1, h]$, $P_{i,j,:}$ is an S$-$dimensional vector that represents the probability of the Nth grid of the Mth runway line; F denotes the global features of the image, and it is a $(w+1)-$dimensional vector. It represents the probability of the $(w+1)$th grid cells for the ith runway line. For each grid in every row/column anchor box, the network predicts the probability of the corresponding grid. Thus, the grid with the highest probability represents the predicted position of the runway line. If no runway line is predicted on a particular row/column anchor box, then the probability of the last grid in that anchor box is set to 1.

4.2. Network Structure

The network architecture consists of three parts: feature extraction, classification prediction, and segmentation. The feature extraction part is responsible for extracting the features of the runway lines from the image. The classification prediction part is used to classify these features, while the segmentation part helps to fuse multi-scale features, improving the detection accuracy. To improve the network's inference speed, the segmentation part is only used during training and not utilized during the inference prediction stage [22]. The network structure is illustrated in Figure 3.

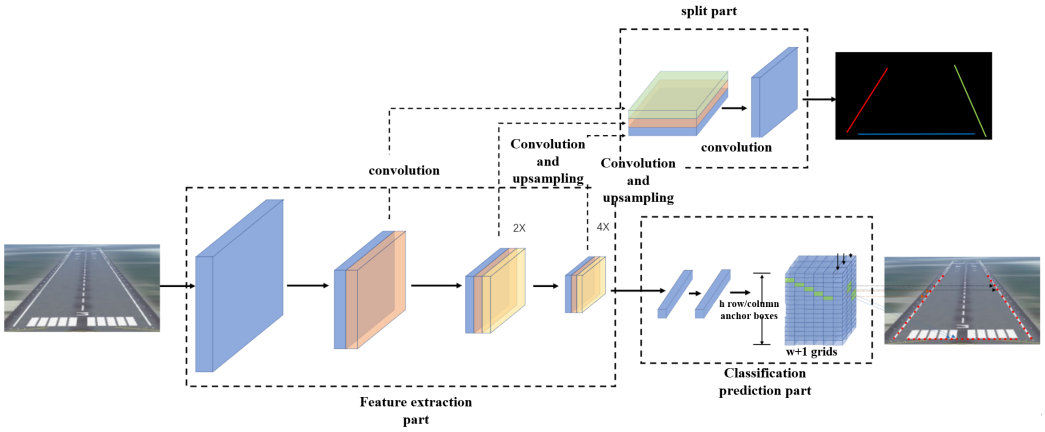


Figure 3. Diagram of runway line detection network structure.

The role of the feature extraction part is to extract the features of the runway lines and provide them to the classification prediction part. Common feature extraction networks, such as ResNet, VGG, MobileNet, ShuffleNet, have been proven to exhibit strong feature extraction capabilities for classification tasks. In this algorithm, ResNet is used as the feature extraction network. ResNet is a type of residual network that addresses the problem of increased loss with increasing network depth [23]. Considering the need for extracting a relatively limited set of features and the requirement for real-time processing on board computers, the algorithm utilizes the lightest variant of ResNet, which is ResNet18 (18 represents the number of layers that require parameter updating through training).

In the classification prediction, the last feature layer of ResNet18 is initially downsampled by the convolutional operation, reducing the number of channels. Then, the resulting feature layer is flattened into a column, resulting in a dimension of $(1 \times 1 \times 1800)$. Next, the feature layer dimension is transformed to $(1 \times 1 \times 13,635)$ using a fully connected layer and the ReLU activation function. Finally, the dimension of the fully connected layer is reshaped to $((w+1) \times h \times 3)$ using the reshape operation. In this context, $w+1$ represents the number of grid cells for each row/column anchor box, h represents the number of row/column anchor boxes, and 3 corresponds to the total number of runway lines. And the condition in Equation (12) needs to be satisfied.

$$h \times (w+1) \times 3 = 13635 \tag{12}$$

Performing softmax on each row/column anchor box for the three runway lines can compute the grid with the highest probability within each anchor box. This is used as the predicted position of the track line and is utilized for calculating classification loss, structural loss, and association loss.

In the segmentation network, the last three feature layers of ResNet18 are first subjected to convolution and upsampling operations. These three feature layers are then concatenated along the channel dimension. Subsequently, convolution is applied to reduce the number of channels in the feature layer to four, these are used for calculating the segmentation loss.

4.3. Loss Function

The classification loss during the network training process can be represented as Equation (13).

$$L_{cls} = \sum_{i=1}^{C} \sum_{j=1}^{h} L_{CE}(P_{i,j,:}, T_{i,j,:}) \tag{13}$$

Here, L_{CE} represents the cross-entropy loss, and $T_{i,j,:}$ represents the ground truth of the track line position on the jth row/column anchor box of the ith track line.

In addition to the classification loss, several other loss functions are used in the algorithm based on the structural prior information of the runway lines. These loss functions are utilized to represent the position relationships of the runway lines, allowing the neural network to learn the structural information of the runway lines. Since each track line must be continuous, the predicted points of the runway lines in adjacent row/column anchor boxes should be as close as possible. Therefore, the continuity of the predicted runway lines can be achieved by constraining the distribution of the classification vectors on adjacent row anchors. The loss function can be represented as Equation (14).

$$L_{sim} = \sum_{i=1}^{C} \sum_{j=1}^{h-1} \| P_{i,j,:} - P_{i,j+1,:} \|_1 \tag{14}$$

Here, $P_{i,j,:}$ represents the predicted track line position on the jth row/column anchor box of the ith track line, $P_{i,j+1,:}$ represents the ground truth of the track line position on the $(j+1)$th row/column anchor box of the ith track line. In the loss function, the distance between the predicted track line positions and the ground truth is minimized through an L_1-norm constraint.

Additionally, based on the prior information that each track line is a straight line, the predicted track points can be constrained using second-order differences. The formula for the second-order difference can be represented as Equation (15).

$$L_{shp} = \sum_{i=1}^{C} \sum_{j=1}^{h-2} \| ((Loc_{i,j} - Loc_{i,j+1}) - (Loc_{i,j+1} - Loc_{i,j+2}) \|_2 \tag{15}$$

Here, $Loc_{i,j}$ represents the predicted point on the jth row/column anchor box of the jth track line, and its calculation method is given as Equation (16).

$$Loc_{i,j} = \sum_{k=1}^{w} k \cdot Prob_{i,j,k} \tag{16}$$

Here, $Prob_{i,j,k}$ represents the probability of the ith track line in the kth grid of the jth row/column anchor box, and its calculation method is given as Equation (17).

$$Loc_{i,j,:} = softmax(P_{i,j,1:w}) \tag{17}$$

Based on the above, the overall structural loss of the network can be represented as Equation (18).

$$L_{str} = L_{sim} + \lambda L_{shp} \tag{18}$$

In addition to the classification loss and structural loss, this paper incorporates an auxiliary segmentation task that utilizes multi-scale features for local feature modeling during the training process. The auxiliary segmentation loss is calculated using the cross-entropy function. To improve the performance of the algorithm, this segmentation task is removed during the testing phase.

Real runway lines are parallel to each other, but due to perspective, the left and right runway lines in the image become closer as they move upward. Based on this prior condition, this paper designs the association loss for the runway lines. The design logic is as follows: if the left and right runway lines above the image are farther apart compared to the left and right runway lines below, a loss is generated; otherwise, no loss is generated. The calculation process of the association loss is shown in Algorithm 1, where ε_T represents the tolerable error threshold, which is defined in terms of vertical grids.

Algorithm 1 Correlation Loss Calculation Process

Input: The predicted track line point on the $j-th$ row anchor box of the $i-th$ track line: $Loc_{i,j}, i \in [0,1], j \in [0,h-1]$.

