STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 2026, pp 0–17.
Published online in International Academic Press (www.IAPress.org)

Accurate and Real-time Localization of Unmanned Aerial Vehicles using Satellite Images based on Swin Transformer

Ahmed Imran Fattah[1,2*], Mohamad Mahdi Kassir[1], and Amir Lakizadeh[1]

[1] *Computer Engineering and Information Technology Department, University of Qom, Qom, Iran*
[2] *Department of Cyber Security, Imam Al-Kadhim University College, Baghdad, Iraq*

Abstract In recent years, attention has grown significantly in the unique applications of unmanned aerial vehicles (UAVs) in various fields. These applications generally require precise localization. Reliance on GPS poses issues like interference and signal loss. Recent studies suggest alternatives such as Google Earth imagery and neural networks. While promising, these methods still struggle with low accuracy and unreliable performance, partly due to limited attention to hyperparameter optimization (HPO) and feature extraction. To improve this class of methods, in this research, an accurate and fast image-based UAV localization method without using GPS is presented. The proposed method has two general stages: pre-flight (training) and flight (testing and localization). In the first stage, the desired flight path is taught to the drone offline. For this purpose, images of the path are sampled and then the data augmentation technique is used to augment the input data to the Swin Transformer network and increase its accuracy in the training process. In order to further improve the efficiency of the Swin Transformer network, the HPO method is used. After the internal compression by the Swin Transformer coder, an external feature extraction stage is performed based on the vector quantization method (VQM) and the Principal Component Analysis (PCA). Then the compressed vectors are stored in the drone memory. In the second stage, the drone records each live image; first it encodes it, then an external feature extraction stage is performed on the encoded vector, and localization is completed with kernel operations and comparison with memory. Simulation results show that the proposed method performs localization operations with low latency (1.65-fold reduction), reasonable memory requirements, and high accuracy (RMSE $\approx$ 1.07 m).

Keywords Unmanned Aerial Vehicles (UAVs), Localization, Satellite Images, Swin Transformer, Hyperparameter Optimization (HPO), Data Augmentation, Feature Extraction

AMS 2010 subject classifications: 68T45, 68U10, 68T40

DOI: 10.19139/soic-2310-5070-4316

1. Introduction

In recent years, the use of UAVs [1] has become increasingly common in a wide range of outdoor applications, including surveillance [2], agricultural monitoring [3, 4], package delivery, and emergency response [5, 6, 7]. Figure 1 shows the application of UAV in smart agriculture [4]. In recent years, many countries have believed in the unique power of this tool for monitoring agricultural products [8]. However, the reliance of UAVs on GPS for localization has certain disadvantages, such as susceptibility to dropout, interference, and intentional jamming. Also, the possible failure of GPS systems raises safety concerns, restricts UAVs to line-of-sight flights, and limits their operational capabilities [9]. To address these problems, studies such as Bianchi et al. [10] have attempted to use pre-rendered Google Earth (GE) images to train an Autoencoder to compress the vector of the path images and perform accurate image localization. Their method estimates the location of the aircraft with acceptable accuracy and very low computational time by comparing the compressed vector of the live UAV image with the GE reference

*Correspondence to: Ahmed Imran Fattah (Email: ahmed.omran@iku.edu.iq). Department of Cyber Security, Imam Al-Kadhim University College, Baghdad, Iraq.

ISSN 2310-5070 (online) ISSN 2311-004X (print)
Copyright © 2026 International Academic Press

images [10]. Although this method has the advantage of not requiring GPS, it still depends on the accuracy of the Autoencoder and lacks structural optimization in the network architecture. On the other hand, using only internal image features may cause poor localization performance in complex lighting conditions or environmental changes [10].

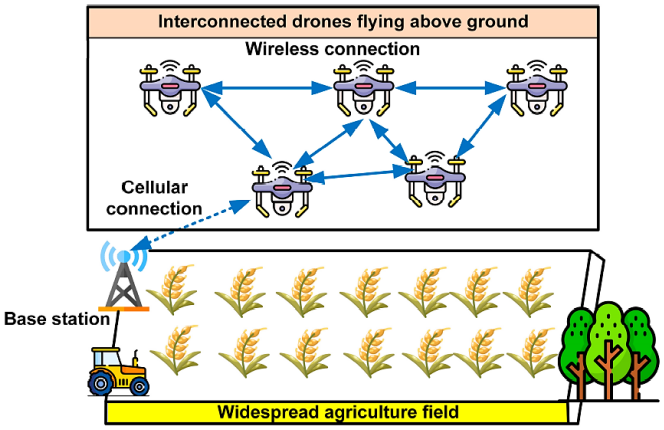

Figure 1. Application of UAV in smart agriculture [4]

Although image-based UAV localization has received increasing attention, many existing methods still face a practical trade-off between localization accuracy, computational latency, and deployment cost. In GPS-denied scenarios, a method is not only required to be accurate, but also sufficiently lightweight to support online operation with limited onboard resources. This challenge becomes more critical when the localization pipeline must compare incoming UAV images against a dense satellite-image database over extended trajectories. Therefore, the main motivation of this work is to design a localization framework that preserves competitive accuracy while reducing runtime and memory burden to make practical deployment more feasible. Therefore, in this study, an accurate and fast system for UAV localization in non-GPS environments is proposed that uses GE data to train the network. However, our key innovations provide significant improvements over the Bianchi method [10]:

i. Real-time UAV localization based on images and without the use of GPS.
ii. Using HPO methodology to increase the accuracy of the Swin Transformer network.
iii. Using two stages of internal feature extraction (Swin Transformer coder) and external feature extraction (VQM-PCA) for better compression.
iv. Using data augmentation techniques to increase the accuracy of the Swin Transformer coder network.

The rest of the paper is as follows. In the second section, a review of research related to UAV localization is done. In the third section, the proposed localization method is presented in two general phases, and the simulation of this method is done in the fourth section of the paper.

2. Literature Review

Past studies can be categorized in various ways. In this study, we have created four categories based on the localization approach.

2.1. Edge Detection and Semantic Segmentation-based Methods

Early methods [11, 12] use vision-based systems integrating inertial sensors and visual odometry with geo-referenced imagery to enable GPS-free navigation. These systems demonstrate promising accuracy, especially at higher altitudes where buildings and roads are present. However, they struggle in texture-less areas like grasslands.

2.2. Standard Features-based Methods

Feature-based approaches (e.g., SIFT, SURF, BRIEF) [13] enable UAV localization in urban settings by matching aerial views with ground-level geotagged databases. Tests in Zurich show robustness against viewpoint and seasonal changes, but low-level features remain sensitive to illumination and modality variations.

2.3. Custom Detectors-based Methods

To address the limitations of point features, some works [14] propose detecting larger structures or using mutual information [15, 16]. Reference [14] introduces unsupervised classifiers trained on urban datasets, significantly reducing localization failure rates. Reference [9] uses normalized mutual information for alignment with pre-rendered Google Earth images but is computationally expensive and unsuitable for real-time use. Obviously, processing time is one of the most important criteria for localization. Any method, no matter how accurate, will be poor if it is not real-time. Some comprehensive reviews are also presented in this section. For example, a systematic and comprehensive review [17] of the ORB-SLAM framework, with an emphasis on its application in UAV navigation and mapping, provides an overview.

2.4. Deep Learning-based Methods

Deep learning methods [18, 19] predict global pose from a single image but require large datasets. Swin Transformers have been used to encode features for robust matching [20, 21, 22, 23, 24]. The method in [10] improves speed by compressing satellite and real-time UAV images into vector representations using a Swin Transformer, enabling rapid matching via inner product kernels. While effective (RMSE < 3 m), it lacks hyperparameter optimization, an issue addressed in this work. Hyperparameter tuning has also been discussed in other studies [25, 26, 27, 28, 29].

