Article

AGCosPlace: A UAV Visual Positioning Algorithm Based on Transformer

Ya Guo, Yatong Zhou * and Fan Yang

School of Electronic and Information Engineering, Hebei University of Technology, 5340 Xiping Road, Beichen District, Tianjin 300401, China; 202011901005@stu.hebut.edu.cn (Y.G.)
* Correspondence: zyt@hebut.edu.cn

Abstract: To address the limitation and obtain the position of the drone even when the relative poses and intrinsics of the drone camera are unknown, a visual positioning algorithm based on image retrieval called AGCosPlace, which leverages the Transformer architecture to achieve improved performance, is proposed. Our approach involves subjecting the feature map of the backbone to an encoding operation that incorporates attention mechanisms, multi-layer perceptron coding, and a graph network module. This encoding operation allows for better aggregation of the context information present in the image. Subsequently, the aggregation module with dynamic adaptive pooling produces a descriptor with an appropriate dimensionality, which is then passed into the classifier to recognize the position. Considering the complexity associated with labeling visual positioning labels for UAV images, the visual positioning network is trained using the publicly available Google Street View SF-XL dataset. The performance of the trained network model on a custom UAV perspective test set is evaluated. The experimental results demonstrate that our proposed algorithm, which improves upon the ResNet backbone networks on the SF-XL test set, exhibits excellent performance on the UAV test set. The algorithm achieves notable improvements in the four evaluation metrics: R@1, R@5, R@10, and R@20. These results confirm that the trained visual positioning network can effectively be employed in UAV visual positioning tasks.

Keywords: UAV visual navigation; visual positioning; graph network; transformer

check for updates

Citation: Guo, Y.; Zhou, Y.; Yang, F. AGCosPlace: A UAV Visual Positioning Algorithm Based on Transformer. *Drones* **2023**, *7*, 498. https://doi.org/10.3390/drones7080498

Academic Editors: Kate Saenko, Kimhung Pho and Jie Yuan

Received: 26 June 2023
Revised: 22 July 2023
Accepted: 26 July 2023
Published: 28 July 2023

1. Introduction

With the rapid advancement of unmanned aerial vehicle (UAV) technology, UAVs are finding increasingly diverse applications in military, civil, and scientific research fields. They are employed for tasks such as aerial photography, logistics transportation, disaster investigation, and agricultural monitoring [1–5]. In order to successfully accomplish these tasks, UAVs require autonomous navigation and obstacle avoidance capabilities. UAVs typically acquire their own states and gather information about their surroundings through a combination of exteroceptive and proprioceptive sensors. The conventional sensor suite employed for navigation primarily includes global positioning systems (GPS), axis accelerometers, gyroscopes, and inertial navigation systems (INS) to obtain positioning information [6,7]. However, UAVs relying solely on external sensors such as GPS often encounter several challenges [7–10]. Some of the challenges include: (1) GPS signals have inherent inaccuracies, resulting in limited position accuracy for UAVs. The precision of GPS readings can vary depending on factors such as signal interference, atmospheric conditions, and line-of-sight obstructions. This imprecision can affect the accuracy of UAV navigation and pose challenges in applications that require precise positioning; (2) in certain scenarios, such as operating in urban environments or flying at low altitudes, GPS signals may be obstructed or weakened. Tall buildings, dense foliage, and other obstructions can cause signal loss or multipath interference, leading to degraded GPS performance or even complete signal dropout. This limitation hinders the reliability and robustness of UAV navigation systems; (3) GPS sensors provide position and velocity information but do not offer direct information about the UAV's immediate surroundings. These challenges primarily arise from the limitations of GPS technology and the reliance on signals from satellites. UAVs relying solely on GPS may lack environmental awareness, making them vulnerable to collisions with obstacles, an inability to navigate complex terrain, or difficulty avoiding dynamic obstacles such as moving vehicles or pedestrians. To overcome these challenges, visual navigation techniques can be employed in conjunction with or as an alternative to GPS-based navigation [7,8,11–13]. Visual navigation leverages onboard cameras and computer vision algorithms to extract information from the environment and enable UAVs to perceive and navigate their surroundings. If GPS signals become unreliable or unavailable, UAVs can rely on visual information for localization and navigation, ensuring continuous operation and resilience to GPS signal issues. Visual navigation allows UAVs to perceive their position and orientation relative to the environment. By analyzing visual features, landmarks, or patterns, UAVs can estimate their position more accurately and precisely than GPS alone. In search and rescue missions, time is critical, and UAVs equipped with visual navigation capabilities can play a crucial role. Visual navigation allows UAVs to autonomously navigate through challenging environments, such as disaster-stricken areas, forests, or mountainous terrain. They can detect and avoid obstacles, locate survivors, and transmit real-time images and data to ground stations. Visual navigation enables quick and effective search operations, improving the chances of finding and rescuing people in distress. In surveillance and monitoring missions, UAVs equipped with visual navigation are invaluable tools for surveillance and monitoring tasks. In law enforcement, border patrol, or security applications, UAVs can autonomously patrol designated areas, identify suspicious activities, and track targets. Visual navigation enables precise tracking and object recognition, enhancing the effectiveness of surveillance operations and improving situational awareness. In environmental monitoring missions, for environmental monitoring tasks, such as tracking wildlife, assessing vegetation health, or monitoring pollution, UAVs with visual navigation capabilities offer significant advantages. Visual navigation allows UAVs to fly close to the ground or follow specific flight paths to capture high-resolution images and data. With accurate visual localization, UAVs can revisit specific locations, enabling longitudinal studies and contributing to environmental research and conservation efforts.

In UAV visual navigation, the UAV compares the current image with previously recorded landmarks or maps using image matching technology to determine its position and attitude, commonly referred to as pose estimation. This process enables the visual positioning of the UAV, which is crucial for achieving autonomous flight and mission execution. However, current visual positioning algorithms that rely on image matching have certain shortcomings that can limit their effectiveness in certain scenarios [14–17]. These shortcomings include: (1) Image matching algorithms often rely on comparing feature descriptors extracted from images. However, these algorithms can be sensitive to changes in lighting conditions or variations in viewpoint; (2) Image matching algorithms can struggle with handling changes in the environment. For example, alterations in scene geometry, object occlusions, or variations in environmental conditions (e.g., different seasons or weather conditions) can make it difficult to find consistent matches between images. The algorithms may not effectively handle variations in scale, rotation, or object appearance, leading to decreased accuracy and reliability; (3) Image matching algorithms typically involve performing extensive feature extraction, matching, and geometric verification steps. These processes can be computationally expensive, especially when dealing with large-scale datasets or real-time applications. The high computational complexity can limit the real-time performance and efficiency of visual positioning systems. To address these shortcomings, an alternative technique called image retrieval can be used for visual positioning. Image retrieval focuses on finding semantically similar images from a database without relying on precise feature matching. Instead of relying on local feature descriptors, image retrieval techniques leverage global image representations and content-based indexing. Compared with image matching algorithms, image retrieval methods can offer lower computational complexity. By employing efficient indexing and retrieval techniques, such as hashing or approximate nearest neighbor search, image retrieval algorithms can provide faster and more scalable solutions, making them suitable for real-time applications and large-scale datasets. In addition, the premise of the visual positioning algorithm based on image matching is that to realize the positioning function, it needs to know the relative pose and intrinsics of the camera, while the visual positioning algorithm based on image retrieval needs fewer parameters to realize the positioning function, such as the latitude and longitude coordinates and heading of the image. Thus, when the existing visual positioning algorithms based on UAV image matching cannot determine the UAV's position without knowledge of the camera's relative poses and intrinsics [18], a visual positioning algorithm based on UAV image retrieval is necessary to provide reliable positioning information.

In recent years, computer vision technology has made significant progress, with deep learning and neural networks playing a vital role in supporting UAV visual navigation. Notably, the visual positioning network CosPlace, trained on the extensive Google Street View dataset, has garnered considerable attention due to its simple architecture, efficient memory usage during training, and strong generalization capabilities [19]. To address the limitations of the unknown relative poses and intrinsics image matching algorithm for UAV visual positioning, this study investigates the visual positioning network CosPlace based on image retrieval. Additionally, a Transformer-based visual positioning network with improved performance, named AGCosPlace, is proposed. The AGCosPlace algorithm is an enhanced version of the CosPlace algorithm. To address low recalls in the transformer-based CosPlace algorithm, an attention-based single-layer graph network coding module is introduced to the backend of the CosPlace feature extraction backbone network, named AGCosPlace. In the attention-based graph network coding module, the feature map obtained from the feature backbone network undergoes a self-attention operation, which expands the extraction range of feature points. The resulting feature map, after the attention operation, is then fused with the original feature map to create a fusion feature map that contains more comprehensive information. Next, the fused feature map is subjected to multi-layer perceptron (MLP) encoding, enabling the extraction of more abstract features. These features, obtained through MLP coding, are further fused with the original feature map for the second time. This fusion process enhances the aggregation of contextual information. The descriptor by this enrichment process becomes more robust for the final classifier, thereby improving its positioning ability. AGCosPlace enhances the CosPlace algorithm by incorporating a graph coding network that integrates an attention module, MLP modules, and a graph neural network (GNN) at the backend of the feature extraction network. In addition, the trained network model is evaluated on a custom UAV dataset specifically created for UAV positioning. This dataset enables the assessment of the performance of the visual positioning algorithm. The primary contributions of this paper are outlined as follows:

(1) Designing a module that integrates multi-head self-attention, MLP, and graph neural networks into the backend of the visual positioning algorithm's backbone network. This module aims to enhance the aggregation of contextual information from the feature map. It addresses the limitation of the poor recall effect observed in Transformer-based research of the CosPlace network.