Output: The value of the association loss :L_{reduce}

1: **for** $j = 0$ to $h-1$ **do**
2: $D_j = Loc_{1,j} - Loc_{0,j}$
3: $\Delta D_j = D_j - D_{j-1}, j \in [1,h-1]$
4: $M_j = 0.5 \times (|\Delta D_j| - \Delta D_j) - \varepsilon_T, j \in [1,h-1]$
5: $N_j = 0.5 \times (|M_j| + M_j), j \in [1,h-1]$
6: $L_{reduce} = \sum_{j=1}^{h-1} \|N_j\|_1, j \in [1,h-1]$;
7: **end for**

In summary, the overall loss of the algorithm can be represented as Equation (19).

$$L_{total} = \alpha L_{cls} + \beta L_{str} + \gamma L_{seg} + \theta L_{reduce} \tag{19}$$

Here, L_{cls} represents the classification loss, L_{str} represents the structural loss, L_{seg} represents the segmentation loss, and L_{reduce} represents the association loss. α, β, γ, and θ represent the weights assigned to the classification loss, structural loss, segmentation loss, and association loss, respectively.

4.4. Evaluation

In order to ensure the stability of distance calculation and accurately reflect the differences between the ground truth and predicted points, this paper first uses the predicted points on the runway lines for least squares fitting to obtain the slope of the fitted line (k). This further allows us to calculate the distance threshold (ε) between the predicted and ground truth points. The calculation method is as Equation (20).

$$\varepsilon = \frac{RealDistance}{cos(arctan(k))} \tag{20}$$

where RealDistance represents the pixel distance between the predicted points and the ground truth points in the horizontal or vertical direction. Considering the actual angle between the left/right runway lines and the x-axis of the pixel coordinate system is close to 90 degrees, and the angle with the y-axis is smaller; the starting track line is close to 90 degrees with the y-axis, and the angle with the x-axis is smaller. Therefore, when calculating the slope (k) of the line, the left/right runway lines adopt the line equation '$x = ky + b$', and the starting track line adopts the line equation '$y = kx + b$'. This means that the angle with the y-axis of the pixel coordinate system is used when calculating the threshold for the left/right runway lines, and the angle with the x-axis is used when calculating the threshold for the starting track line.

Since the runway lines predicted by the neural network are obtained by fitting the grid points on the predicted track line using the least squares method, the least squares method reduces the impact of prediction errors on track line predictions to a certain extent. Therefore, to evaluate the accuracy of the track line predictions, it is necessary to quantitatively calculate the similarity between the predicted and ground truth runway lines. The evaluation metrics include accuracy, miss rate, and over-detention rate. Accuracy represents the similarity in slope between the predicted and ground truth runway lines, and its calculation method is as Equation (21).

$$acc = \frac{\sum_{j=1}^{w} \sum_{i=1}^{h} \Lambda_{ij}}{w \times h} \tag{21}$$

$$\Lambda_{i,j} = \begin{cases} 1 & |\arctan(k_{real,i,j}) - \arctan(k_{pred,i,j})| < \varepsilon \\ 0 & else \end{cases} \tag{22}$$

Additionally, the miss rate represents the proportion of the dataset that has ground truth runway lines but no corresponding predicted results. The over-detection rate represents the proportion of the dataset that has predicted runway lines but no corresponding ground truth.

5. Algorithms for Visual Positioning and Combined Navigation

5.1. Algorithm for Visual Positioning

The coordinate systems involved in performing position solving in this paper include the navigation coordinate system, runway coordinate system, airframe coordinate system, camera coordinate system, phase plane coordinate system, and pixel coordinate system, etc., which are defined as Figure 4:

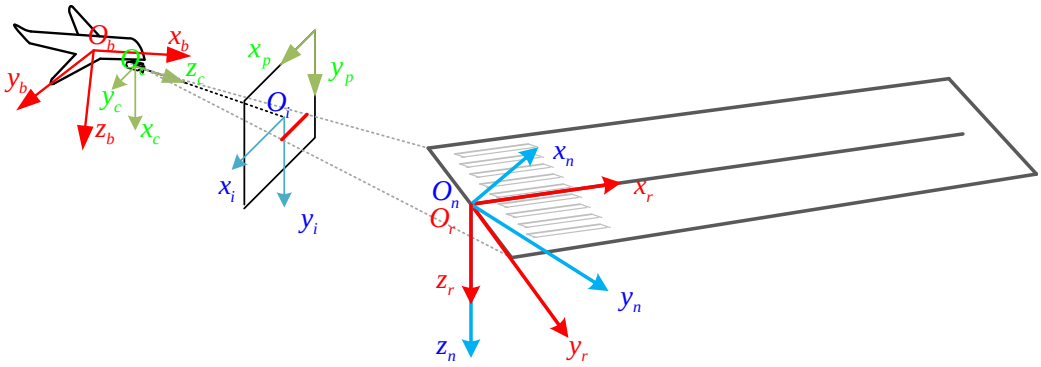


Figure 4. The coordinate systems involved in position algorithm.

The navigation coordinate system ($O_n x_n y_n z_n$) is defined as the northeast ground coordinate system. The runway coordinate system ($O_r x_r y_r z_r$) is the base coordinate system for the position solving in the visual guidance landing process of the UAV in this paper. The fuselage coordinate system ($O_b x_b y_b z_b$) has the origin O_b at the center of the UAV IMU. The camera coordinate system ($O_c x_c y_c z_c$) is solidly connected to the camera. The image plane coordinate system ($O_i x_i y_i z_i$) has the image plane located in front of the camera at z = f (f is the focal length). The pixel coordinate system ($O_p x_p y_p z_p$) is used to describe the original image information.

The conversion relationship from runway coordinate system to pixel coordinate system is as Equation (23) [24].

$$\begin{aligned} \vec{v}_p &= \frac{1}{Z_c} K\vec{v}_c = \frac{1}{Z_c} K\mathbf{C}_\gamma(\vec{v}_b - \vec{v}_1) \\ &= \frac{1}{Z_c} K\mathbf{C}_\gamma\left[\mathbf{C}_n^b(\vec{v}_n - \vec{v}_0) - \vec{v}_1\right] \\ &= \frac{1}{Z_c} K\mathbf{C}_\gamma\left[\mathbf{C}_n^b(\mathbf{C}_r^n\vec{v}_r - \vec{v}_0) - \vec{v}_1\right] \\ &= \frac{1}{Z_c} K\mathbf{C}_\gamma\mathbf{C}_n^b C_r^n\left[\mathrm{I}_{3\times3} \mid -\left(\mathbf{C}_n^b\mathbf{C}_r^n\right)^{-1}\left(\mathbf{C}_n^b\vec{v}_0 + \vec{v}_1\right)\right]\begin{bmatrix}\vec{v}_r \\ 1\end{bmatrix} \\ &= \frac{1}{Z_c} K\mathbf{R}_{cr}\left[\mathbf{I}_{3\times3} \mid -\vec{t}_{cr}\right]\begin{bmatrix}\vec{v}_r \\ 1\end{bmatrix} = \frac{1}{Z_c}\mathbf{P}\begin{bmatrix}\vec{v}_r \\ 1\end{bmatrix} \end{aligned} \tag{23}$$

where $P = KR_{cr}[I_{3\times3}| - \vec{t}_{cr}]$ represents the projection matrix from the runway coordinate system to the pixel coordinate system; $R_{cr} = C_\gamma C_n^b C_r^n = C_\gamma C_r^b$ represents the rotation matrix from the runway coordinate system to the camera coordinate system; and