3. Proposed Method

The aim of the approach presented in this research is to perform UAV localization based only on images and without using GPS. Therefore, the proposed method of this research has two general phases: Phase 1: Pre-flight (training) and Phase 2: Flight (testing and localization). In the first phase, the UAV must be taught the desired flight path. The purpose of training is to enable the UAV to extract unique patterns from the existing images. For this purpose, Google Earth images must be downloaded offline from the map and trained on the UAV. In this phase, the images of the path are sampled and then the data augmentation technique is used to augment the input data to the deep network (Swin Transformer) and increase its accuracy in the training process. The UAV system has a Swin Transformer that is responsible for compressing and training the images. In order to further improve the efficiency of this deep network, the HPO mechanism has also been used. As shown in Figure 2, the Swin Transformer network consists of two separate parts: the Swin Transformer coder and the Swin Transformer decoder. The Swin Transformer decoder section is used to check the reconstruction error and train the network (offline) and is practically useless after the Swin Transformer coder is trained.

One of the main requirements of the proposed method for real-time localization with low computational cost is data compression. Therefore, another step for feature extraction is also considered in the proposed method and is called the external feature extraction step. After internal compression by the Swin Transformer coder and external feature extraction, the compressed images are stored in the UAV memory. Thus, the pre-flight phase is completed. In the second phase, which is also called the flight or localization phase, the goal is for the UAV to be able to perform localization online and in real time by examining live images. In this phase, the Swin Transformer coder section of the Swin Transformer network as well as the information stored in the memory is uploaded to the UAV electronic board. By recording each live image, the UAV first codes it (by passing through the Swin Transformer coder network), then an external feature extraction step is performed on the coded vector and with kernel operations (comparison with memory), the UAV location is determined, and localization is completed. The proposed method performs localization operations in real time without the need for GPS and relying only on images.

3.1. Phase 1: Pre-Flight

In the first phase, the main goal is to train the Swin Transformer coder part of the Swin Transformer network and store information in memory.

3.1.1. Choice of Desired Path The first step in the proposed approach for localization is to choose the desired path. The desired path is the path that the UAV should follow by only imaging and examining the environment. This path varies depending on the type of application. For example, if the purpose of using the UAV is to monitor agricultural products, this path could be a useful area of a field. Or if the purpose of the UAV is to investigate and assist in an incident, this path could be its protection area. Regarding the size of the desired path, the main limitation is related to the hardware installed on the UAV. In short, the two main factors that should be considered in path design are battery capacity and memory capacity.

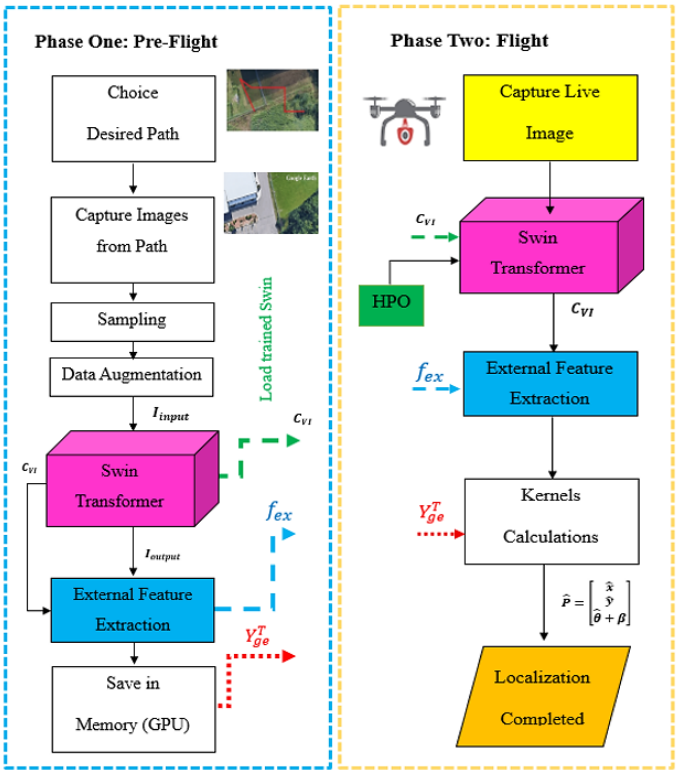

Figure 2. Block diagram of the proposed method and the structure of the Swin Transformer network used in this research. Phase One: Pre-Flight (Left), Phase Two: Flight (Right)

3.1.2. Capture Images from Path In the next step, after determining the desired path, the images of that path must be extracted offline and the Swin Transformer trained using them. The best source for this is GE images.

3.1.3. Sampling Obviously, capturing images of an entire path will create a long sequence of frames. Therefore, it is better to separate a part of the images by sampling. In this research, according to the desired path, images from GE in the desired orientation are presented every 0.5 m along the path. Additional images are presented at a distance of 0.5 m lateral to 5 m on both sides of the path. This process requires 42 images for each meter of the path. The sampled images along the path must be fed into the Swin Transformer neural network. This network learns the path according to these images.

3.1.4. Data Augmentation In this research, 6 images are stored for each sampled image. These images are created from the base image by rotating it as follows:

$$\mathbf{I_{input}} \rightarrow \left[\mathbf{I_{input}}_0, \mathbf{I_{input}}_{\mp 45}, \mathbf{I_{input}}_{\mp 90}, \mathbf{I_{input}}_{180}\right] \tag{1}$$

These angles (0°, 60°, 120°, 180°, 240°, and 300°) were selected to provide uniform orientation coverage while keeping the offline database size computationally manageable. We adopted six rotations as a practical compromise between viewpoint diversity and storage/computation cost. Regardless of the sampling method and data augmentation technique, although data augmentation leads to increased accuracy, creating this dataset also leads to a large number of images to be processed. Storing and comparing these images at full size is impossible and very heavy. The next step and key aspect of this method is to use the Swin Transformer to compress the images into a low-dimensional representation while preserving the key features of each image.

3.1.5. Training the Swin Transformer To train the path to the UAV, a deep neural network needs to be selected. For this purpose, a Swin Transformer network is used in this research. Accordingly, the input image is fed into the Swin Transformer coder to produce a compressed representation of $\mathbf{C_{VI}}$.

$$\mathbf{C_{VI}} = net_{enc}\left(\mathbf{I_{input}}\right), \qquad \mathbf{C_{VI}} \in \mathbb{R}^n \tag{2}$$

The output of the Swin Transformer coder ($\mathbf{C_{VI}}$) is called the latent feature vector. This vector is actually a compressed set with a proper representation of the input image. The latent vector represents a summary of the input information that stores its main and key features in a low-dimensional space. After the Swin Transformer coder, a Swin Transformer decoder acts exactly the opposite of the Swin Transformer coder to reconstruct the original image from the compressed representation.

$$\mathbf{I_{output}} = net_{dec}\left(\mathbf{C_{VI}}\right) \tag{3}$$

Then a function is modeled as follows, which is used to model the loss between the reconstructed image and the original image. The goal is to minimize the value of this function. In this relation, hp represents the hyperparameters of the model.

$$\mathbf{L_{loss}} = \left(\mathbf{I_{input}} - \mathbf{I_{output}}\right)^2 \quad \rightarrow \quad \min_{hp} \mathbf{L}_{loss} \tag{4}$$

After the value of $\mathbf{L_{loss}}$ is minimized, the training process of the Swin Transformer coder network is finished. In this case, the Swin Transformer coder network can be used to compress and encode other images. In the proposed method, the input is a 320 × 160 grayscale GE image. The Swin Transformer coder consists of six layers. Each of the first five layers performs a two-dimensional convolution with kernel size four and a step of two followed by a batch normalization layer. In the model, the number of channels is doubled. Finally, a linear layer maps the output of the final convolution layer to a bottleneck vector. In this research, different sizes of bottleneck dimensions were tested, but a bottleneck of 1000 was determined, as it was the smallest size that could still achieve the desired accuracy. Using the trained Swin Transformer, 320 × 160 images can be compressed into a 1000-dimensional vector ($\mathbf{C_{VI}}$). This is a significant reduction compared to the 51200 pixels in the original image. Using fewer dimensions to represent the same image speeds up any computation.

$$\mathbf{I_{input}} : 160 * 320 = 51200 \quad \overset{encoder}{\longrightarrow} \quad \left[\mathbf{C_{VI}}\right]_{1*1000} \tag{5}$$

According to the above values, it can be said that the dimensionality is reduced by about 2% by using the Swin Transformer coder. In short, the Swin Transformer coder network has five convolutional layers, each followed by a batch normalization layer. The number of input channels, output channels and step are shown in parentheses. The final Swin Transformer coder layer is a linear layer that produces a 1000-dimensional vector. The first Swin Transformer decoder network layer is a linear layer that maps the bottleneck vector to 1024 channels. There are five more layers, each of which performs sampling by a factor of two, followed by convolution and batch normalization.