(2) Investigating the impact of the sinusoidal position coding module on the proposed visual localization algorithm. This exploration sheds light on how the introduction of the sinusoidal position coding module influences the performance of the visual positioning algorithm.

(3) Constructing a UAV dataset that closely resembles aerial perspectives and annotating it with UTM coordinates. This dataset serves as a means to test the performance of the trained visual positioning algorithm. It addresses the challenge posed by the image matching algorithm's inability to achieve UAV positioning when the camera's relative poses and intrinsics are unknown.

2. Related Works

One essential aspect of UAV communication security is the vulnerability of drones to cyberattacks. Drones, like any other connected device, can be susceptible to hacking attempts and unauthorized access. The consequences of compromised UAV communications can be severe, including potential data breaches, loss of control over the drone, or even malicious takeovers [20,21]. To address these security concerns and enhance the reliability of drone operations, visual navigation plays a significant role. Visual navigation allows UAVs to reduce their reliance on external communication links, such as GPS or radio frequency (RF) communication, for position estimation and control. By leveraging onboard cameras and computer vision algorithms, visual navigation enables UAVs to navigate autonomously, perform obstacle avoidance, and maintain their flight trajectory without constant reliance on external communications. The significance of visual navigation in the context of UAV communication security lies in its ability to provide a more robust and secure navigation alternative. By reducing the dependence on external communication links, visual navigation mitigates the risk of signal jamming, spoofing, or other cyberattacks that could compromise the UAV's control and mission objectives.

UAV visual navigation involves utilizing the visual sensor onboard the UAV to capture image information from the environment. This information is then processed and analyzed using computer vision technology and image processing algorithms. The objective is to achieve autonomous navigation, position estimation, path planning, and obstacle avoidance for the UAV during flight [22,23]. Image matching serves as a fundamental approach in UAV visual navigation, where the current image is compared with pre-recorded landmarks or maps to determine the UAV's position and attitude, known as pose estimation, enabling visual positioning of the UAV.

Consequently, research on visual positioning algorithms based on UAV image matching has gained significant attention. Wan et al. [24] proposed an illumination-invariant UAV image matching algorithm. By employing the phase correlation image matching algorithm, UAV images captured under different illumination conditions and reference satellite images with geographic labels were used for autonomous positioning of the UAV, thereby facilitating effective navigation. Zhang et al. [25] introduced a deep learning-based local feature matching algorithm for UAV navigation using infrared and reference satellite image matching. Deep network models were utilized to extract corresponding features from satellite images and UAV infrared images, addressing the challenges posed by different imaging modalities. The iterative training method was employed to overcome the sparsity and labeling complexity issues associated with current public datasets. To address the challenges posed by significant visual content differences between UAV and satellite images, Kan et al. [26] proposed a quality-aware template matching method based on scale-adaptive deep convolution features. This method selects appropriate template sizes for UAV images and satellite images, and the obtained feature maps undergo similarity measurement to generate a matching probability map. This enables the selection of the best match and achieves target positioning. Similarly, Ding et al. [27] tackled the matching of UAV and satellite images from different perspectives. They simplified the retrieval problem by treating it as a classification problem, addressing the imbalance in the number of input samples between UAV images and satellite images. A UAV image and satellite image matching algorithm based on UAV geolocation was proposed, facilitating two-way matching for UAV image positioning and navigation tasks.

However, the positioning algorithm based on matching UAV images with satellite images faces several challenges, including limited memory on the UAV's onboard computer and significant viewpoint differences between the pre-stored satellite images and real-time images captured by the UAV [24–28]. Therefore, it is crucial to explore positioning algorithms based solely on UAV images. This approach involves storing ground feature-rich images captured by the UAV sensor in the onboard computer and matching them with real-time UAV images to achieve UAV positioning. However, there are limited algorithms available for pure UAV image matching, and existing UAV datasets primarily focus on target tracking and detection [29–34]. Additionally, image matching algorithms require camera relative poses and intrinsics to realize visual positioning [18].

The visual positioning algorithm based on image retrieval estimates the position of a query image by querying a database with geographical markers to find the most visually similar position. This is achieved with only the latitude, longitude, and heading information of the captured image, without needing knowledge of the camera's relative poses and intrinsics. Local invariant features, such as SIFT [35], are used for each database image, and they are aggregated into a single vector, known as the descriptor, to represent the entire image. With the advancements in deep learning, Chen et al. [36] proposed an auxiliary indoor positioning algorithm based on CNN image retrieval. The algorithm employs a pre-trained deep convolutional network to extract image features for similarity comparison. It outputs images that match the target image and calculates precise positioning results for pose estimation. However, the algorithm relies solely on the pre-trained network for experiments and only tests on the trajectory test set, leaving the precision and recall indicators of the model unknown. Similarly, Ha et al. [37] used a pre-trained VGG network for feature extraction and similarity evaluation of 143 images of actual buildings. Unfortunately, this approach also suffers from the same drawbacks as the previous algorithm. On the other hand, Arandjelov et al. [38] designed an end-to-end architecture based on convolutional neural networks that can be directly used for location recognition tasks. However, this algorithm has a high memory requirement for visual geolocation systems due to the output of high-dimensional descriptors. In addition, state-of-the-art visual geolocation algorithms often rely on contrast learning methods [38–40]. However, these algorithms can be computationally expensive and may struggle to handle large databases.

To address these limitations, Berton et al. [19] introduced a novel method called CosPlace, which offers a promising solution for visual positioning tasks. Its effective training on large-scale datasets, simple architecture, strong generalization capabilities, and reduced GPU memory requirements make it a valuable tool for a wide range of applications that demand accurate and efficient visual positioning. Hence, applying the CosPlace algorithm to the UAV visual positioning task is considered. However, the CosPlace algorithm combined with the transformer has lower recalls. Inspired by the SuperGlue algorithm [18], the utilization of the graph network in image matching tasks has demonstrated robust descriptors and image matching performance in indoor and outdoor scenes through the incorporation of the transformer module. Thus, in the proposed AGCosPlace, an attention-based single-layer graph network coding module is introduced to the backend of the CosPlace feature extraction backbone network. In the attention-based graph network coding module, a self-attention operation expands the extraction range of feature points and allows for a broader context to be considered during feature extraction. One fusion operation creates a fusion feature map that contains more comprehensive information. An MLP encoding enables the extraction of more abstract features. Secondary fusion operations enhance the aggregation of contextual information. The feature map output by the backbone network undergoes further encoding to enrich the features introduced to the aggregation layer. This enrichment process enhances the descriptor available for the final classifier, thereby improving its positioning ability. Given the strong generalization performance of AGCosPlace, the network on the publicly available SF-XL dataset is trained, a dedicated UAV image test set is constructed, and the performance of the trained network on this test set is evaluated.

3. AGCosPlace

A visual positioning network called AGCosPlace is proposed to address the challenge of locating UAVs when the camera's relative poses and intrinsics are unknown in image registration algorithms. In the AGCosPlace network, the input image undergoes preprocessing before being passed through the backbone network for feature extraction, resulting in a feature map. To overcome the recall limitations of the Transformer-based CosPlace, a single-layer GNN module inspired by SuperGlue [18] is designed in AGCosPlace. This GNN module incorporates multi-head self-attention and MLP operations to further encode the feature map, expanding the extraction range of network feature points and facilitating the aggregation of feature point context. Subsequently, the descriptor is obtained using dynamically adaptive pooling with GeM and a fully connected layer module with a 512-dimensional output.