$\vec{t}_{cr} = (C_n^b C_r^n)^{-1}(C_n^b \vec{v}_0)$ represents the coordinates of the camera coordinate system origin in the runway coordinate system. $\vec{v}_r$ indicates the coordinates of a point in the runway coordinate system; $\overrightarrow{v_p}$ denotes the coordinates of the corresponding point in the pixel coordinate system; $\overrightarrow{v_{0r}}$ denotes the coordinates of the origin of the airframe coordinate system in the runway coordinate system; $\overrightarrow{v_0}$ denotes the coordinates of the origin of the fuselage coordinate system in the navigation coordinate system; $\overrightarrow{v_1}$ denotes the coordinates of the camera optical center in the body coordinate system; K is an internal parameter of the camera that can be obtained by calibrating; Z_c is the z-axis coordinate of the point in the camera coordinate system; γ is the camera mounting angle; $\mathbf{C}_\gamma$ is the rotation matrix of the camera mounting angles; $\mathbf{C}_r^b$ represents the rotation matrix from the runway system to the machine system; and $\mathbf{C}_r^n$ represents the rotation matrix from the runway coordinate system to the navigation coordinate system. Since the role of the runway coordinate system relative to the world coordinate system is very small and can be neglected, $\mathbf{C}_r^n$ can be represented as Equation (24).

$$\mathbf{C}_r^n = (\mathbf{C}_n^r)^T = \left(\mathbf{C}_{\phi_r}\mathbf{C}_{\theta_r}\mathbf{C}_{\psi_r}\right)^T \approx \left(\mathbf{C}_{\theta_r}\mathbf{C}_{\psi_r}\right)^T \tag{24}$$

If the equation of a line in the pixel coordinate system is $y = kx + b$, then any point, $\vec{v}_p = [x_p \quad y_p \quad 1]^T$ on that line must satisfy Equation (25).

$$[k \quad -1 \quad b]\vec{v}_p = [k \quad -1 \quad b]\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = 0 \tag{25}$$

By multiplying both sides of Equation (23) by $[k \quad -1 \quad b]$, we can obtain

$$[k \quad -1 \quad b]K\mathbf{R}_{cr}[\mathbf{I}_{3\times 3}| - \vec{t}_{cr}]\begin{bmatrix} \vec{v}_r \\ 1 \end{bmatrix} = 0 \tag{26}$$

So $\vec{A} = [a_1 \quad a_2 \quad a_3]KR_{cr} = [a_1 \quad a_2 \quad a_3]$, and

$$\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & -x_{cr} \\ 0 & 1 & 0 & -y_{cr} \\ 0 & 0 & 1 & -z_{cr} \end{bmatrix}\begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix} \tag{27}$$

Expanding the above equation, we can obtain

$$a_1 x_r + a_2 y_r + a_3 z_r = a_1 x_{cr} + a_2 y_{cr} + a_3 z_{cr} \tag{28}$$

Assuming the width of the runway is denoted as W_r, then in the runway coordinate system, the coordinates of any point on the left runway line can be represented as Equation (29):

$$\begin{bmatrix} x_r & -\frac{W_r}{2} & 0 \end{bmatrix}^T \tag{29}$$

where x_r is an arbitrary variable. By substituting Equation (29) into Equation (28), we can obtain

$$a_1 x_r - a_2 \frac{W_r}{2} = a_1 x_{cr} + a_2 y_{cr} + a_3 z_{cr} \tag{30}$$

Since Equation (30) holds true for any variation in x_r, we can conclude that $a_1 = 0$, which leads to

$$-\frac{W_r}{2} = y_{cr} + \frac{a_3}{a_2} z_{cr} \tag{31}$$

so

$$\begin{bmatrix} 1 & \frac{a_3^l}{a_2^l} \end{bmatrix} \begin{bmatrix} y_{cr} \\ z_{cr} \end{bmatrix} = -\frac{W_r}{2} \tag{32}$$

Similarly, for the right runway, we have

$$\begin{bmatrix} 1 & \frac{a_3^r}{a_2^r} \end{bmatrix} \begin{bmatrix} y_{cr} \\ z_{cr} \end{bmatrix} = \frac{W_r}{2} \tag{33}$$

By combining Equation (32) and Equation (33), we have

$$\begin{bmatrix} 1 & \frac{a_3^l}{a_2^l} \\ 1 & \frac{a_3^r}{a_2^r} \end{bmatrix} \begin{bmatrix} y_{cr} \\ z_{cr} \end{bmatrix} = \begin{bmatrix} -\frac{W_r}{2} \\ \frac{W_r}{2} \end{bmatrix} \tag{34}$$

For the starting runway line, it must satisfy the following equation in the runway coordinate system:

$$\vec{v}_r = \begin{bmatrix} 0 & x_r & 0 \end{bmatrix}^T \tag{35}$$

where x_r is an arbitrary variable. By substituting Equation (35) into Equation (28), we can obtain

$$a_2^s x_r = a_1^s x_{cr} + a_2^s y_{cr} + a_3^s z_{cr} \tag{36}$$

To ensure that Equation (36) holds true for any variation in x_r, we have $a_2^s = 0$. Therefore, we can conclude that

$$x_{cr} = -\frac{a_3^s}{a_2^s} z_{cr} \tag{37}$$

For the variables a_2 and a_3 in Equation (33) and Equation (37), they can be calculated from $\vec{A}$, as Equation (38).

$$\begin{cases} \vec{A}_l = \begin{bmatrix} k_l & -1 & b_l \end{bmatrix} K \mathbf{R}_{cr} = \begin{bmatrix} a_1^l & a_2^l & a_3^l \end{bmatrix} \\ \vec{A}_r = \begin{bmatrix} k_r & -1 & b_r \end{bmatrix} K \mathbf{R}_{cr} = \begin{bmatrix} a_1^r & a_2^r & a_3^r \end{bmatrix} \\ \vec{A}_s = \begin{bmatrix} k_s & -1 & b_s \end{bmatrix} K \mathbf{R}_{cr} = \begin{bmatrix} a_1^s & a_2^s & a_3^s \end{bmatrix} \end{cases} \tag{38}$$

5.2. Algorithm for Combined Navigation

The visual/inertial fusion navigation system designed in this paper consists of an IMU (gyroscope, accelerometer) and a visual localization system [25]. Initially, the paper utilizes the visual localization system to obtain the UAV's position in the runway coordinate system. The Kalman filter utilizes the difference between the visual localization system's output position and the current position calculated by the combined navigation algorithm as the measurement information for position error [26].

$$\begin{cases} \dot{\mathrm{q}}_t = \frac{1}{2} \mathrm{q}_t \otimes \mathrm{w}_t \\ \dot{\mathrm{v}}_t = C_{b,t}^n (\mathrm{a}_t) \\ \dot{\mathrm{p}}_t = \mathrm{v}_t \\ \dot{\varepsilon}_r = -\frac{1}{\tau_g} \varepsilon_r + \mathrm{w}_{\varepsilon r} \\ \dot{\nabla}_r = -\frac{1}{\tau_a} \nabla_r + \mathrm{w}_{\nabla r} \end{cases} \tag{39}$$

where

$$\mathrm{w}_t = \mathrm{w}_m - \varepsilon_g \tag{40}$$

$$\mathrm{a}_t = \mathrm{a}_m - \nabla_a \tag{41}$$

In the equation, ω_m represents the gyroscope measurement value, and a_m represents the accelerometer measurement value. Since the above kinematic state equation contains noise terms that cannot be directly eliminated in practical measurements, it is necessary to estimate these noise terms through the Kalman filter [27]. According to the inertial system error model, after eliminating the noise terms, the IMU error and its derivative should be constant. By using the above two sets of equations to calculate the error for each state variable, we can obtain the error state equation for the state variable $[\theta, \mathrm{v}, \mathrm{p}, \varepsilon, \nabla_r]^T$:

$$\begin{cases} \delta\dot{\theta} = -[\omega_{\mathrm{m}} - \varepsilon_r]_\times \delta\theta - \varepsilon_r - \mathrm{w}_\varepsilon \\ \delta\dot{\mathrm{v}} = -C_b^n[\mathrm{a}_m - \nabla_r]_\times \delta\theta - C_b^n \nabla_r - C_b^n \mathrm{w}_\nabla \\ \delta\dot{\mathrm{p}} = \delta\mathrm{v} \\ \dot{\varepsilon}_r = -\frac{1}{\tau_g}\varepsilon_r + \mathrm{w}_{\varepsilon r} \\ \dot{\nabla}_r = -\frac{1}{\tau_a}\nabla_r + \mathrm{w}_{\nabla_r} \end{cases} \tag{42}$$