The Swin Transformer decoder part of the network behaves in front of the Swin Transformer coder. A linear layer first maps the bottleneck variable to 1024 channels. It is then passed to the first of five layers, each of which is down-sampled by a factor of two, followed by convolution with a kernel size of three and a step of one, and batch normalization. The number of channels in each layer is halved until an output gray image with the same dimensions as the input image is produced. The output after only part of the Swin Transformer coder network is used to obtain the compressed image vector.

3.1.6. Tuning Swin Transformer Hyperparameters The role of Swin Transformers in UAV localization is very important in the proposed method. The Swin Transformer is introduced as a versatile architecture designed to serve as a general-purpose backbone for various computer vision tasks, including both image classification and dense prediction tasks such as object detection and semantic segmentation. While Transformers have revolutionized natural language processing by serving as universal models, applying them directly to vision tasks presents unique challenges due to fundamental differences between the two domains. The Swin Transformer introduces a hierarchical architecture designed to efficiently process high-resolution images by leveraging a series of stages that progressively reduce spatial dimensions while increasing feature complexity. The overall architecture can be seen in Figure 3.

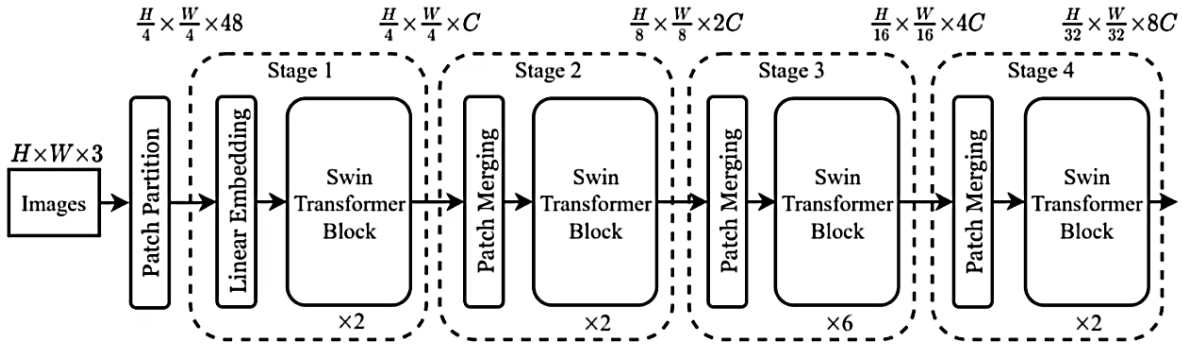

Figure 3. Overall architecture of the Swin Transformer

Another idea of this research is the important issue of HPO in the Swin Transformer. Hyperparameter optimization can help improve the performance and generalization ability of the proposed Swin Transformer in UAV localization.

3.1.7. External Feature Extraction In this research, data augmentation methods were used to increase the accuracy of image coding. Then, the compressed vector of the image with the trained Swin Transformer coder was called $\mathbf{C_{VI}}$. Real-time localization of UAV is a fundamental requirement. If this condition is not met, UAV navigation may face serious problems. For this reason, an external feature extraction step is also needed to achieve this goal more smoothly. If the external feature extraction function is called f_{ex}, then we have:

$$\overline{\mathbf{C_{VI}}} = f_{ex}(\mathbf{C_{VI}}) \tag{6}$$

The vector $\mathbf{C_{VI}}$ is a 1000-dimensional vector. This vector must be transformed into smaller dimensions ($\overline{\mathbf{C_{VI}}}$). Hence, the following inequality always holds:

$$n\left(\overline{\mathbf{C_{VI}}}\right) \leq n(\mathbf{C_{VI}}) \tag{7}$$

There are different methods for dimensionality reduction. In this research, the vector quantization method (VQM) [30, 31] and the Principal Component Analysis (PCA) are used. The architecture of the VQM method is shown in Figure 4 (right). In the VQM method, the input signal $\mathbf{C_{VI}}$ is mapped to a set of predefined vectors (codes). These codes are stored in a codebook. Each code in this codebook represents a region of the input data space. In the VQM method, the codebook is a set of vectors that are typically trained through learning processes (such as the k-means algorithm) to estimate representative vectors (or codes) from the input data. The goal is to map each input data

to one of these vectors with the least error (usually the Euclidean distance). The weights in the VQM method are modeled by the following relations.

$$W_{xy}^{new} = W_{xy}^{old} + a(X - W_{xy}) \tag{8}$$

$$W_{xy}^{new} = W_{xy}^{old} - a(X - W_{xy}) \tag{9}$$

In these relations, a is the learning coefficient that indicates the update speed of the weights and X are the components of the $\mathbf{C_{VI}}$ vector. Also, n is the dimension of the Swin Transformer coder output vector, which is taken to be 1000. The purpose of these relations is to optimize the weights so that the input data is more optimally matched to the codebooks with the lowest quantization error, and thus the data (Swin Transformer coder output) is compressed better. For the VQM, the codebook was initialized using the k-means clustering algorithm applied to the extracted offline training feature vectors. We established a codebook size of 256 codewords (representing a bottleneck dimension of 256) to ensure a high compression ratio of the feature maps while keeping the quantization distortion minimal.

Despite the advantages of the VQM method such as accuracy and speed, designing a suitable codebook can be challenging and requires optimal training. Hence, a Principal Component Analysis (PCA) has also been used to extract the external feature and reduce the dimension. The architecture of the PCA for 3 features is shown in Figure 4 (left).

Unlike the VQM method, the PCA method is very fast and has a very small computational burden. PCA uses linear algebra and matrix calculations to reduce the dimensions of the data and display them in the directions where the data changes the most. PCA uses three principles based on the covariance matrix. In this method, the covariance matrix shows how correlated two features are with each other. If the covariance is positive, it means that an increase in one usually causes an increase in the other. If it is negative, it means that an increase in one causes a decrease in the other. And if it is close to zero, it means that the two features are independent of each other. The next step is to calculate the eigenvalues and eigenvectors of this matrix. The eigenvectors indicate the direction of the principal components. The eigenvalues also indicate how much of the variance each component explains in the data. Ultimately, not all principal components are equally important, and only the important ones remain. The following relationship outlines the PCA method.

$$\overline{\mathbf{C_{VI}}} = w_{pca}^{Tp}(\mathbf{C_{VI}} - \boldsymbol{\mu}) \tag{10}$$

In relation (10), w_{pca}^{Tp} represents the matrix of eigenvectors corresponding to the largest eigenvalues and $\boldsymbol{\mu}$ is the mean of the training data. For PCA-based settings, the image matrix was estimated from offline training representations and its components were reduced, while preserving the dominant variance structure of the original feature space. It should be noted that the presence of a dimensionality reduction algorithm is necessary in our method because we used the data augmentation mechanism to increase accuracy.