The obtained descriptor, along with the category information, is fed into the classifier to generate the final classification result. The cross-entropy loss function is employed to calculate the loss between the classification result and the category, and an iteration is performed during the training process to optimize the loss function. During the evaluation, query descriptors obtained through the aggregation layer perform a strong L2 distance search in database descriptors obtained through the aggregation layer to obtain predicted results. For real label data, real query labels perform K-nearest neighbors (KNN) searches on real database labels to obtain real query results. Finally, it is judged whether the predicted result exists in real query results to obtain the discriminant result. Various search methods are employed to predict descriptors and labels due to their distinct representations and task characteristics. Utilizing L2 distance for searching similar descriptors effectively identifies database descriptors that align with the query descriptors, resulting in more accurate descriptor predictions. On the other hand, the KNN search for labels allows the algorithm to determine the final label prediction through a voting mechanism involving multiple closest database labels, thus enhancing the stability and robustness of label predictions. By employing different search methods, the AGCosPlace algorithm can comprehensively leverage the information within the database, consequently enhancing the algorithm's accuracy and performance. The architecture of the AGCosPlace network is illustrated in Figure 1.

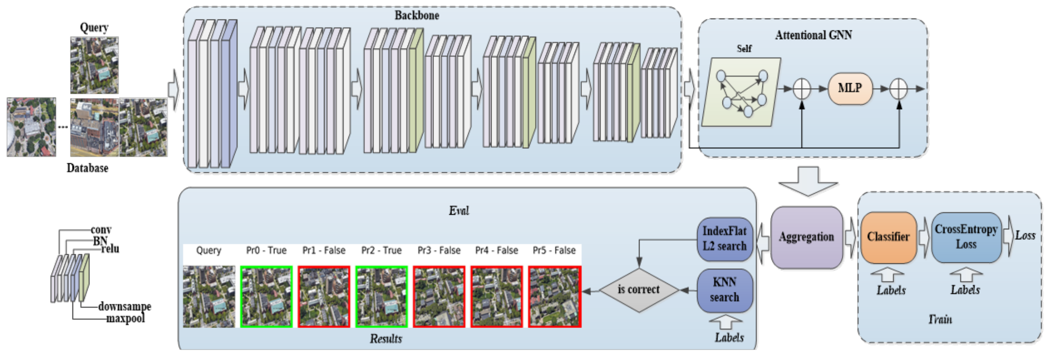


Figure 1. The architecture of the AGCosPlace network.

3.1. Backbone

The purpose of this module is to extract the features from the input image, capturing its high-level semantic information. Let's assume that the input image is denoted as I. Prior to feature extraction, the image undergoes various preprocessing steps, including data augmentation, cropping, and normalization, resulting in the transformed image being denoted as I'. These preprocessing techniques are applied to facilitate training and prepare it for subsequent feature extraction processes.

$$I' = Pre(I), \tag{1}$$

The image after data enhancement is shown in Figure 2. The preprocessing operations applied to image I, are represented by Pre. Figure 2 illustrates the images after undergoing data enhancement, showcasing the enhanced features and characteristics resulting from the preprocessing steps.

The transformed image I' is then fed into the backbone network for feature extraction. In this case, the ResNet18 architecture is employed as the backbone network. ResNet18 consists of five convolutional group modules, as illustrated in Figure 3. By utilizing the ResNet backbone, the AGCosPlace network can effectively extract discriminative features that contribute to accurate visual positioning and localization tasks.

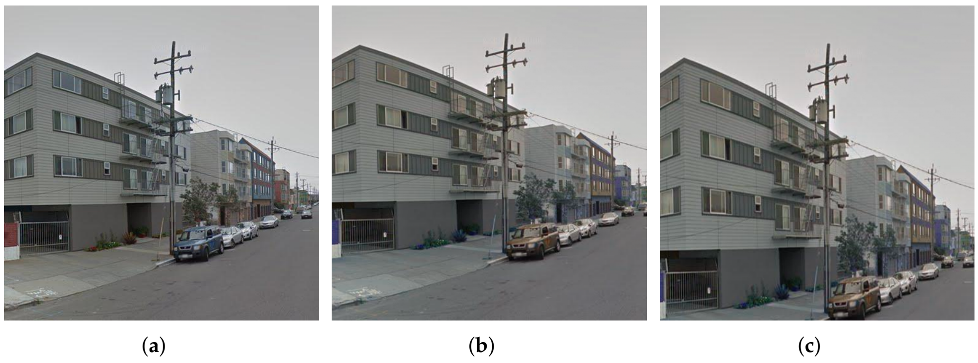


Figure 2. Data enhancement examples. (**a**) Original image (**b**) Data enhancement image 1 (**c**) Data enhancement image 2.

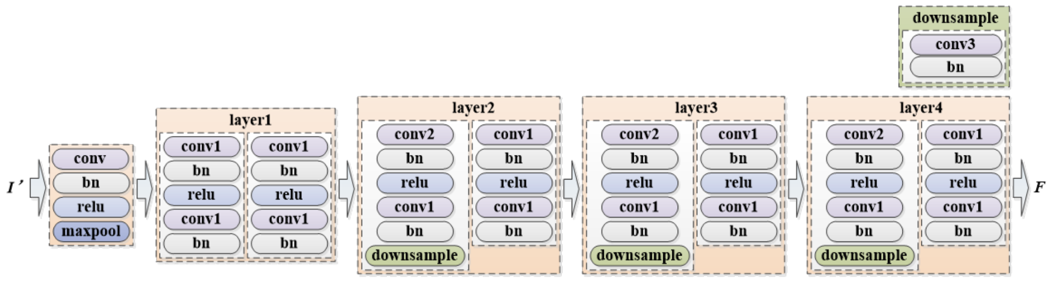


Figure 3. Backbone network.

In each of the five modules, several operations are performed to process the input image I'. These operations include 2D convolution (conv) with a kernel size of 7, a stride of 2, and a padding of 3; Batch Normalization (bn); max pooling (maxpool) with a kernel size of 3, a stride of 2, a padding of 1, and a dilation of 1; 2D convolution (conv1) with a kernel size of 3, a stride of 1, and a padding of 1; 2D convolution (conv2) with a kernel size of 3, a stride of 2, and a padding of 1; and a downsample module composed of a 2D convolution (conv3) operation with a kernel size of 1, a stride of 2, and Batch Normalization (bn). Additionally, the rectified linear unit (relu) activation function is applied within the module.

To represent the feature extraction process of ResNet18 on the transformed image I', the symbol f is used. Consequently, the resulting feature map is denoted as F. This feature map F contains the extracted features that encode important information from the input image, which will be utilized for subsequent steps in the AGCosPlace network.

$$F = f(I'), \tag{2}$$

3.2. Attentional GNN

To enhance the performance of the Transformer-based visual positioning algorithm, the feature map F is introduced into a single-layer GNN with multi-head self-attention and MLP. This allows us to further encode the feature map and address the issue of the poor recall effect observed in the Transformer-based CosPlace algorithm. Our goal is to expand the range of network feature points and aggregate their features effectively.

In the proposed module, a graph network, a self-attention mechanism, and MLP coding operations work together to expand the extraction range of network feature points and aggregate their contextual information. The module follows a specific network structure, as depicted in Figure 4. It begins with the feature map F, which is encoded using a single-layer graph neural network for attention coding. The resulting feature map is then combined with the original feature map through a skip connection to obtain a fused feature map. This fused feature map undergoes further MLP coding and skips connections to generate a secondary fused feature map. By incorporating these operations and connections, the encoding and aggregation of feature point information are enhanced, ultimately improving the recall effect of the Transformer-based CosPlace algorithm.

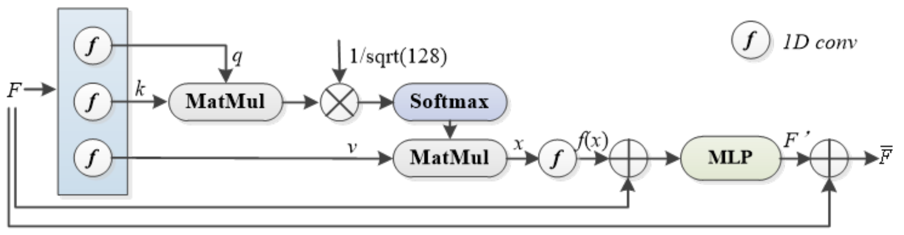


Figure 4. Attentional GNN module network structure.

In the Attentional GNN network module, three 1D convolution operations on the feature map F are performed. Each convolution operation has an input channel and output channel of 512, a convolution kernel size of 1, and a step size of 1. These operations result in three transformed feature maps, namely q, k, and v. Next, self-attention is applied to the transformed feature maps. Self-attention involves computing the attention weights between different positions within each feature map. Specifically, calculate the attention scores by taking the dot product between q and k. The self-attention mechanism can be mathematically represented as follows:

$$x = v^T softmax(\frac{q^T k}{\sqrt{128}}), \tag{3}$$

q and k represent the queries and keys, respectively. The softmax function is applied to normalize the attention scores. Additionally, 128 represents the dimension of the keys. By computing the attention scores, a weighted sum of the values (v) based on the attention weights is obtained. This allows the network to focus on important features and capture contextual information. The output of the self-attention operation is a refined representation of the feature map F, which incorporates both local and global dependencies. It is worth noting that the self-attention operation is performed across multiple heads (4 heads) to capture different relationships and enhance the overall representation power of the network.