By taking partial derivatives of each state variable, we can obtain the state-space equations for the following prediction model:

$$\dot{\mathrm{X}}(t) = \mathrm{F}(t)\mathrm{X}(t) + \mathrm{G}(t)\mathrm{w}(t) \tag{43}$$

where

$$\dot{\mathrm{X}}(t) = [\delta\dot{\theta}, \delta\dot{\mathrm{v}}, \delta\dot{\mathrm{p}}, \dot{\varepsilon}_r, \dot{\nabla}_r]^T \tag{44}$$

$$\mathrm{X}(t) = [\delta\theta, \delta\mathrm{v}, \delta\mathrm{p}, \varepsilon_r, \nabla_r]^T \tag{45}$$

The continuous-time state transition matrix is as Equation (46).

$$\mathrm{F}(t) = \begin{bmatrix} -[\omega_m - \varepsilon_r]_\times & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\mathrm{I}_{3\times3} & \mathbf{0}_{3\times3} \\ -C_b^n[\mathrm{a}_m - \nabla_r]_\times & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -C_b^n \\ \mathbf{0}_{3\times3} & \mathrm{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\frac{1}{\tau_g}\mathrm{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\frac{1}{\tau_a}\mathrm{I}_{3\times3} \end{bmatrix} \tag{46}$$

The continuous-time noise matrix is as Equation (47).

$$\mathrm{G}(t) = \begin{bmatrix} -\mathrm{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & -C_b^n & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathrm{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathrm{I}_{3\times3} \end{bmatrix} \tag{47}$$

The noise term is given by Equation (48).

$$\mathrm{w}(t) = [\mathrm{w}_\varepsilon, \mathrm{w}_\nabla, \mathrm{w}_{\varepsilon r}, \mathrm{w}_{\nabla r}]^T \tag{48}$$

Since the position output by the visual localization system is in a discrete form, the measurement equation in the discrete form can be represented as Equation (49).

$$\mathrm{Z}_{k+1} = \mathrm{H}_{k+1}\mathrm{X}_{k+1} + \mathrm{v}_{k+1} \tag{49}$$

where

$$\mathrm{Z}_{k+1} = \begin{bmatrix} X_n - \hat{p}_x \\ Y_n - \hat{p}_y \\ Z_n - \hat{p}_z \end{bmatrix} \tag{50}$$

$[X_n, Y_n, Z_n]^T$ represents the three-dimensional position coordinates of the UAV in the runway coordinate system, calculated by the visual localization system.

The measurement noise v_{k+1} can be represented as Equation (51).

$$v_{k+1} = n_p \tag{51}$$

And n_p satisfies Equation (52).

$$n_p \sim N(0, \sigma_{n_p}^2) \tag{52}$$

The measurement equation can be represented as Equation (53).

$$\begin{bmatrix} X_n - \hat{p}_x \\ Y_n - \hat{p}_y \\ Z_n - \hat{p}_z \end{bmatrix} = H_p X_{k+1} + n_p \tag{53}$$

where

$$H_p = [\mathbf{0}_{3\times6} \quad I_{3\times3} \quad \mathbf{0}_{3\times6}] \tag{54}$$

In summary, the structure of the visual/inertial navigation system developed in this paper is shown in Figure 5.

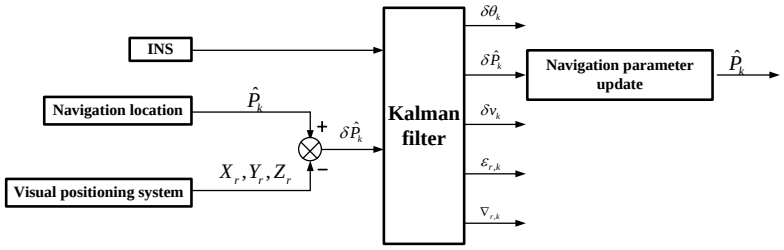
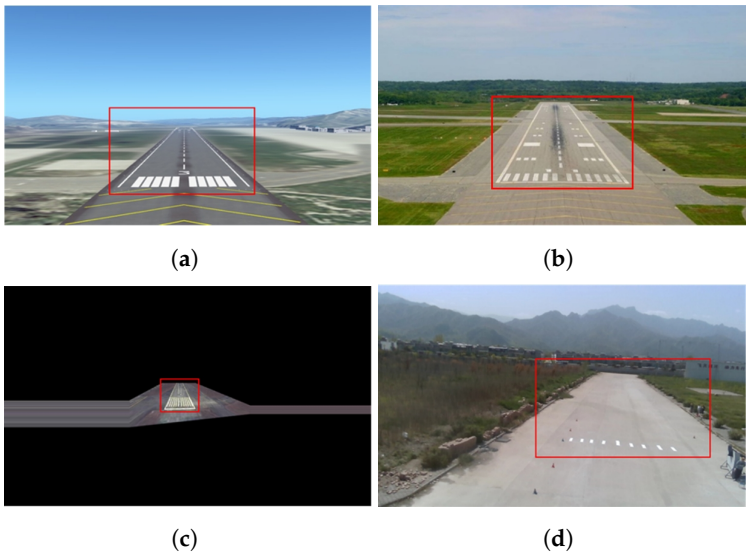


Figure 5. Vision/inertial combined navigation structure diagram.

6. Simulation Results of Detection and Localization Algorithms

6.1. Runway ROI Selection Network Training and Testing

This paper tests the optimized network, and the experimental results are shown in Figure 6. The runway ROI selection network can give the pixel location, category, and probability of the runway, which lays the foundation for the subsequent segmentation of the image based on the pixel location and detection of the runway line.



(**a**) (**b**)

(**c**) (**d**)

Figure 6. Prediction results of the runway ROI selection network. (**a**) Airport runway line detection results in virtual environment; (**b**) Real airport runway line detection results; (**c**) Airport runway line detection results after transformation; (**d**) Simple airport runway line detection results.

This paper also tested the performance of the improved model on a constructed dataset and plotted the P-R curves before and after optimization. The experimental results are shown in Figure 7, where YOLOX represents the model before optimization, and YOLOX$^+$ is used to represent the optimized model for ease of description. From the test results, it can be observed that the P-R curve of the optimized model has a larger enclosed area with respect to the horizontal and vertical axes. This confirms the effectiveness of the proposed optimization measures in improving the model's performance.