3.1.8. Save in Memory (GPU) After compressing and extracting the image features, a set of vectors is stored in memory:

$$\overline{\mathbf{C_{VI_1}}},\ \overline{\mathbf{C_{VI_2}}},\ \overline{\mathbf{C_{VI_3}}},\ \ldots,\ \overline{\mathbf{C_{V}}k} \tag{11}$$

These vectors are arranged in an $h * k$ matrix named Y_{ge}^{T}, where k represents the number of images and h is the dimension of the compressed vector. The Y_{ge}^{T} matrix, as an important matrix containing the compressed image information, must be placed in the UAV memory. Thus, the first phase is completed and now the UAV can fly and perform localization with three features. The first feature is a trained Swin Transformer coder. The second feature is the f_{ex} function, which can be used to extract the external feature. The third feature is Y_{ge}^{T}, which represents the robot's memory of the path.

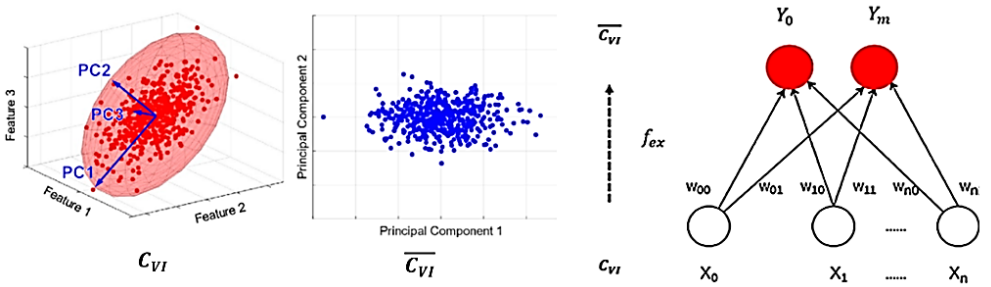

Figure 4. Top: architecture of the Vector Quantization Method (VQM). Bottom: architecture of the Principal Component Analysis (PCA)

3.2. Phase Two: Flight

In the second phase, the input image must be captured live and the robot localization completed from it. This phase is performed online and is an important phase. The details and subsections of this phase are explained below.

In the image acquisition phase, a live frame is first captured by a conventional camera; it is assumed that this camera has no special capabilities. The goal is to locate the drone by analyzing the content of this image alone. After capturing the image, it is put into the encoding pass and converted into a compressed vector. This operation is performed by the Swin Transformer model, which is trained in the offline phase and reused during flight without any changes, so that the localization process is performed with the same learned weights. Next, an external feature extraction step is applied to the compressed vector; this function should be exactly the same as the one used in the offline phase. VQM or PCA can be used for this part. Finally, the most important step of the online phase is the conversion of the feature vector into geographic coordinates. That is, the received image is entered into Swin, converted into a compressed vector, and then, through further calculations, the latitude and longitude values are extracted from it.

To minimize the time spent loading auto-encoded GE reference images, all image vectors are stacked and loaded into a $1000 * N$ dimensional matrix called Y_{ge}. Y_{ge} is indexed to include only the reference images that are 4 m ahead and behind along the path, based on the prediction of the current live image pose.

y is the compressed $1000 * 1$ vector resulting from passing the live image through the trained Swin Transformer (of course, we assume that an external feature reducer is not used in general). Kernels, such as the inner product kernel or exponential kernel, are often used to match patches between images. They provide a measure of the similarity between two patches and are calculated quickly but are not usually used to compare entire images due to the high number of pixels involved. In this new method, we use kernels on auto-encoded representations of whole images. Since these auto-encoded images are small enough to quickly compute a kernel between them, it eliminates the need to extract and match patches from an image. In this research, we improve the localization process with a kernel as another innovation. Then, with the following relation, weights are calculated with a kernel operation.

$$w = Y_{ge}^T y, \qquad y = \overline{\mathbf{C_{VI}}} \tag{12}$$

The similarity between the live image and each reference image is measured using w. With 336 images being compared, many of them have lower weights that still have an impact on the average value. These weights tend to influence the result towards the center of the reference image area. To obtain a result closer to the images with the highest weights, a new set of weights, w_{th}, is generated. All weights below one standard deviation of the maximum weight are set to zero to avoid this influence. The thresholded weights are then normalized.

$$\overline{\mathbf{w}}_{\mathbf{th}} = \frac{\mathbf{w_{th}}}{\sum_i \mathbf{w_{th},}_i} \tag{13}$$

To calculate the localization for the longitude and latitude, the thresholded weights are applied to the $2 * N$ matrix X_{ge}, where the longitude and latitude coordinates of each reference image are stored.

$$\widehat{\mathbf{X}} = \begin{bmatrix} \widehat{\mathbf{x}} \\ \widehat{\mathbf{y}} \end{bmatrix} \tag{14}$$

It is obtained by placing in the above relationship:

$$\widehat{\mathbf{X}} = \begin{bmatrix} \widehat{\mathbf{x}} \\ \widehat{\mathbf{y}} \end{bmatrix} = \mathbf{X_{ge}}\, \overline{\mathbf{w}}_{\mathbf{th}} \tag{15}$$

We calculate the covariance using the weight values as originally provided.

$$\mathbf{P} = \sum_i \mathbf{w}_i \left(\mathbf{x}_{ge,i} - \widehat{\mathbf{x}}\right) \left(\mathbf{x}_{ge,i} - \widehat{\mathbf{x}}\right)^T \tag{16}$$

After completing the earlier computation, the heading computation is performed separately. The reference image with the highest weight, y^*_{ge}, is chosen for comparison. The uncompressed live image is then rotated in 1-degree increments ranging from -5 degrees to 5 degrees. These rotated images are then encoded and arranged in a $1000 * 11$ matrix called Y_θ. The kernel computation form is repeated in this process.

$$\mathbf{w}_\theta = \mathbf{Y}_\theta^T\, \mathbf{Y}^*_{ge} \tag{17}$$

To compute the heading measurement using the normalized weights ($\overline{w}_\theta$), we follow the same procedure as in equation (5) by using a stacked vector of rotation values (X_θ^T).

$$\widehat{\theta} = \mathbf{X}_\theta^T\, \overline{\mathbf{w}}_\theta \tag{18}$$

The mean heading value, represented as $\widehat{\theta}$, is added to the heading of the chosen reference image, β, resulting in a global heading of $\widehat{\theta} + \beta$. It is worth mentioning that a similar approach can be applied to estimate altitude without the need to generate extra reference images. Therefore, the complete localization process is:

$$\widehat{\mathbf{P}} = \begin{bmatrix} \widehat{\mathbf{x}} \\ \widehat{\mathbf{y}} \\ \widehat{\theta} + \beta \end{bmatrix} \tag{19}$$

$\widehat{\mathbf{P}}$ is considered as the output that represents the localization coordinates of the UAV. In summary, in this method, first the input image from the environment is given to the Swin Transformer coder and converted into a compressed vector. Then, to calculate the latitude and longitude coordinates, all the compressed image vectors are collected in a $1000 \times N$ matrix. To compare the similarity between the live image and the reference images, internal kernels and self-encoded representations are used. The weights calculated using the kernel operation are used to simulate the similarity, and then through a thresholding of the weights so that the images with high weights have more influence, the location coordinates (latitude and longitude) and orientation are calculated. Finally, this process is completely carried out to determine the position and heading. The proposed pseudocode is given below.

4. Simulation Results

MATLAB was used for simulation. The dataset used is UTIAS. This data was collected on a 1.1 km route at the UTIAS Institute, which includes 6 flight times in different lighting conditions (sunrise, morning, noon, afternoon, evening and sunset). Satellite images of the flight path were also extracted from Google Earth and used to train the Swin Transformer and compare with real images. To evaluate the accuracy of the proposed method, RMSE criteria, execution speed and required memory size are used. The formula for RMSE is given in Equation (20). In this study, this metric calculates the location accuracy based on the difference in longitude and latitude. Also, simple time measurement in seconds is used to measure the processing speed.