In the attention-based GNN network module, after the self-attention operation, the resulting tensor x is further processed using a 1D convolution operation. This operation has an input channel of 512 and an output channel of 512. The convolution kernel size is 1, and the step size is also set to 1. This convolutional operation is applied to refine and compress the information in x, resulting in a tensor denoted as $f(x)$.

After obtaining $f(x)$, perform a fusion between the concatenated tensors F (from the backbone network) and $f(x)$. Then perform an encoding operation by applying an MLP. The MLP consists of fully connected layers, as shown in Figure 5, which help capture the complex relationships and interactions between the feature maps. The output of this MLP encoding operation is a fused feature map F' that incorporates the contextual information from both F and $f(x)$.

The purpose of this fusion and encoding process is to enhance the representation power of the feature maps and capture higher-level features that are beneficial for visual positioning tasks. By combining the features from different stages of the network, the performance of the Transformer-based visual positioning algorithm is improved.

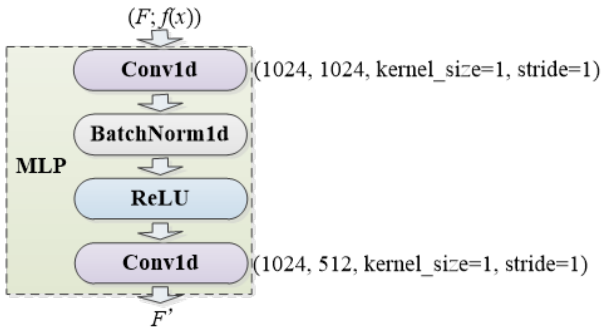
The formula for the fusion and encoding operations using the *MLP* is as follows:

$$F' = MLP(F; f(x)), \tag{4}$$

In this formula, the input to the *MLP* is the fused feature maps $(F; f(x))$, and the output is denoted as F'. The secondary fused feature map after the *MLP* coding and skip connection is denoted as $\bar{F}$. It can be represented as:

$$\bar{F} = F + F', \tag{5}$$

In this formula, F represents the feature map obtained from the backbone network, and F' represents the refined feature map obtained after the encoding operation using the *MLP*. This final feature map $\bar{F}$ incorporates both the low-level features from the backbone network and the higher-level contextual information captured through the fusion and encoding processes. It represents a more comprehensive and enriched representation of the input image, which can be used for further processing and classification tasks in the visual positioning algorithm.



(F; f(x))
MLP
Conv1d
(1024, 1024, kernel_size=1, stride=1)
BatchNorm1d
ReLU
Conv1d
(1024, 512, kernel_size=1, stride=1)
F'

Figure 5. MLP encoder.

3.3. Aggregation Module

In the module shown in Figure 6, the feature map $\bar{F}$ obtained from the previous step is further processed before being passed into the classifier. This processing involves dynamically adaptive pooling and a fully connected layer.

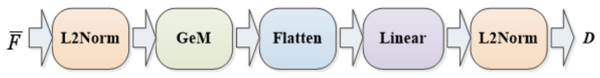


Figure 6. Aggregation module.

The feature map $\bar{F}$ is first subjected to an L_2 normalization operation to obtain $\bar{F}\_norm$. The $\bar{F}\_norm$ can be mathematically represented as follows:

$$\bar{F}\_norm = \|\bar{F}\|_2, \tag{6}$$

Then the normalized feature map $\bar{F}\_norm$ dynamically is subjected to an adaptive pooling using the GeM pooling operation.

$$\bar{F}\_gem = \left(\frac{1}{N}\sum_{i=1}^{N}(\bar{F}\_norm_i)^p\right)^{1/p}, \tag{7}$$

By setting p as a learnable parameter and adapting it through the network, the GeM module enables a flexible pooling operation that can dynamically adjust between average pooling and max pooling. This adaptability allows the network to learn the optimal pooling strategy for the given task without prior knowledge of whether average pooling or max pooling would be more effective. This pooling operation helps capture discriminative information from the feature map.

After the GeM pooling, the resulting pooled features are flattened and passed through a fully connected layer. This fully connected layer has an output dimension of 512, which means it transforms the features into a 512-dimensional representation. After the Flatten and Linear operations, an L_2 normalization is performed on the obtained descriptor to normalize its magnitude. The descriptor D can be mathematically represented as follows:

$$D = \|f(\bar{F}\_gem)\|_2, \tag{8}$$

In this formula, f represents the fully connected operation. The output of the Aggregation module is then passed into the classifier for the final classification. Overall, this module refines the features obtained from the previous steps and prepares them for final classification, enabling accurate positioning and navigation of the UAV based on visual information. Instead of using a traditional classification layer, the classifier in this case utilizes a cosine similarity measure layer. Mathematically, the cosine similarity between two vectors *D* and *Label* is computed as:

$$cos\theta = \frac{D \cdot Label^T}{\|D\| \cdot \|Label\|}, \tag{9}$$

The output of the classifier can be represented as follows:

$$output = s \times (cos\theta - m), \tag{10}$$

This formula, which includes parameters *s* and *m*, is a modified version of the softmax operation. The modified softmax operation with parameters *s* and *m* helps the model to better capture the similarities and differences between image descriptors, leading to improved classification accuracy or similarity measurement in the context of visual positioning tasks. Set m = 0.4 and s = 30.

The cross-entropy loss function is commonly used in classification tasks, including visual positioning. It measures the dissimilarity between the predicted probability distribution and the true label distribution. In the context of the trained network model, the loss function can be formulated as follows:

$$Loss = -\frac{1}{32}\sum_{i}\sum_{c=1}^{5965} Label_{ic} log(output_{ic}), \tag{11}$$

The loss function calculates the cross-entropy loss for each sample in the batch. The number of categories is 5965, the batch size is 32, $Label_{ic}$ is the symbol function that indicates whether the true category of sample i is equal to c (1 if true, 0 otherwise), and $output_{ic}$ is the predicted probability of sample i belonging to category c. For each sample, it compares the predicted probability of the true category with the true label distribution, penalizing incorrect predictions and encouraging the network to improve its accuracy. The loss function is summed over all samples in the batch and across all categories (represented by the summation symbol $\sum$). The goal of the training process is to minimize this loss by adjusting the network's parameters through backpropagation and gradient descent. By optimizing the cross-entropy loss function, the classifier can effectively utilize the extracted features to make accurate predictions about the pose or position of the UAV in the visual positioning algorithm.

During the evaluation, query descriptors D_{qu} obtained through the aggregation layer performed a strong L2 distance search in database descriptors D_{da} obtained through the aggregation layer to obtain 20 predicted results for every query image. *predictions* represents the predicted query results. Assuming that the number of query images is L, the size of the *predictions* is $L \times 20$.

$$predictions = L2(D_{qu}, D_{da}), \tag{12}$$

For real label data, real query labels perform KNN searches in real database labels to obtain real query results with distances less than 25 m. *positives* represents the real query results generated based on tags. The *positives* is composed of L one-dimensional arrays, each of which is different in length, and each array represents a sample.

$$positives = KNN(Label_{qu}, Label_{da}), \tag{13}$$

Finally, it is judged whether the prediction result of *predictions* exists in the real query result of *positives* to obtain the discriminant result. By checking whether the first

1, 5, 10, and 20 results of the *predictions* also exist in the *positives* generated according to the labels, the recalls under different thresholds are calculated.

$$R@1 = \frac{\sum_{i=0}^{L-1}((predictions[i][0]\ in\ positives[i]) = 1)}{L} \times 100\%, \tag{14}$$

$predictions[i][0]$ in $positives[i]$ indicates that the first result of the ith sample in *predictions* exists in the ith sample of positives. $\sum_{i=0}^{L-1}((predictions[i][0]\ in\ positives[i]) = 1)$ denotes the total number of positive samples that the first result of L samples in *predictions* exists in L samples of *positives*.

$$R@5 = \frac{\sum_{i=0}^{L-1}((any\ of\ predictions[i][:5]\ in\ positives[i]) = 1)}{L} \times 100\%, \tag{15}$$

$\sum_{i=0}^{L-1}((any\ of\ predictions[i][:5]\ in\ positives[i]) = 1)$ denotes the total number of positive samples that any of the first five results of L samples in *predictions* exists in L samples of *positives*.

$$R@10 = \frac{\sum_{i=0}^{L-1}((any\ of\ predictions[i][:10]\ in\ positives[i]) = 1)}{L} \times 100\%, \tag{16}$$

$\sum_{i=0}^{L-1}((any\ of\ predictions[i][:10]\ in\ positives[i]) = 1)$ denotes the total number of positive samples that any of the first ten results of L samples in *predictions* exists in L samples of *positives*.

$$R@20 = \frac{\sum_{i=0}^{L-1}((any\ of\ predictions[i][:20]\ in\ positives[i]) = 1)}{L} \times 100\%, \tag{17}$$

$\sum_{i=0}^{L-1}((any\ of\ predictions[i][:20]\ in\ positives[i]) = 1)$ denotes the total number of positive samples that any of the first twenty results of L samples in *predictions* exists in L samples of *positives*.