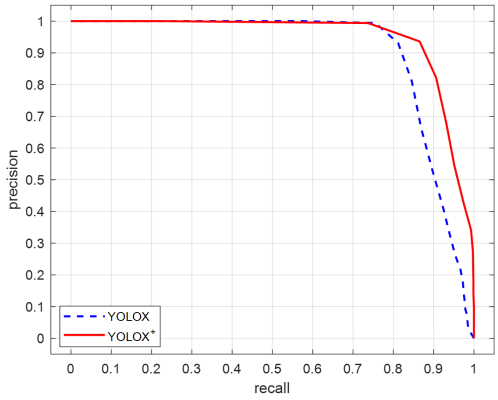
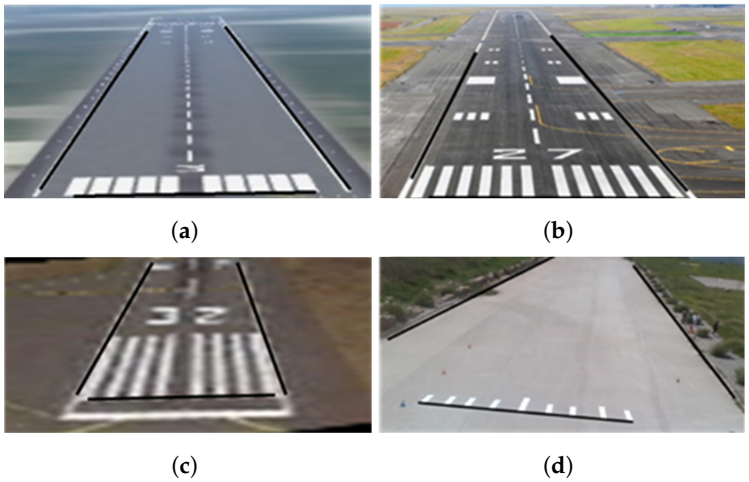


Figure 7. The P-R curves before and after the improvement of YOLOX.

6.2. Runway Line Detection Network Training and Testing

In this paper, the input image resolution for training the runway line detection network is set to 480 × 480. The Adam optimization strategy is chosen for model optimization due to its ease of implementation, efficiency, and low memory requirements. Additionally, a cosine decay strategy is employed to update the learning rate, with an initial learning rate set to $4 \times e^{-4}$. The batch size is set as 32, and the training process is conducted for 100 epochs. The weights assigned to the classification loss (α), structural loss (β), segmentation loss (γ), and correlation loss (θ) are set to 1, 1, 1, and 0.6, respectively. The row/column anchor box grid is configured with 50 cells.

The trained network is tested using a partitioned test dataset, and the predicted results of the network are fitted using the least squares method. The experimental results are illustrated in Figure 8. From the test results, it can be observed that the runway line detection network accurately predicts the positions of the three runway lines.



(**a**) (**b**)

(**c**) (**d**)

Figure 8. Prediction results of the runway line detection network. (**a**) Airport runway line detection results in virtual environment; (**b**) Real airport runway line detection results; (**c**) Airport runway line detection results after transformation; (**d**) Simple airport runway line detection results.

6.3. Visual Localization Simulation Results

In this paper, a test dataset was used to validate the localization algorithm. The test dataset includes the runway width, relative position between the UAV and the runway, and relative attitude angles. The UAV's flight trajectory in the dataset is illustrated in Figure 9, where x and y represent the UAV's position in the runway coordinate system, and height represents the UAV's altitude above the ground.

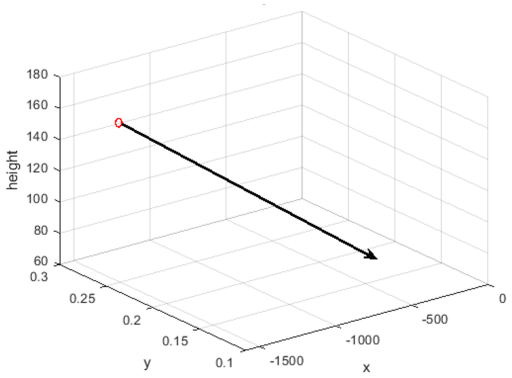


Figure 9. Schematic diagram of the flight path of the drone.

The visual localization results are shown in Figure 10. From the experimental results, it can be observed that the visual localization results follow a similar trend to the ground truth. However, there is some deviation and significant fluctuations between the visual localization results and the ground truth in the initial stage. This deviation is mainly due to the UAV being farther from the runway, resulting in a smaller representation of the runway in the image. Under the same detection accuracy, this leads to larger pixel errors. As the UAV comes closer to the runway, the deviation between the visual localization results and the ground truth gradually reduces.

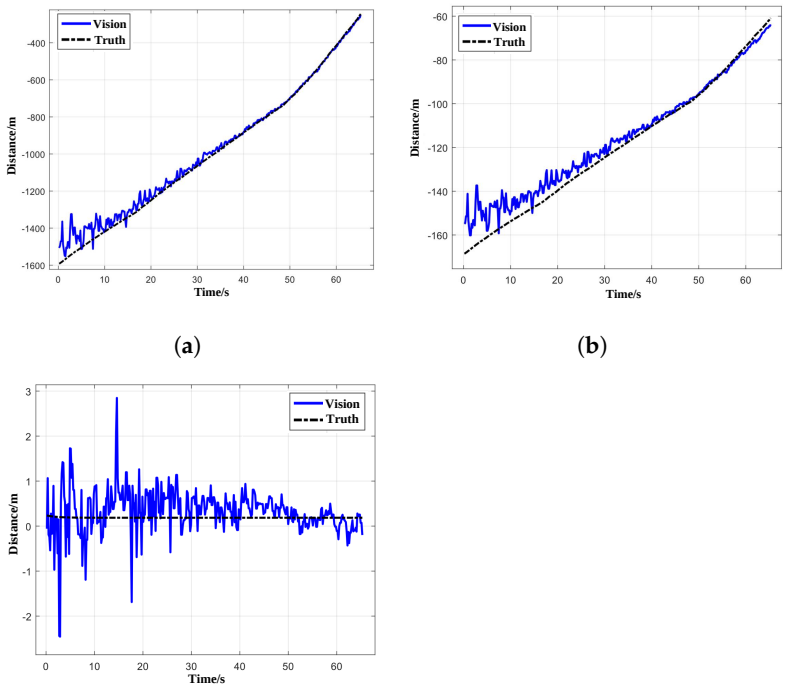


Figure 10. Visual positioning simulation results. (**a**) x-direction visual positioning results; (**b**) z-direction visual positioning results; (**c**) y-direction visual positioning results.

To quantitatively describe the error characteristics of visual localization, this paper calculated the MAE (mean absolute error) and RMSE (root-mean-square error) of the visual localization in three directions. The calculation results are shown in Table 2. Due to the larger pixel errors as the UAV is farther from the runway, the localization errors in all three directions are larger and more fluctuating in the initial stage, with some outliers present. As the UAV approaches the runway, the localization errors in all three directions gradually reduce, and the number of outliers decreases. Additionally, from the error curves, it can be observed that when approaching the runway, the localization error in the x-direction is approximately 4 m. Since the runway has a certain margin in the x-direction, this localization error meets the landing requirements for the UAV on the runway. The localization error in the y-direction is about 0.3 m, which can satisfy the accuracy requirement for UAV landing. However, the localization error in the z-direction is around 2.5 m, which indicates the need for further fusion with other sensor data to improve the positioning accuracy in the z-direction.

Table 2. Visual positioning error characteristics.