Algorithm 1 Localization of Unmanned Aerial Vehicles Using Satellite Images Based on Swin Transformer

Require: $\mathbf{I_{input}}$
Ensure: $\widehat{x}, \widehat{y}, \widehat{\theta} + \beta$

1: Start: set the initial parameters. Launch the UAV.
2: Train the Swin Transformer network offline (offline phase).
3: Set the Swin Transformer hyperparameters $HP_1, HP_2, \ldots$
4: Record live image by UAV.
5: Using the Swin Transformer network, convert the input image into a compressed image vector ($\mathbf{C_{VI}}$). Then use external feature extraction to reduce dimension:
$\overline{\mathbf{C_{VI}}} = f_{ex}(\mathbf{C_{VI}})$
6: Create a $1000 * N$ dimensional matrix called Y_{ge} for reference images (fetch from memory).
7: Create a vector called y for the input live compressed image (pass from Swin Transformer):
$y = \overline{\mathbf{C_{VI}}}$
8: Calculate w_{th}.
9: Calculate $\overline{w}_{th}$ (new kernel): $\overline{w}_{th} = \dfrac{w_{th}}{\sum_i w_{th,i}}$
10: Form X_{ge} and get the vector $\widehat{\mathbf{X}}$: $\widehat{\mathbf{X}} = \begin{bmatrix} \widehat{x} \\ \widehat{y} \end{bmatrix} = X_{ge}\, \overline{w}_{th}$
11: From the previous results, calculate the covariance: $P = \sum_i w_i (x_{ge,i} - \widehat{x})(x_{ge,i} - \widehat{x})^T$
12: Get the value of w_θ: $w_\theta = Y_\theta^T y_{ge}^*$
13: Calculate the value of $\widehat{\theta}$ by forming the necessary vectors: $\widehat{\theta} = X_\theta^T \overline{w}_\theta$
14: Get the localization output: $\widehat{P} = \begin{bmatrix} \widehat{x} \\ \widehat{y} \\ \widehat{\theta} + \beta \end{bmatrix}$
15: Is there a new image captured? If yes, go to step 4; otherwise, go to step 16.
16: End.

$$\mathbf{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left((\Delta x_i)^2 + (\Delta y_i)^2 \right)} \tag{20}$$

4.1. Simulation Parameters

The simulation parameters and hyperparameters are listed in Table 1. This table also specifies the hyperparameters that have been optimized with HPO. The hyperparameters selected by the optimization mechanism are also specified in Table 1. These values were obtained using the genetic optimization algorithm and the maximum accuracy fitness function.

4.2. Results

In this section, the simulation results are presented, which are divided into three main sections. In the first section, the simulation results of the data augmentation technique are presented. In the second section, the simulation results of the Swin Transformer for localization are presented. Finally, a comparison between different methods is made.

4.2.1. Data Augmentation Technique In this section, simulation results of the data augmentation technique for two sample images are presented. The presence of this mechanism in our approach is a necessity to increase the localization accuracy. It should not be forgotten that our approach for localization relies only on images, and hence the more images are seen from more directions in the training process, the more accurately the transformer is trained.

Table 1. Simulation parameters and hyperparameters

Parameter	**Value**
Image Resolution	160×320
$\mathbf{C_{VI}}$ dimension	1000
Local window-based self-attention	7
Embedding Dimension	96
Activation functions	ReLU
Learning Rate*	0.01
Batch Size*	32
Regularization parameters*	0.001
Dropout rate*	0.1
Cache size (bottleneck layer)	256
Number of hidden layers*	128
Threshold	2 m

*Hyperparameters tuned by the HPO (genetic optimization) mechanism.

The simulation results of the data augmentation technique for the two sample images are shown in Figure 5. As is clear, 5 new images have been generated from the original image.

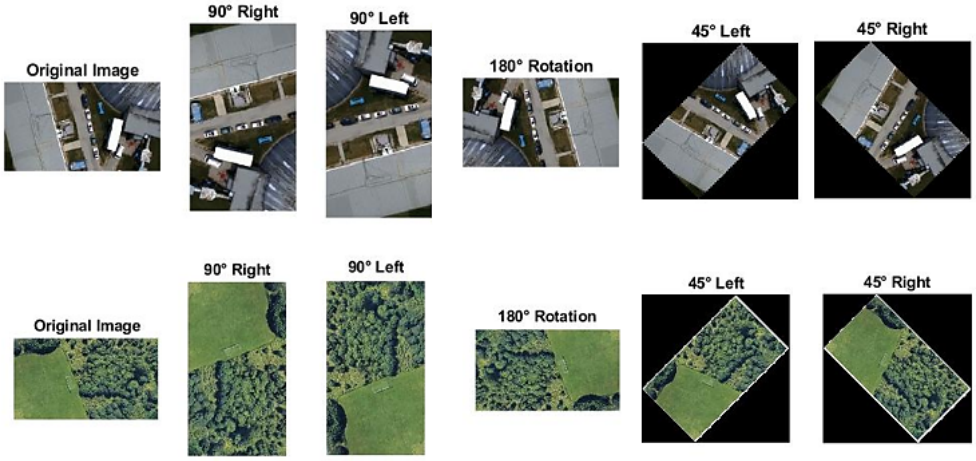

Figure 5. Simulation results of the data augmentation technique (two sample images)

In the data augmentation technique, the original image is converted into similar images with the same texture by several methods. In this research, 6 images are stored for each sampled image. These images are created by rotations of the base image (first sample and second sample). Given that the localization method of this study is image-based, it is necessary for the UAV to have a better view of the scene. Hence, the data augmentation technique helps the UAV to have a better view of the scene from different angles, and as a result, the final accuracy increases.

4.2.2. Localization Results The results of the proposed localization algorithm in different light conditions (sunrise, morning, noon, afternoon, evening and sunset) are presented in Table 2. The results presented in this table are averaged. In other words, these results have been averaged in different modes. It should be noted that these results have been obtained with optimal hyperparameters. As the results show, the error of the proposed method in longitude and latitude is 1.59 and 1.19 meters, respectively, which shows that the proposed method has good accuracy. Also, the speed of the proposed method is high, and it is real-time.

Table 2. Results of the proposed localization algorithm

Criterion	Result
x	1.59 (m)
y	1.19 (m)
Heading	0.41 (degree)
Speed (Runtime)	243 (s)
Storage (Memory)	3.11 (kB)

In Figure 6, the difference between successful localization (top) and rejected localization (bottom) is highlighted. As it is clear, if the localization coordinates obtained with the ground reality are less than a threshold, the localization is considered successful and otherwise, the localization is rejected. This threshold is considered 2 meters in this research. As shown in Figure 6, the image on the top has more details. These details have enabled the drone to perform localization operations based on them successfully. But the image on the bottom does not have any special details; the texture of the image in this case is uniform and therefore the localization is rejected. In general, most cases of localization rejection occur when the drone flies over uniform and sparsely textured environments such as open grass fields, vast waters, or highway sections that lack clear landmarks. In these areas, the representation of features extracted by the encoder lacks the high-frequency spatial variations needed to distinguish one geographic coordinate from another, resulting in flat similarity distributions, making it impossible for the drone to determine its coordinates based on this image.