4. Results

4.1. Dataset

Considering the SF-XL dataset training set, it consists of a substantial 41.2 million images, amounting to approximately 1TB in size. This poses a significant challenge due to the sheer volume of data involved. Thus, our training set consists of a subset of the SF-XL dataset; the number of images is 59,650, the number of groups is 1, the number of images per category is 10, and the number of categories is 5965. Test Set: The database size is 27,191 images, and the query size is 1000 images. Verification Set: The database size is 8015 images, and the query size is 7993 images. UAV Test Set: The database size is 533 aerial images, and the query size is 93 aerial images. Figures 7–9 represent examples of the training set, the test set, and the verification set. Figure 10 provides an example of the UAV test set, which includes aerial images of 292 buildings obtained from Google Earth Pro software 1.3.36.242.

Figure 7. Training set example.

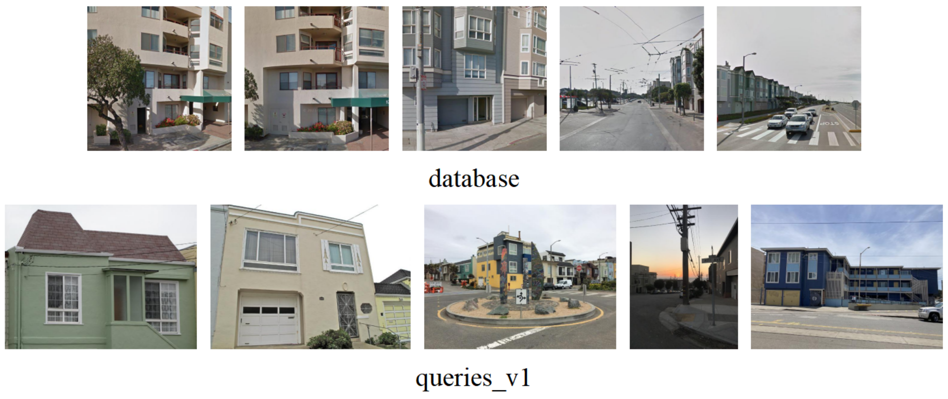


Figure 8. Test set example.

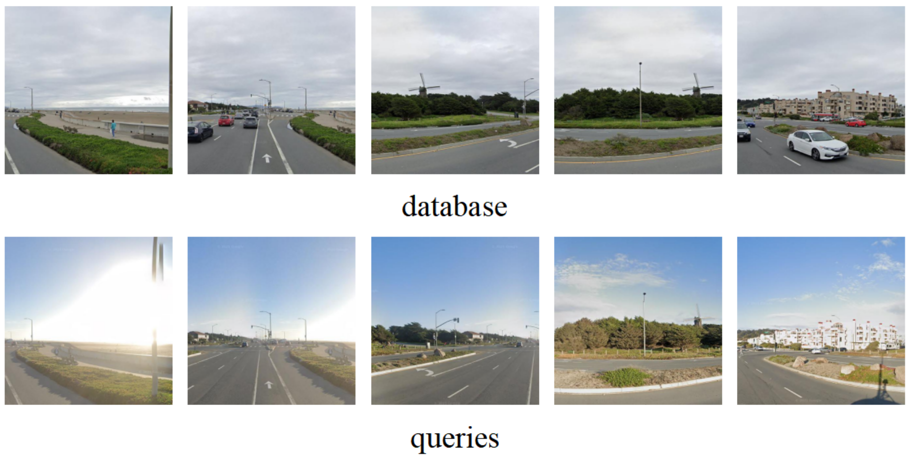


Figure 9. Verification set example.

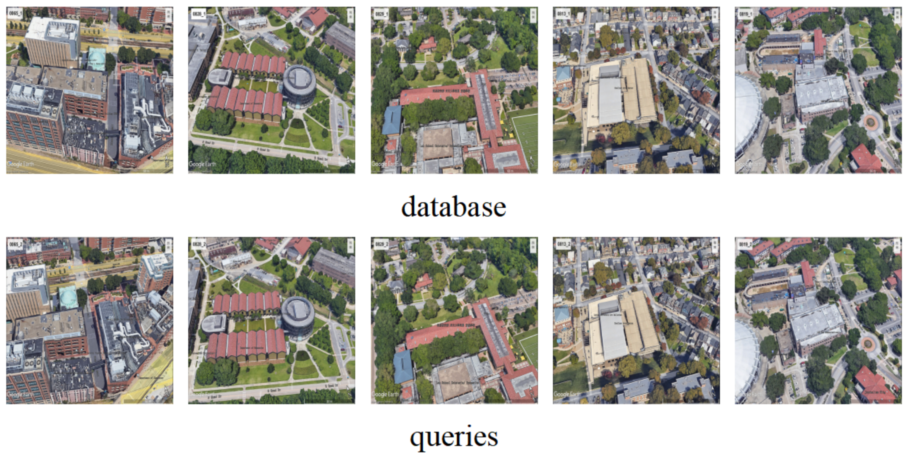


Figure 10. Drone test set example.

To construct the training set, a subset of SF-XL with a latitude range of 37.70 to 37.81 was chosen. In Figure 7, five street scenes from the training set are presented, each corresponding to latitudes 37.70, 37.71, 37.72, 37.73, and 37.74. These examples illustrate that the training set encompasses diverse scenes, including variations in illumination conditions. For the test set, the database comprises 27,191 images that cover the entire geographic area of SF-XL in 2013. Figure 8 showcases five schematic diagrams from the database, revealing a mixture of street and building images. Furthermore, the database incorporates images of the same building captured from different perspectives. The test set comprises 1000 query images collected from Flickr, intentionally selected from a distinct domain to closely resemble real-world scenes. Figure 8 exhibits five query images, indicating that they also contain a combination of street and building scenes. To form the validation set, a set of images throughout the city, encompassing 8015 database images and 7993 query images, was chosen. In Figure 9, five database images exhibiting different street views, along with five query images displaying varying illumination conditions within the same scene as the database, are presented. For the UAV test set, the Google Earth Pro software was utilized to capture images of 292 buildings, ensuring they had similar perspectives to those in the University-1652 drone dataset [41]. Each building was photographed from two different perspectives, which were subsequently divided into database images and query images for the drone test set. Some of the query images also have images with the same perspective as the database images. Furthermore, the image obtained by Google Earth Pro can also obtain the kml file containing the image coordinate information. The kml file obtained from Google Earth Pro contains image coordinate information, including longitude, latitude, and heading data. This information is leveraged to generate retrieval labels for the UAV test images. To obtain the retrieval labels, the kml file's longitude, latitude, and heading details are used. An online conversion tool is employed to derive the UTM coordinates from the latitude and longitude coordinates. Consequently, five retrieval labels for a UAV image are obtained, comprising UTM coordinates, latitude and longitude coordinates, and heading information. Figure 10 showcases five database images exhibiting distinct buildings, alongside five query images captured from various rotation angles, all within the collected UAV test set.

4.2. Training Environment

The network models were trained on two servers equipped with Ubuntu 18.04 and CUDA 11.1. The deep learning framework PyTorch was used for training the models. The ResNet18, ResNet50, ResNet101, and VGG16 backbone architectures were selected for research. For the ResNet18 architecture, three variations were trained: ResNet18, AGResNet18 (AGCosPlacle network applied to ResNet18), and PE + AGResNet18 (AGResNet18 with an additional sinusoidal position coding module). The ResNet18, AGResNet18, and PE + AGResNet18 models were trained on an RTX3060 GPU. For the other backbone architectures (ResNet50, ResNet101, and VGG16), the following models were trained: ResNet50, AGResNet50 (AGCosPlacle network applied to ResNet50), ResNet101, AGResNet101 (AGCosPlacle network applied to ResNet101), VGG16, and AGVGG16
(AGCosPlacle network applied to VGG16). These models were trained on an RTX3090 GPU. The choice of different backbones and variations of the AGCosPlacle network allows for a comparative analysis of their performance in the visual positioning tasks.

AGCosPlace hyperparameter settings: grouping parameters, training parameters, and data augmentation parameters are set as shown in Tables 1–3.

Table 1. Grouping parameters.

Groups Parameters	Setting
M	10
α	30
N	5
L	2
ground_num	1
Min_images_per_class	10

Figure 11 depicts the Loss curves obtained during the training of various models, including ResNet18, AGResNet18, ResNet50, AGResNet50, ResNet101, AGResNet101, VGG16, and AGVGG16.