Axis	Evaluation Metrics	
	MAE (m)	RMSE (m)
x-axis	23.7810	40.375
y-axis	0.3548	0.5030
z-axis	4.3948	6.2123

6.4. Simulation Results of Combined Navigation Algorithm

During the simulation process, this paper set the time update frequency and measurement update frequency of the Kalman filter and the adaptive fading Kalman filter to 100 Hz and 10 Hz, respectively. Additionally, the initial state covariance matrix P_0, process noise covariance matrix Q_k, and measurement noise covariance matrix R_k for the traditional Kalman filter and the adaptive fading Kalman filter were set as Equation (55).

$$\begin{cases} P_0 = diag\{(0.1^\circ)^2, (0.1^\circ)^2, (0.1^\circ)^2, \\ \qquad (0.1\ \text{m/s})^2, (0.1\ \text{m/s})^2, (0.1\ \text{m/s})^2, \\ \qquad (1\ \text{m})^2, (1\ \text{m})^2, (1\ \text{m})^2, \\ \qquad (0.01)^2 \cdot I_{3\times3}, (0.01)^2 \cdot I_{3\times3}\}_{15\times15} \\ Q_k = diag\{(0.01\ \text{rad/s})^2 \cdot I_{3\times3}, (0.01\ \text{m/s}^2)^2 \cdot I_{3\times3}, \mathbf{0}_{9\times9})\}_{3\times15} \\ R_k = diag\{(40\ \text{m})^2, (5\ \text{m})^2, (10\ \text{m})^2\}_{3\times3} \end{cases} \tag{55}$$

This paper compared the localization results of visual localization, traditional Kalman filter-based combined navigation, and adaptive fading Kalman filter-based combined navigation [28,29]. The experimental results are shown in Figure 11. In the initial stage, where the UAV is far from the runway and the runway occupies a small portion of the camera's field of view, the pixel errors on the image processing side are larger. As a result, the visual localization results in the initial stage exhibit significant fluctuations and outliers. The traditional Kalman filter adjusts the weights of the state prediction and innovation in the state prediction process based on the state noise covariance matrix and measurement noise covariance matrix. This helps to reduce the number of outliers to some extent. However, the filtering performance of the traditional Kalman filter heavily relies on the accuracy of the measurement noise covariance matrix. If the statistical properties of the measurement noise are not well understood, the improvement in filtering accuracy may be limited. In the adaptive fading Kalman filter, an adaptive factor is introduced, which reduces the dependence on the accuracy of the measurement noise covariance matrix [30]. This enables better removal of outliers in the visual localization results, resulting in improved filtering performance.

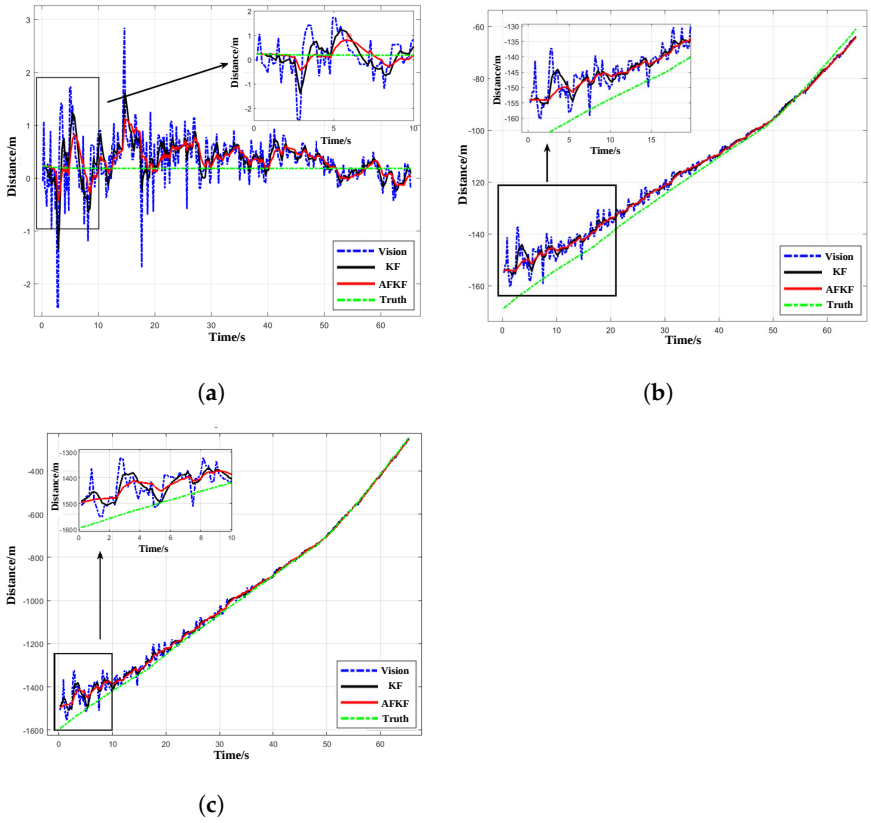


Figure 11. Comparison results of the true value, visual localization, combined navigation based on traditional Kalman filter and combined navigation based on adaptive fading Kalman filter. (**a**) Positioning result in y-direction; (**b**) Positioning result in z-direction; (**c**) Positioning result in x-direction.

In order to quantitatively describe the error characteristics of the two filtering algorithms, this paper calculates the MAE and RMSE of the two filtering algorithms in three directions, and the experimental results are shown in Table 3. From the experimental results, it can be seen that the MAE and RMSE of the traditional Kalman filter and the adaptive fading Kalman filter are smaller than the visual positioning results, and the accuracy has been improved. Since the adaptive factor is added to the adaptive fading Kalman filter, the covariance matrix of the measurement can be dynamically adjusted, thereby reducing the proportion of the measurement in the prediction and estimation, so the filtering effect is better, and its MAE and RMSE are better than those of the traditional Kalman filter. The Mann filter is small, and the positioning accuracy is further improved. The experimental results show the effectiveness of the combined navigation algorithm designed in this paper.

Table 3. Error characteristics of combined navigation algorithm.

Filtering Algorithm	Axis	Evaluation Metrics	
		MAE (m)	RMSE (m)
KF	x	22.7146	36.6027
	y	0.2819	0.3673
	z	4.3260	5.7779
AFKF	x	22.7146	36.6027
	y	0.2819	0.3673
	z	4.3260	5.7779

6.5. Flight Test in Real Scenario

6.5.1. Model Compression and Acceleration

To accelerate the model training process, this paper utilized a high-performance NVIDIA GeForce RTX3090 during training. However, the trained model needs to be deployed on an onboard computer with limited computational power. Direct deployment may result in insufficient inference speed and latency, which may not meet the real-time requirements for UAV landing navigation. Furthermore, if models built using different deep learning frameworks are directly deployed, conflicts may arise, preventing them from running simultaneously on the onboard computer. Therefore, it is necessary to use a unified framework to refactor these models. In this paper, when deploying the runway coarse positioning network and the runway line detection network on the onboard computer, ONNX and TensorRT were used to compress and accelerate both models [31]. The program execution flow is illustrated in Figure 12. The speed of the algorithm on an NVIDIA Jetson Xavier NX is shown in Table 4. The results show that the real-time requirements of the algorithm can be achieved after model compression and acceleration.

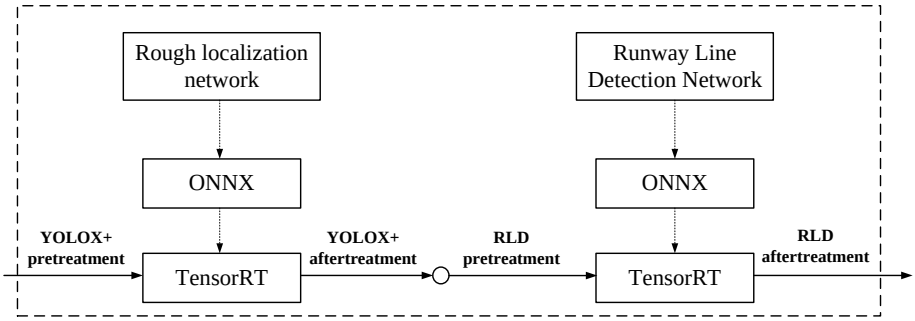


Figure 12. Model compression and acceleration process.