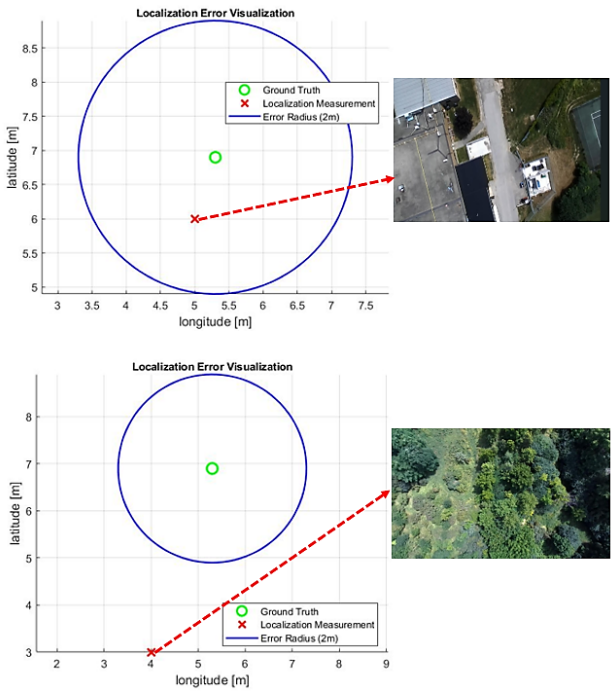

Figure 6. Difference between successful localization (top) and rejected localization (bottom)

4.2.3. Discussion and Comparison In this section, the results of the proposed method are compared with other methods. This comparison is made in three aspects: RMSE, execution speed and required memory size.

Accuracy Comparison. In Table 3, a comparison is made between the accuracy of the three localization methods. In this table, the performance of our proposed method is compared with the two methods of Patel et al. and Bianchi et al. in three indices: x (meters), y (meters) and heading angle (degrees). Our method performs best in the y coordinate with a value of 1.19 m, outperforming Patel et al. (4.50 m) and Bianchi et al. (1.39 m). In the x coordinate, our method with 1.63 m is a significant improvement over Patel et al. (3.24 m), but slightly worse than Bianchi et al. (1.54 m).

It should be noted that the Bianchi et al. [10] method does not have an external feature extraction algorithm. In the heading angle, our method with 0.41 degrees is much better than Patel et al. (3.06 degrees), but slightly behind Bianchi et al. (0.37 degrees), which can also be explained by the lack of an external feature extraction block in Bianchi et al. Overall, our method has better accuracy and speed than Patel et al. [9] and in some cases surpasses Bianchi et al., but requires minor improvements in the x coordinate and heading angle. However, our method has the highest successful localization rate. The reason for reaching this conclusion is that we have used the data augmentation technique in the proposed method. Increasing the data (creating 6 images from one image) leads to an increase in successful localization. However, to eliminate the computational burden of adding the data augmentation technique, we have also used the external feature extraction algorithm.

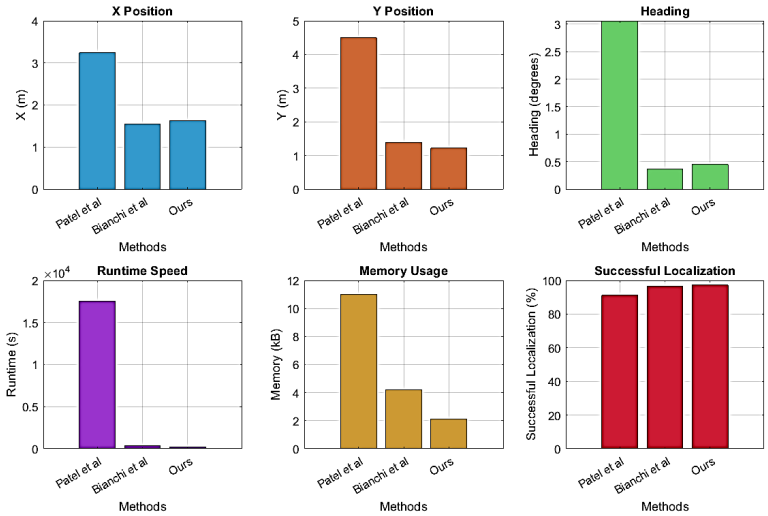

Figure 7. Comprehensive comparison between the three methods from different aspects

Table 3. Comparison of the accuracy of the proposed method with other methods

Method	x **(m)**	y **(m)**	**Heading (degree)**	**Successful localization (%)**
Patel et al. [9]	3.24	4.50	3.06	91.23
Bianchi et al. [10]	1.54	1.39	0.37	96.70
Ours	1.59	1.19	0.41	97.28

Comparison of Speed and Memory Consumption. In Table 4, the execution speed (execution time in seconds) for the three methods of Patel et al., Bianchi et al. and the proposed method of this study are compared. The Patel et al. method is very slow with a running time of 17549 seconds (about 5 hours) and is practically unusable for real-time applications. On the other hand, the Bianchi et al. method shows a significant improvement in speed with a running time of 345 seconds (about 5.8 minutes). In fact, this is the most important advantage of the Bianchi et al. method over the Patel et al. method. Our approach is also faster than both methods with a runtime of 239

seconds (about 4 minutes). This result indicates proper optimization and high efficiency of our algorithm in reducing processing time. The main factors of this superiority are the addition of the external feature extraction structure and the HPO mechanism to the basic localization algorithm. This speed improvement makes our method very suitable for practical and real-time applications, such as agricultural land monitoring or counterterrorism.

Table 4. Comparison of the speed and memory requirements of the proposed method with other methods

Method	**Speed (s)**	**Storage**
Patel et al. [9]	17549	11 kB
Bianchi et al. [10]	345	4.2 kB
Ours	239	3.11 kB

It should be noted that the reported runtime of 239 seconds corresponds to the cumulative inference and matching time over the entire 1.1-kilometer trajectory (consisting of 1200 frames). On average, the per-frame processing latency during the online in-flight phase is approximately 199 milliseconds (equivalent to 5.02 frames per second). These hardware-in-the-loop simulations were performed on a system equipped with an NVIDIA RTX 3080 GPU and an Intel i9-10900K CPU. This per-frame execution speed demonstrates that the proposed compressed pipeline can comfortably run within the control loop frequency of standard UAV flight controllers.

Also, Table 4 compares the memory requirements (storage) for the three methods of Patel et al., Bianchi et al., and the proposed method of this study. The Patel et al. method requires 11 KB of memory, while the Bianchi et al. method requires 4.2 KB of memory, showing a significant improvement in memory optimization. Our proposed method, with only 3.11 KB of memory, performs best and has significantly reduced memory consumption compared to both other methods, which is due to the presence of two feature extraction steps. This memory optimization makes our method very suitable for use in UAV systems with limited resources (such as low internal memory) and allows for the implementation of more complex algorithms in the future. Table 5 provides a more comprehensive comparison between our method and the Bianchi et al. method. The Bianchi et al. study is a very close method to our method. As shown in the table, our method has an HPO mechanism, data augmentation technique and external feature extraction structure which are not present in the Bianchi et al. study. Adding these features to our study has made our method have acceptable accuracy and at the same time be the best method among similar methods in terms of speed and memory.

Table 5. Comparison of parameters and mechanisms between Bianchi et al. and our method

Parameters and mechanisms	**Bianchi et al. [10]**	**Ours**
Image Resolution	160×320	160×320
Number of images per m	42	242
Data Augmentation	–	✓
HPO mechanism	–	✓
External feature extraction	–	✓
Practical localization	✓	✓

A comprehensive comparison between the three methods is made in Figure 7. As is clear, our proposed method performs better than the other two methods in terms of accuracy, speed and required memory. Also, the successful localization rate in our method has the highest value, and this is due to the use of the data augmentation technique.

4.2.4. Analysis of Different Mechanisms and Techniques As mentioned, our method has an HPO mechanism, data augmentation technique and external feature extraction structure which are not present in the study of Bianchi et al. Adding these features to our study has made our method have acceptable accuracy and at the same time be the best method among similar methods in terms of speed and memory. Table 6 examines the effect of the external feature extraction structure on the accuracy, speed and storage of the proposed method.