Table 2. Training parameters.

Training Parameters	Setting
resized	512 × 512
batch size	32
epoch	50
iterations_per_epoch	10,000
lr	0.00001
classifiers_lr	0.01
optimizer	Adam
seed	0

Table 3. Data augmentation parameters.

Data Augmentation Parameters	Setting
brightness	0.7
contrast	0.7
hue	0.5
saturation	0.7
random_resized_crop	0.5

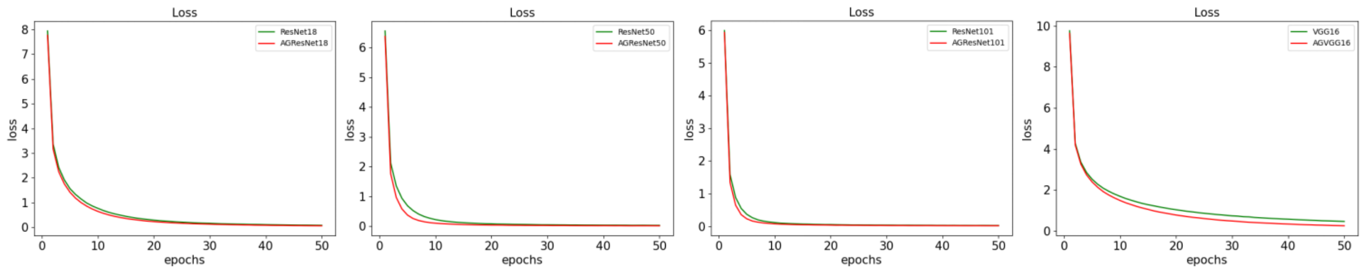


Figure 11. Loss curve of CosPlace and AGCosPlace models training.

It can be seen from Figure 11 that the loss curves corresponding to the AGCosPlace network models (AGResNet18, AGResNet50, AGResNet101, AGVGG16) with our designed Attentional GNN module have lower loss values, which have better convergence effects than the models with CosPlace network models (ResNet18, ResNet50, ResNet1011, VGG16).

4.3. Performance Evaluation of AGCosPlace and CosPlace

The evaluation of the trained models (ResNet18, AGResNet18, PE + AGResNet18, ResNet50, AGResNet50, ResNet101, AGResNet101, VGG16, and AGVGG16) was performed using the positive sample distance threshold set to 25. The recall rate at different ranks, including R@1, R@5, R@10, and R@20, was used to measure the performance. The results are presented in Table 4.

In our research, experiments using a 512-dimensional descriptor were conducted to align with the optimal performance achieved by the CosPlace algorithm. To accelerate the training process, a selective training strategy was employed for the ResNet and VGG16 networks. Specifically, only the third and fourth layers of the ResNet network were trained, while the preceding layers were frozen. Similarly, for the VGG16 network, only the last layer was trained, while the previous layers were frozen.

Considering the number of trainable layers, the ResNet-based AGCosPlace algorithm shows improvements in the four evaluation indicators (R@1, R@5, R@10, and R@20) compared with the baseline ResNet models, as shown in Table 4. Specifically, AGResNet18 achieves a 0.6% increase in R@1, 0.1% increase in R@5, 0.3% increase in R@10, and a 1% increase in R@20 compared with ResNet18. AGResNet50 outperforms ResNet50 with a 1% increase in R@1, a 1.9% increase in R@5, a 1.8% increase in R@10, and a 1.1% increase in R@20. AGResNet101 shows improvements of 0.3% in R@1, 0.4% in R@5, 0.8% in R@10, and 0.6% in R@20 compared with ResNet101. However, in the case of AGVGG16, a slight decrease in performance is observed. AGVGG16 exhibits a 0.2% decrease in R@1, a 1.5% decrease in R@5, a 1.4% decrease in R@10, and a 0.2% decrease in R@20 compared with VGG16. Based on these results, it can be concluded that the Attentional GNN module designed in this study is more suitable for enhancing the performance of the ResNet network compared with VGG16. The improvements achieved by AGCosPlace indicate the effectiveness of the Attentional GNN module in enhancing the feature representation capabilities of ResNet-based models.

Table 4. Performance of 9 network models based on SF-XL test set.

Model	Desc.dim	channels. num.in. last.conv	R@1 (%)	R@5 (%)	R@10 (%)	R@20 (%)	Train_Time (Days)
ResNet18	512	512	53.7	66.0	71.5	75.3	2
AGResNet18	512	512	54.3	66.1	71.8	76.3	2
PE+ AGResNet18	512	512	53.5	65.9	71.3	76.5	2
ResNet50	512	2048	59.1	72.1	76.9	80.5	2
AGResNet50	512	2048	60.1	74.0	78.7	81.6	3
ResNet101	512	2048	62.4	73.8	77.8	82.4	4
AGResNet101	512	2048	62.7	74.2	78.6	83.0	5
VGG16	512	512	61.0	73.3	77.7	80.6	3
AGVGG16	512	512	60.8	71.8	76.3	80.4	3

Furthermore, experiments were conducted by incorporating a sinusoidal position encoding module into the AGResNet18 network. However, the results presented in the table indicate that the position encoding module only improves the recall rate at a threshold of R@20 while diminishing the performance at the other three recall rate values. Based on this observation, we have decided not to include the position encoding module in our further research on the remaining network models.

The training time for ResNet18, AGResNet18, and PE + AGResNet18 on the RTX3060 is the same, as they have similar network structures. Similarly, ResNet50 and ResNet101, which have deeper network structures, require more training time compared with ResNet18. The RTX3090, being a more powerful GPU compared with the RTX3060, can significantly reduce the training time for complex networks such as ResNet50. Therefore, the training time for ResNet50 on the RTX3090 can be similar to the training time for ResNet18 on the RTX3060. The addition of the Attentional GNN module increases the complexity of the network model and the number of parameters, which can further impact the training time.

In the case of simpler networks such as AGResNet18 and AGVGG16 (where only the last layer is trained), the additional training time required compared with ResNet18 and VGG16 is not significant. Since the majority of the network remains frozen, which has a relatively smaller number of parameters. However, for more complex networks such as AGResNet50 and AGResNet101, the training time can be extended by approximately one day compared with ResNet50 and ResNet101. This increase in training time is due to the added complexity of the Attentional GNN module and the larger number of parameters to train in these networks. The additional computations required by the attention mechanisms and the increased model capacity contribute to the longer training duration. The complexity of each network can be inferred from the number of convolutional output channels in the last layer of the aforementioned table and the corresponding training times.

From the perspective of network complexity, it can be observed that for the CosPlace models (ResNet18, ResNet50, and ResNet101), increasing the depth of the network structure leads to improved performance in terms of R@1, R@5, R@10, and R@20 metrics. This improvement is consistent with the common understanding that deeper networks have the potential to capture more complex features and representations, leading to better recognition accuracy. Similarly, for the AGCosPlace models (AGResNet18, AGResNet50, and AGResNet101), increasing the depth of the network structure also results in improved performance across R@1, R@5, R@10, and R@20 metrics. The inclusion of the Attentional GNN module in AGCosPlace further enhances the network's ability to capture and aggregate feature point information, leading to better recognition accuracy. Regarding the VGG16 network model, although its structure is not as complex as ResNet50, it still achieves slightly better performance than ResNet50 in terms of R@1, R@5, R@10, and R@20 metrics. However, it is important to note that the training time for the VGG16 network model is one day longer than ResNet50. Therefore, considering the trade-off between performance and training efficiency, network research based on ResNet models is generally more efficient.

In summary, our designed AGCosPlace algorithm, utilizing the ResNet network as the backbone, demonstrates improvements in the R@1, R@5, R@10, and R@20 indicators when the descriptor dimension is set to 512. This improvement validates the effectiveness of our algorithm in addressing the issue of poor recall rates observed in Transformer-based CosPlace research. By incorporating the Attentional GNN module and leveraging the ResNet architecture, the performance of the CosPlace algorithm is successfully enhanced.

In addition, a Drone test set is created to simulate a UAV perspective, following the format of the SF-XL test set. This test set comprises 533 aerial images with UTM coordinate information for the database and 93 aerial images with UTM coordinate information for the queries. Evaluate the performance of our proposed algorithm based on this UAV dataset and apply it to the UAV visual positioning task. The evaluation results of the AGCosPlace and CosPlace network models using aerial images are presented in Table 5.

Table 5. Performance of 9 network models based on Drone test set.