Table 4. Comparison of runway line detection and localization algorithm update frequency.

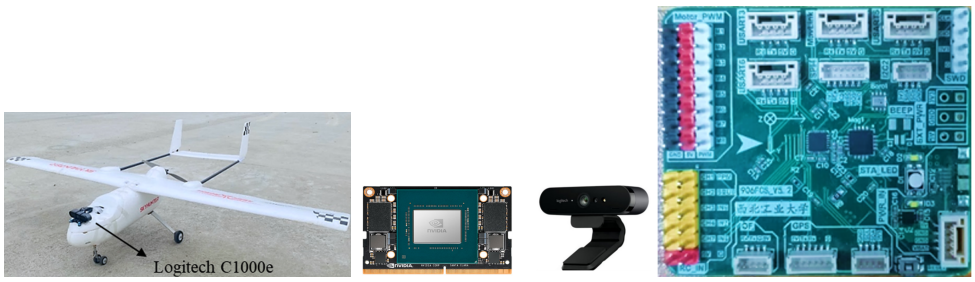
	ONNX and TensorRT	Update Frequency (Hz)
Detection algorithms	×	44
	√	56
Detection and localization algorithms	×	40
	√	50

6.5.2. Introduction of Flight Test Equipment

The fixed-wing UAV platform built during the actual flight test is shown in Figure 13a, with a wingspan of 1.8 m and a maximum flight height of 120 m. The equipment carried by the UAV mainly includes cameras, onboard computers, flight controllers, and power supplies. Equipment such as the control and power supply are installed inside the fuselage. The runway coarse positioning network, runway line detection network, and visual positioning algorithm are deployed on the onboard computer for image processing and visual navigation data calculation, and the visual navigation results are transmitted to the flight control terminal through USART. After the control rate in the flight controller solves the control command, it controls the steering gear in the form of PWM, and finally, controls the UAV to land smoothly.

The onboard computer selected in this paper is an NVIDIA Jetson Xavier NX. Due to its powerful computing power and small size, it can be widely used in drones, small robots, and security systems. Considering the image quality, volume, weight and other factors, the camera selected in this paper is a Logitech C1000e in the flight test, and the onboard computer and camera are shown in Figure 13b. In addition, the flight control system can ensure the stability and controllability of the UAV flight. The flight control used in this

paper is self-developed flight control, an onboard IMU, barometer, magnetometer, and other common sensors. The physical picture is shown in Figure 13c.



(**a**) (**b**) (**c**)

Figure 13. Real flight test platform. (**a**) UAV platform; (**b**) Onboard computer and camera; (**c**) Physical diagram of flight control system.

6.5.3. Flight Test Results

In the work, the landing process of the UAV on a simplified runway, treating the left and right edges of the runway as the left and right runway lines are simulated, respectively. Additionally, a starting runway line was added to the simplified runway. The runway line detection results are shown in Figure 14. From the experimental results, it can be observed that the designed model accurately detects all three runway lines, demonstrating good robustness and generalization.



Figure 14. Results of real runway line detection.

In the flight experiment, the visual localization results and the corresponding localization error curves in the x-, y-, and z-directions are shown in Figure 15. Due to various factors such as the short length of the simplified runway, surrounding building interference, fast UAV flight speed, and significant UAV maneuverability, the runway appears in the camera's field of view for a short time. As a result, there is limited effective localization data, and the errors are larger compared to the simulation results. However, overall, the localization trends in all three directions align with the ground truth.

From the error curves, it can be observed that the localization error in the x-direction gradually decreases as the UAV approaches the runway, reaching around 2 m near the runway. Since the runway has some margin in the x-direction, this accuracy level is sufficient for UAV landing requirements. In the y-direction, the localization error is around 2.7 m, which deviates significantly from the simulation results. The reason could be the curved edges of the left and right runway lines and possible image distortion due to the rolling shutter effect of the camera, or there may be some deviation in the ground truth setting. In the z-direction, the localization error is approximately 1.5 m, which cannot be applied to autonomous navigation during the UAV landing process. Therefore, it is necessary to further incorporate data such as IMU and laser sensor data to improve the localization accuracy.

In addition, In the work, the error characteristics of the localization results in the three directions is also calculated, as shown in Table 5. The accuracy of the errors in the y-axis and z-axis is low, and the root-mean-square error in the x-axis reaches 22.1081 m. However, given that the accuracy in the x-direction is not very demanding during the

flight landing process, the visual localization algorithm proposed in this paper is effective. Furthermore, in the flight experiment, only visual localization is validated. If the adaptive fading extended Kalman filter algorithm mentioned earlier is further used to fuse the visual localization results and IMU data, it can theoretically improve the localization accuracy.

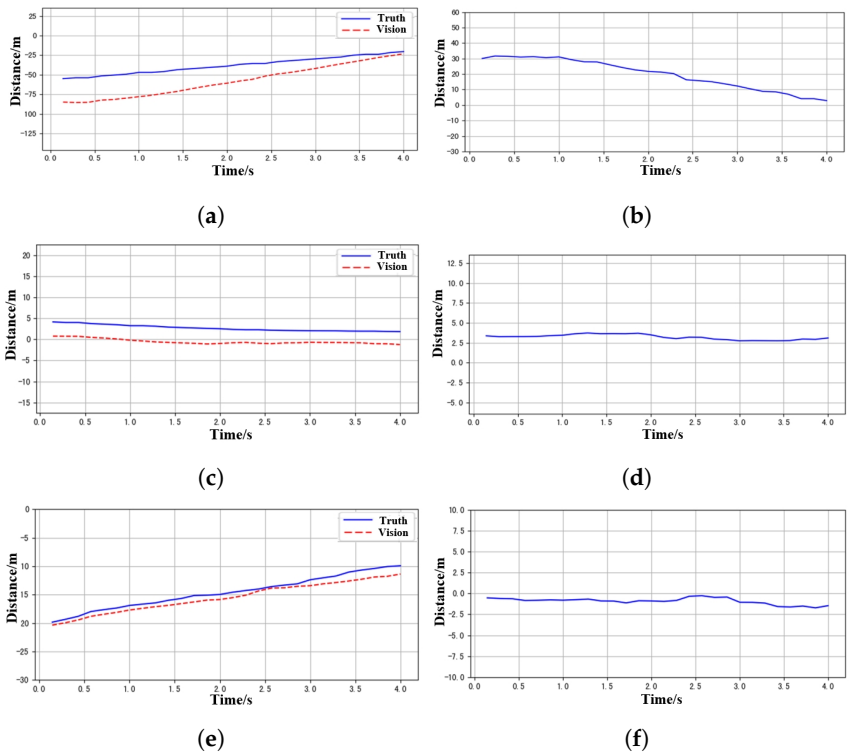


Figure 15. Experiment error curves in real scenario. (**a**) Visual positioning error in x-direction; (**b**) Visual positioning error in x-direction; (**c**) Visual positioning error in y-direction; (**d**) Visual positioning error in y-direction; (**e**) Visual positioning error in z-direction; (**f**) Visual positioning error in z-direction.

Table 5. Error characteristics of visual localization algorithm.