Table 6. Effect of the external feature extraction algorithm (VQM or PCA) on accuracy, speed and storage of the proposed method

Method	**RMSE (m) $\pm$ STD**	**Speed (s)**
Localization without VQM or PCA	0.96 ± 0.03	456
Localization with VQM	1.07 ± 0.05	239
Localization with PCA	1.12 ± 0.03	244

The results presented in Table 6 show that our method is improved by using the HPO mechanism, data augmentation technique and external feature extraction structure. These features are not present in the study of Bianchi et al. and make our method perform better in terms of accuracy, speed and memory consumption. As specifically presented in Table 6, the effect of two external feature extraction algorithms (VQM and PCA) is investigated. Without using VQM or PCA, the accuracy (RMSE) is 0.98 m and the runtime is 359 s; using VQM, the accuracy decreases to 1.07 m but the runtime improves to 239 s; and using PCA, the accuracy is 1.12 m and the execution time is 244 seconds. These results show that using VQM and PCA significantly reduces the execution time but has a negative impact on accuracy. In other words, by further reducing the features, some important image features for UAV localization may be lost. However, due to the use of the data augmentation technique, it is definitely necessary to have the external feature extraction structure in the proposed method (in the first phase).

In Table 7, the different modes of presence or absence of three techniques and mechanisms in the basic localization algorithm are examined. The basic localization algorithm includes a Swin Transformer [5]. This algorithm is the same method presented in Bianchi et al. (Mode 1). Also, in Mode 8, all three mechanisms and techniques are added to the basic algorithm, which is our proposed method.

Table 7. Different modes of presence or absence of three techniques and mechanisms in the basic localization algorithm (DA: Data Augmentation, HPO: Hyperparameter Optimization, EFE: External Feature Extraction, DES: Description)

Mode	**DA**	**HPO**	**EFE**	**DES**
M1	NO	NO	NO	[10]
M2	YES	NO	NO	–
M3	NO	YES	NO	–
M4	YES	YES	NO	–
M5	NO	NO	YES	–
M6	YES	NO	YES	–
M7	NO	YES	YES	–
M8	YES	YES	YES	Ours

In Table 8, the results of accuracy, speed and storage are presented for these eight modes. As it is clear, our method has the HPO mechanism, data augmentation technique and external feature extraction structure which are not present in the study of Bianchi et al. Adding these features to our study has made our method have acceptable accuracy and at the same time be the best method among similar methods in terms of speed and memory.

Table 8. Accuracy, speed and storage results for eight different modes

Mode	**RMSE (m) $\pm$ STD**	**Speed (s)**	**Storage (kB)**
M1	1.06 ± 07	411	5.3
M2	$\mathbf{0.82} \pm 04$	455	6.75
M3	0.99 ± 04	425	5.99
M4	0.93 ± 02	401	5.99
M5	1.52 ± 11	**238**	**4.10**
M6	1.01 ± 03	251	4.21
M7	1.12 ± 03	244	7.81
M8	1.07 ± 05	239	3.11

The graphs in Figure 8 show the results of accuracy, speed, and storage for eight different modes. As is clear, the best case requires a compromise between accuracy, speed, and storage, which is provided in Mode 8.

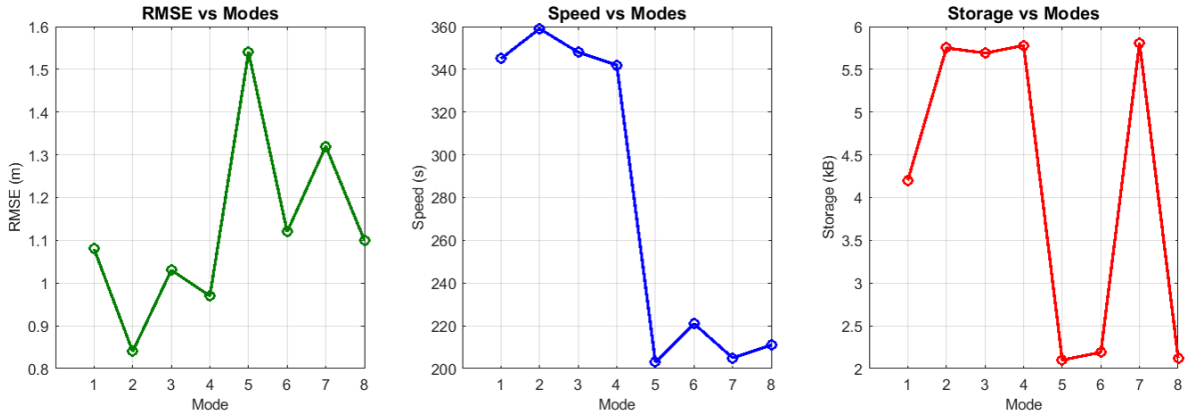

Figure 8. Changes in accuracy, speed, and storage for eight different modes

The comparison across the eight operational modes highlights the synergistic relationship between the framework's components. Implementing data augmentation (Mode 2) yields a noticeable reduction in RMSE compared to the baseline (Mode 1), verifying that explicitly training the network on multiple rotations helps the Swin Transformer construct rotation-invariant representations. When hyperparameter optimization (HPO) is introduced (Mode 3 and Mode 4), the model converges to a more robust parameter set, mitigating the overfitting risks associated with augmented datasets. Finally, incorporating external feature extraction (PCA or VQM) introduces a slight trade-off: it compresses the descriptors to a fraction of their original size, which marginally increases the RMSE but significantly reduces the in-flight matching latency, making Mode 8 the most balanced configuration for deployment. For better visual comparison, the heatmap of accuracy, speed and storage results for eight different modes is also shown in Figure 9. It is clearly evident that the choice of mode has a significant impact on the final result.

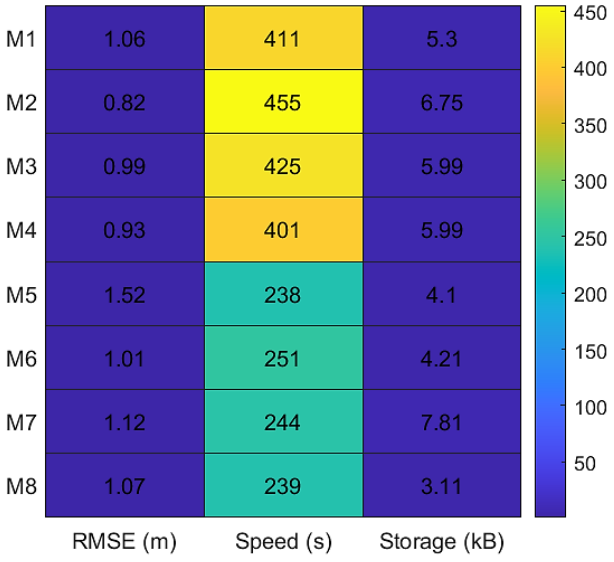

Figure 9. Heatmap of accuracy, speed and storage results for eight different modes

It should be noted that environmental variations such as changing illumination, seasonal dynamics, and transient obstacles present major challenges to vision-based localization. The self-attention mechanism of the

Swin Transformer helps mitigate these issues by focusing on structural and contextual features rather than local pixel intensities. Furthermore, to address the viewpoint discrepancy between top-down satellite imagery and UAV camera captures, the system assumes that the UAV camera is stabilized using a gimbal to maintain a downward-facing (nadir) perspective. This geometric alignment minimizes perspective distortion, allowing the model to perform matches based primarily on translation and rotation differences. While the UTIAS dataset provides a highly controlled environment to validate the core framework, the 1.1-kilometer trajectory represents a limited testing scope. As the operational route scales to longer distances, the size of the pre-built reference database grows linearly, which increases the search space for the in-flight matching phase. To deploy this system over longer missions or more diverse environments (e.g., urban or forested terrains), hierarchical indexing or spatial partitioning of the database would be required to maintain real-time performance. Evaluating the framework on larger, heterogeneous datasets remains an important direction for our future research.