Model	**Test Set**	**Desc.dim**	**channels. num.in. last.conv**	**R@1 (%)**	**R@5 (%)**	**R@10 (%)**	**R@20 (%)**
ResNet18	Drone	512	512	61.3	66.7	66.7	68.8
AGResNet18	Drone	512	512	64.5	67.7	67.7	71.0
PE+ AGResNet18	Drone	512	512	64.5	67.7	67.7	68.8
ResNet50	Drone	512	2048	66.7	68.8	69.9	71.0
AGResNet50	Drone	512	2048	66.7	69.9	72.0	74.2
ResNet101	Drone	512	2048	64.5	69.9	72.0	72.0
AGResNet101	Drone	512	2048	62.4	67.7	71.0	72.0
VGG16	Drone	512	512	67.7	74.2	75.3	76.3
AGVGG16	Drone	512	512	69.9	72.0	74.2	76.3

From Table 5, it can be observed that the proposed AGCosPlace visual positioning algorithm performs well on the UAV-based dataset, particularly when ResNet18 and ResNet50 are used as the backbone networks. It is evident that AGResNet18 outperforms ResNet18 by 3.2%, 1%, 1%, and 2.2% under the R@1, R@5, R@10, and R@20 indicators, respectively. While the PE + AGResNet18 network, incorporating position coding, achieves promising results compared with ResNet18, the addition of the position coding module does not further improve the performance of AGResNet18. The PE + AGResNet18 network performs equally to AGResNet18 under the R@1, R@5, and R@10 indicators but exhibits attenuated performance under the R@20 indicator. This suggests that the position-coding module does not enhance the performance of the AGCosPlace visual positioning model on the aerial image test set.

With regards to deeper network models, for ResNet50, the addition of the Attentional GNN module in AGResNet50 results in a performance improvement of 1.1%, 2.1%, and 3.2% under the R@5, R@10, and R@20 indicators, respectively, compared with ResNet50. However, the performance remains the same as ResNet50 under the R@1 indicator. In the case of ResNet101, the network model is already well-trained on the aerial image test set, and the inclusion of the Attentional GNN module causes overfitting, leading to inferior performance of AGResNet101 compared with ResNet101 at R@1, R@5, and R@10. As for the VGG16 network, the addition of the Attentional GNN module in AGVGG16 only results in a 2.2% increase in performance under the R@1 indicator, while its performance deteriorates under the R@5 and R@10 indicators. Consequently, the proposed AGCosPlace algorithm is more suitable for improving the ResNet18 and ResNet50 backbone networks in the context of testing based on aerial images.

During the visualization, query descriptors obtained through the aggregation layer perform a strong L2 distance search in database descriptors obtained through the aggregation layer to obtain the predicted six results. For real label data, real query labels perform KNN searches in real database labels to obtain real query results. If the predicted result exists in the real query result, the judgment is correct, indicated by a green box; if it does not exist in the real query result, the judgment is wrong, indicated by a red box.

$$Judge[j] = predictions[i][j] \ in \ positives[i], \ i = 0, \ldots, L-1 \ j = 0, \ldots, 5 \tag{18}$$

predictions represents the predicted query results, and the size is $L \times 6$. L represents the number of query images. *positives* represents the real query results generated based on tags, which are composed of L arrays of different lengths. $Judge$ is a 1×6 array that stores Boolean values. If the jth result of the ith sample in *predictions* exists in the ith sample of *positives*, $Judge[j]$ is True, represented by a green box. If it does not exist in the ith sample of *positives*, $Judge[j]$ is False, represented by a red box. The reasoning visualization diagrams of the eight network models trained on both the SF-XL test set and the Drone test set are in Appendix A.

5. Discussion

The AGCosPlace network, which combines the Attentional GNN module with ResNet18, ResNet50, and ResNet101 networks, has shown improvements in the four indicators (R@1, R@5, R@10, and R@20) on the SF-XL test set. This algorithm effectively enhances the performance of the Transformer-based CosPlace algorithm without altering the input image size. Our research has focused on specific datasets within the SF-XL dataset. In the future, our algorithm can be applied to all SF-XL datasets for further improvement and performance validation. Furthermore, the proposed algorithm has been verified on the Drone test set, demonstrating its effectiveness in achieving accurate visual positioning for UAV applications. The combination of the Attentional GNN module with ResNet18 and ResNet50 networks has yielded even better performance, underscoring the generalizability of the trained network model. However, the constructed UAV test set only includes UAV perspective images captured under two rotation angles for performance verification, which lacks verification in various scenarios such as different illuminations, weather conditions, and terrains. Because the proposed AGCosPlace model is trained on the training subset of SF-XL, which contains different illumination scenarios, its robustness to UAV test sets with diverse illumination conditions can be expected. However, the performance of AGCosPlace for UAV test sets in scenarios such as different weather and terrains remains to be further explored. In future research, it would be beneficial to conduct a comprehensive performance evaluation of visual positioning algorithms using UAV test sets that encompass different scenarios. This evaluation will help to assess the algorithms' capabilities to handle varying environmental conditions, such as different illumination, weather patterns, and terrains. In addition, the research on visual positioning algorithms for UAV applications may include labeling a larger number of UAV images to train the network model of the proposed algorithm, further advancing its capabilities. This will ultimately contribute to the development of more robust and versatile visual positioning solutions for UAVs in real-world settings.

Author Contributions: Conceptualization, Y.G.; methodology, Y.G.; software, Y.G. and F.Y.; validation, Y.G., Y.Z. and F.Y.; formal analysis, Y.G.; data curation, Y.G.; writing—original draft preparation, Y.G.; writing—review and editing, Y.G. and Y.Z.; visualization, Y.G.; supervision, Y.Z. and F.Y.; project administration, Y.Z.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Special Foundation for Beijing Tianjin Hebei Basic Research Cooperation (J210008, H2021202008), and the Inner Mongolia Discipline Inspection and Supervision Big Data Laboratory (IMDBD202105).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A

Figures A1 and A2 depict the reasoning visualization diagrams of the eight network models trained on both the SF-XL test set and the Drone test set.

In Figure A1, the query image from the SF-XL test set is used to generate inference results from the trained network models. Each network model predicts six results, represented by green and red boxes. The green boxes indicate correctly matched images, while the red boxes represent incorrect matches. It can be observed that even for the false matching images, there exist some similar structural patterns to the query image, although they are not actual matches.

Moving on to Figure A2, it shows the inference results for a query image from the Drone test set generated by the trained network models. Similarly, green boxes indicate correctly matched images, while red boxes indicate incorrect matches. Both the AGCosPlace algorithm and the CosPlace algorithm demonstrate the ability to retrieve correctly matched UAV images, thereby enabling UAV visual positioning. Notably, when the predicted correctly matched image has a different angle from the query image, the network models of the AGCosPlace algorithm (AGResNet18, AGVGG16) and the CosPlace algorithm (VGG16) yield more comprehensive detection results compared with other network models. Moreover, the mismatched images identified by the proposed AGCosPlace algorithm based on ResNet18 exhibit more similar structural patterns than those identified by the CosPlace algorithm.