Axis	Evaluation Metrics	
	MAE (m)	RMSE (m)
x-axis	19.8979	22.1081
y-axis	3.2175	3.2331
z-axis	0.9074	0.9831

7. Conclusions

In recent years, UAVs have been used in various fields on a large scale. Landing as an important stage of flight, realizing unmanned autonomous landing is of great significance for UAV intelligence. This paper takes the visual navigation algorithm for autonomous UAV landing as the research purpose, constructs an end-to-end visual guidance landing navigation system, and optimizes the detection algorithm at the image processing end, and fuses the IMU information and visual localization data at the localization end by combining the navigation algorithms, in response to the requirements of this paper's application scenarios for accuracy and real-time. The innovations of this paper are as follows:

(1) To meet the requirements of UAV visual-guided landing, a deep-learning-based system for runway ROI detection, runway line detection, visual localization, and combined navigation is constructed.

(2) The paper optimizes the runway ROI detection algorithm and runway line detection algorithm to meet the navigation accuracy and real-time performance requirements in the application scenario.

(3) To further improve visual localization accuracy, the paper utilizes the Kalman filtering algorithm to fuse IMU information and visual localization results.

Simulation and experimental results demonstrate the significant advantages of the proposed algorithms in terms of detection accuracy, real-time performance, and generalization ability. The paper provides a reliable solution for the visual navigation problem in UAV landing.

Author Contributions: Conceptualization, X.L. and W.X.; methodology, W.X.; software, W.X.; validation, W.X. and X.X.; formal analysis, W.X.; investigation, W.X.; resources, X.L.; data curation, X.L., W.X., B.Q., and X.X.; writing—original draft preparation, W.X.; writing—review and editing, W.X., X.X., and B.Q.; visualization, W.X., X.X., and M.Z.; supervision, X.L.; project administration, X.L.; funding acquisition, X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number No.62073266, and the Aeronautical Science Foundation of China, grant number No.201905053003.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Acknowledgments: Gratitude is extended to the Shaanxi Province Key Laboratory of Flight Control and Simulation Technology.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Jiao, S.; Ding, H.; Zhong, Y.; Yao, X.; Zheng, J.J. A UAV Target Tracking and Control Algorithm Based on SiamRPN. *J. Syst. Simul.* **2023**, *35*, 1372–1380.
2. Xu, W.; Li, P.; Han, B. An attitude estimation method for MAV based on the detection of vanishing point. In Proceedings of the 2010 8th World Congress on Intelligent Control and Automation, Jinan, China, 7–9 July 2010; pp. 6158–6162.
3. Chong-ming, W.; Xiao-dan, W.; Jin, G.; Wen, Q. Image Matching Method Based on the Modified Hough Transform and the Line Characteristics. In Proceedings of the 2010 Chinese Conference on Pattern Recognition (CCPR), Beijing, China, 22–24 October 2008.
4. Feng, G.; Zhang, D.; Wu, W. Pose estimation of moving object based-on dual quaternion from monocular camera. *Geomat. Inf. Sci. Wuhan Univ.* **2010**, *35*, 1147–1150.
5. Zhenghong, X.; Qiang, Z.; Xinping, Z. Research on photoelectric surveillance warning system and design scheme for airport surface. *China Saf. Sci. J.* **2020**, *30*, 136.
6. Wei-Dong, Z.; Xiao-Cheng, L.; Peng, H. Progress and challenges of overwater unmanned systems. *Acta Autom. Sin.* **2020**, *46*, 847–857.
7. Tao, L.; Hong, T.; Chao, X. Drone identification and location tracking based on YOLOv3. *Chin. J. Eng.* **2020**, *42*, 463–468.
8. Liu, J.; Wang, W.; He, Q.; Kong, X.; Ye, B.; Wang, S. Autonomous patrol technology and system of leapfrogcharging UAV (II): Automatic charging control based on machine vision. *J. Electr. Power Sci. Technol.* **2022**, *36*, 182–188.
9. Yin, H.; Zhang, X.; Zhang, X.; Liu, X. Interference analysis to aerial flight caused by UHV lines using airborne GPS. *Geomat. Inf. Sci. Wuhan Univ.* **2009**, *34*, 774–777.
10. Zhang, L.; Wang, Y.J.; Sun, H.H.; Yao, Z.J.; Wu, P. Adaptive scale object tracking with kernelized correlation filters. *Guangxue Jingmi Gongcheng Optics Precis. Eng.* **2016**, *24*, 449–459.
11. Liu, X.; Li, C.; Xu, X.; Yang, N.; Qin, B. Implicit Neural Mapping for a Data Closed-Loop Unmanned Aerial Vehicle Pose-Estimation Algorithm in a Vision-Only Landing System. *Drones* **2023**, *7*, 529. [CrossRef]
12. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. Yolox: Exceeding yolo series in 2021. *arXiv* **2021**, arXiv:2107.08430.
13. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
14. Weng, K.; Chu, X.; Xu, X.; Huang, J.; Wei, X. EfficientRep: An efficient RepVGG-style convnets with hardware-aware neural network design. *arXiv* **2023**, arXiv:2302.00386.
15. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. Ghostnet: More features from cheap operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 1580–1589.

16. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27–28 October 2019; pp. 1314–1324.
17. Misra, D. Mish: A self regularized non-monotonic activation function. *arXiv* **2019**, arXiv:1908.08681.
18. Liu, S.; Huang, D.; Wang, Y. Learning spatial fusion for single-shot object detection. *arXiv* **2019**, arXiv:1911.09516.
19. Li, H.; Li, J.; Wei, H.; Liu, Z.; Zhan, Z.; Ren, Q. Slim-neck by GSConv: A better design paradigm of detector architectures for autonomous vehicles. *arXiv* **2022**, arXiv:2206.02424.
20. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
21. Wang, Q.; Wu, B.; Zhu, P.; Li, P.; Zuo, W.; Hu, Q. ECA-Net: Efficient channel attention for deep convolutional neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 11534–11542.
22. Qin, Z.; Wang, H.; Li, X. Ultra fast structure-aware deep lane detection. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Part XXIV 16. Springer: Berlin/Heidelberg, Germany, 2020; pp. 276–291.
23. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
24. Jing-hong, L.; Bo, J.; Gang, L.; Qian-fei, Z. Geometric correction of oblique images for array CCD aerial cameras. *Chin. J. Liq. Cryst. Displays* **2015**, *30*, 505–531.
25. Liu, J.; Zhao, Z.; Hu, N.; Huang, G.; Gong, X.; Yang, S. Summary and prospect of indoor high-precision positioning technology. *Geomat. Inf. Sci. Wuhan Univ.* **2022**, *47*, 997–1008.
26. Jing, B.; Jian-ye, L.; Xin, Y. Study of fuzzy adaptive kalman filtering technique. *Inf. Control* **2002**, *31*, 193–197.
27. Chen, X.; Zhou, J.; Li, J.; Guo, L. Data processing of wind profiler radar based on nonlinear filtering. *Nanjing Xinxi Gongcheng Daxue Xuebao* **2013**, *5*, 533.
28. Llerena Caña, J.P.; García Herrero, J.; Molina López, J.M. Error Reduction in Vision-Based Multirotor Landing System. *Sensors* **2022**, *22*, 3625. [CrossRef]
29. Wubben, J.; Fabra, F.; Calafate, C.T.; Krzeszowski, T.; Marquez-Barja, J.M.; Cano, J.C.; Manzoni, P. Accurate landing of unmanned aerial vehicles using ground pattern recognition. *Electronics* **2019**, *8*, 1532. [CrossRef]
30. Gao, W.; Yang, Y.; Cui, X.; Zhang, S. Application of adaptive Kalman filtering algorithm in IMU/GPS combined navigation system. *Geo-Spat. Inf. Sci.* **2007**, *10*, 22–26. [CrossRef]
31. Yang, X.; Jiawei, W.; Jianxue, L.; Jun, L. A dynamic routing algorithm based on deep reinforcement learning. *Inf. Commun. Technol. Policy* **2020**, *46*, 48.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.