5. Conclusion

In this paper, a new framework for accurate positioning of UAVs without GPS was presented. The method was based on image alignment and the powerful Swin Transformer network. With the help of this architecture and the application of hyperparameter optimization methods, the model was able to achieve an accuracy of about 1.07 m RMSE while maintaining a processing time suitable for real-time applications. The use of data augmentation and a lightweight PCA-based external feature extraction step also played an important role in reducing the computational burden and increasing the stability of the model and showed that the combination of Swin with an effective dimensionality reduction can be a reliable alternative to GPS systems in noisy and disturbed environments. Despite the acceptable performance of the proposed localization algorithm, the limited dataset and lack of samples from some environments, such as certain complex urban spaces, have led to some of the model's behavior being less studied in these areas. Also, the comprehensive hyperparameter tuning process and the limitations associated with the complexity of the feature extraction algorithms are among the main challenges of this research. Future research could exploit emerging technologies such as event cameras that are robust to rapid lighting changes and could provide new avenues for increasing accuracy and speed of positioning. The study of geomagnetic navigation algorithms is also an attractive area [32]. In addition, the use of Domain Adaptation concepts could enable the transfer of the Swin Transformer-based model to new environments without the need for extensive labeled data. These directions could pave the way for the development of the next generation of UAV image positioning methods.

REFERENCES

1. B. He, X. Ji, G. Li, and B. Cheng, "Key technologies and applications of UAVs in underground space: a review," *IEEE Transactions on Cognitive Communications and Networking*, vol. 10, no. 3, pp. 1026–1049, 2024.
2. Z. Fang and A. V. Savkin, "Strategies for optimized uav surveillance in various tasks and scenarios: A review," *Drones*, vol. 8, no. 5, p. 193, 2024.
3. P. Seralathan and S. Edward, "Revolutionizing Agriculture: Deep Learning-Based Crop Monitoring Using UAV Imagery-A Review," in *2024 International Conference on Expert Clouds and Applications (ICOECA)*, 2024: IEEE, pp. 777–784.
4. R. Mahmoud, H. ElKadi, A. Abdelhafeez, and D. ElMenshawy, "Pest Detection and Classification Approach based on YOLOv11-S and Real-ESRGAN," *Statistics, Optimization & Information Computing*, 2026.
5. N. Abbas, Z. Abbas, X. Liu, S. S. Khan, E. D. Foster, and S. Larkin, "A survey: Future smart cities based on advance control of Unmanned Aerial Vehicles (UAVs)," *Applied Sciences*, vol. 13, no. 17, p. 9881, 2023.
6. S. A. H. Mohsan, N. Q. H. Othman, Y. Li, M. H. Alsharif, and M. A. Khan, "Unmanned aerial vehicles (UAVs): Practical aspects, applications, open challenges, security issues, and future trends," *Intelligent Service Robotics*, vol. 16, no. 1, pp. 109–137, 2023.
7. R. Shokirov, N. Abdujabarov, T. Jonibek, K. Saytov, and S. Bobomurodov, "Prospects of the development of unmanned aerial vehicles (UAVs)," *Technical Science and Innovation*, vol. 2020, no. 3, pp. 4–8, 2020.
8. J. Agrawal and M. Y. Arafat, "Transforming farming: A review of AI-powered UAV technologies in precision agriculture," *Drones*, vol. 8, no. 11, p. 664, 2024.
9. B. Patel, T. D. Barfoot, and A. P. Schoellig, "Visual localization with google earth images for robust global pose estimation of uavs," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020: IEEE, pp. 6491–6497.
10. M. Bianchi and T. D. Barfoot, "UAV localization using autoencoded satellite images," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1761–1768, 2021.

11. G. Conte and P. Doherty, "An integrated UAV navigation system based on aerial image matching," in *2008 IEEE Aerospace Conference*, 2008: IEEE, pp. 1–10.
12. A. Nassar, K. Amer, R. ElHakim, and M. ElHelw, "A deep CNN-based framework for enhanced aerial imagery registration with applications to UAV geolocalization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1513–1523.
13. A. L. Majdik, D. Verda, Y. Albers-Schoenberg, and D. Scaramuzza, "Air-ground matching: Appearance-based GPS-denied urban localization of micro aerial vehicles," *Journal of Field Robotics*, vol. 32, no. 7, pp. 1015–1039, 2015.
14. C. Linegar, W. Churchill, and P. Newman, "Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016: IEEE, pp. 787–794.
15. D. Dai, Z. Lu, X. Dai, and Q. Quan, "RflyPano: A Panoramic Benchmark for Ultra-low Altitude UAV Localization Powered by RflySim," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026, vol. 40, no. 22, pp. 18216–18224.
16. T. Liu, K. Ren, and Q. Chen, "DiffusionUavLoc: Visually Prompted Diffusion for Cross-View UAV Localization," *IEEE Internet of Things Journal*, 2026.
17. A. M. Eltaher, A. Maher, M. A. Abozied, and A. Elrwainy, "Unpacking ORB-SLAM for UAV Navigation and Evaluating Its Efficiency Across Diverse Environments: A Systematic Review," *Statistics, Optimization & Information Computing*, vol. 15, no. 6, pp. 5359–5388, 2026.
18. M. Gridseth and T. D. Barfoot, "Deepmel: Compiling visual multi-experience localization into a deep neural network," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020: IEEE, pp. 1674–1681.
19. I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu, "Relative camera pose estimation using convolutional neural networks," in *International Conference on Advanced Concepts for Intelligent Vision Systems*, 2017: Springer, pp. 675–687.
20. X. Hou, L. Shen, K. Sun, and G. Qiu, "Deep feature consistent variational autoencoder," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017: IEEE, pp. 1133–1141.
21. J. Snell, K. Ridgeway, R. Liao, B. D. Roads, M. C. Mozer, and R. S. Zemel, "Learning to generate images with perceptual similarity metrics," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017: IEEE, pp. 4277–4281.
22. A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.
23. X. Chen *et al.*, "Variational lossy autoencoder," *arXiv preprint arXiv:1611.02731*, 2016.
24. M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 699–715.
25. Z. Fouad, M. Alfonse, M. Roushdy, and A.-B. M. Salem, "Hyper-parameter optimization of convolutional neural network based on particle swarm optimization algorithm," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 6, pp. 3377–3384, 2021.
26. R. de Sousa Maximiano, V. A. de Santiago Júnior, and E. H. Shiguemori, "On the benefits of automated tuning of hyper-parameters: an experiment related to temperature prediction on UAV computers," in *Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*, 2022: SBC, pp. 509–520.
27. H. Bertrand, "Hyper-parameter optimization in deep learning and transfer learning: applications to medical imaging," Université Paris Saclay (COmUE), 2019.
28. P. Kumar, S. Batra, and B. Raman, "Deep neural network hyper-parameter tuning through twofold genetic approach," *Soft Computing*, vol. 25, no. 13, pp. 8747–8771, 2021.
29. G. Afifi and Y. Gadallah, "Autonomous 3-D UAV localization using cellular networks: Deep supervised learning versus reinforcement learning approaches," *IEEE Access*, vol. 9, pp. 155234–155248, 2021.
30. R.-C. Zheng, H.-P. Du, X.-H. Jiang, Y. Ai, and Z.-H. Ling, "ERVQ: Enhanced residual vector quantization with intra-and-inter-codebook optimization for neural audio codecs," *IEEE Transactions on Audio, Speech and Language Processing*, 2025.
31. I. Jarraya, F. Alahmed, M. Abdelkader, K. Gabr, M. B. Kadri, and W. Boulila, "CLAK: CNN-LSTM-Attention and Kolmogorov-Arnold Networks for non-visual UAV localization in GNSS-denied environments," *Satellite Navigation*, vol. 7, no. 1, p. 6, 2026.
32. A. M. F. Alfawaz, M. T. A. Zyadeh, and I. S. Fathi, "Advancing Agricultural Yield Forecasting through Geomagnetic Navigation Algorithm-Optimized Machine Learning: for Accurate Multi-Crop Agricultural Yield Prediction," *Statistics, Optimization & Information Computing*, vol. 16, no. 1, pp. 808–830, 2026.