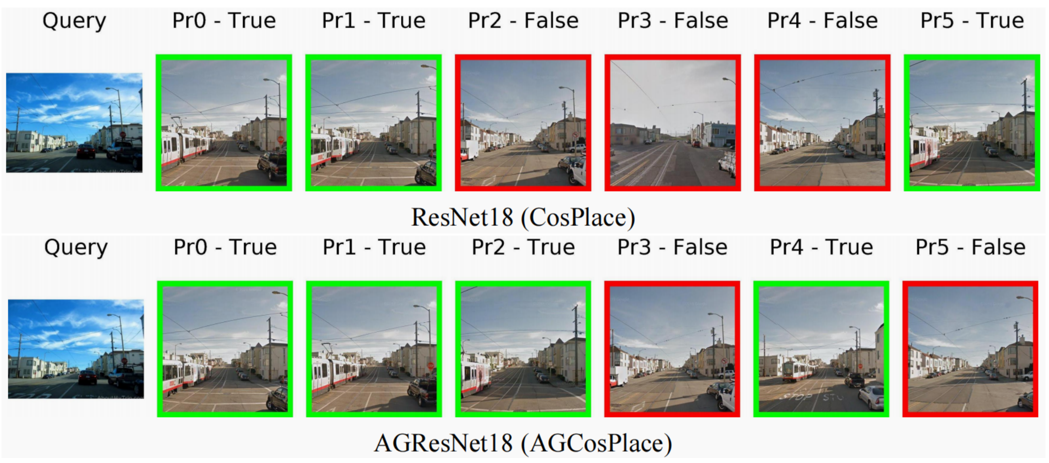


Figure A1. *Cont.*

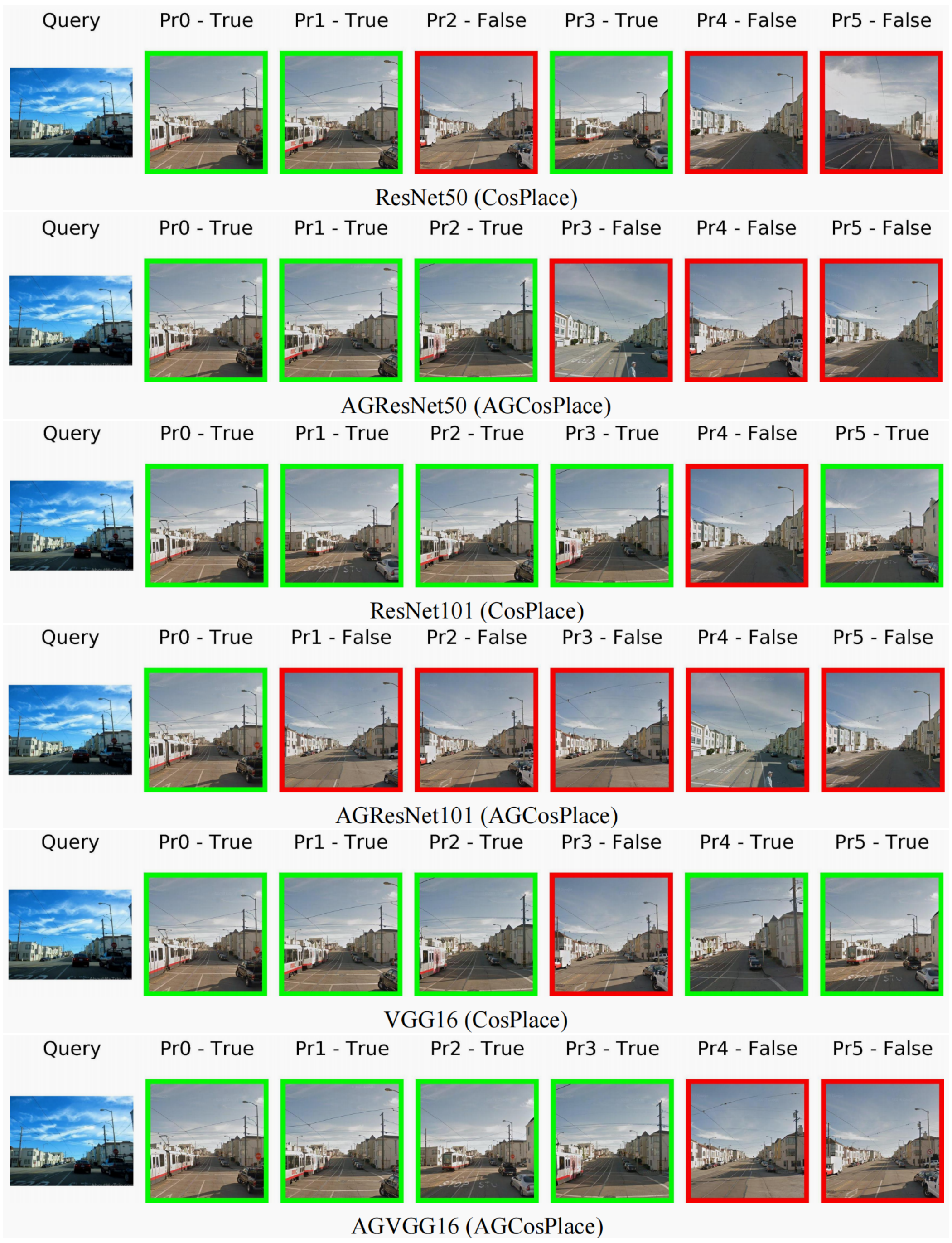


Figure A1. Visualization diagrams of AGCosPlace and CosPlace reasoning based on SF-XL test set.

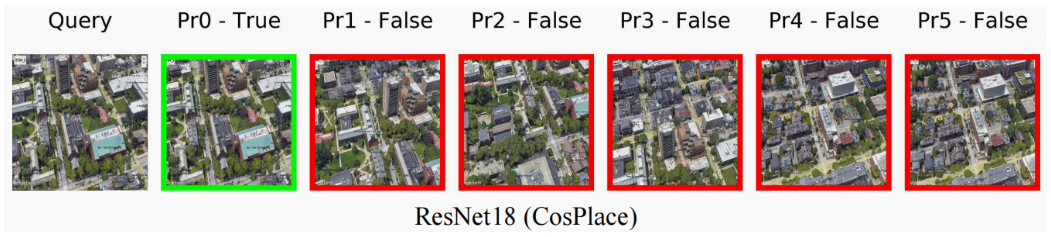


Figure A2. *Cont.*

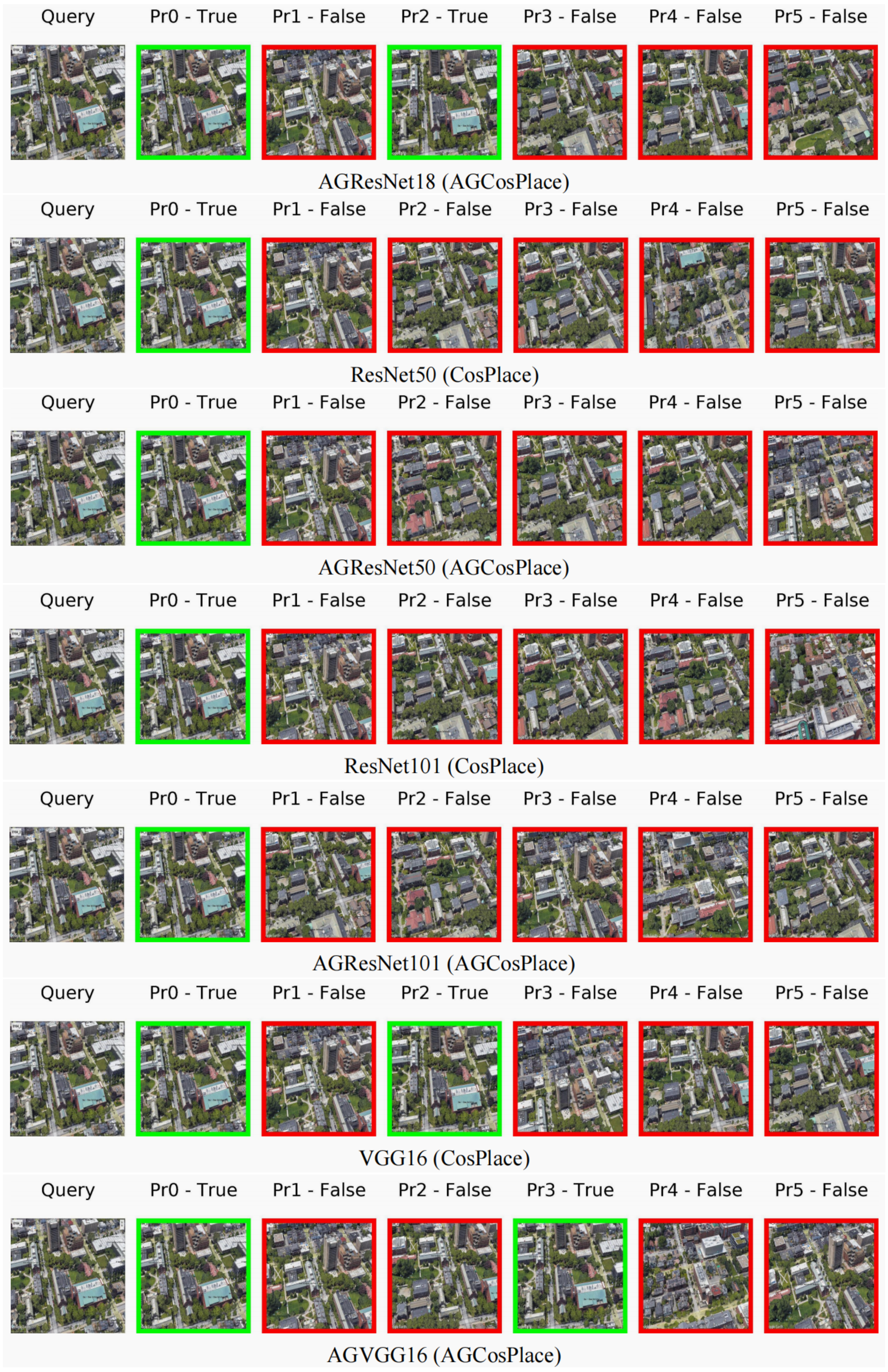


Figure A2. Visualization diagrams of AGCosPlace and CosPlace reasoning based on Drone test set.

References

1. Liu, M.; Yang, J.; Gui, G. DSF-NOMA: UAV-assisted emergency communication technology in a heterogeneous Internet of Things. *IEEE Internet Things J.* **2019**, *6*, 5508–5519. [CrossRef]
2. Radoglou-Grammatikis, P.; Sarigiannidis, P.; Lagkas, T.; Moscholios, I. A compilation of UAV applications for precision agriculture. *Comput. Netw.* **2020**, *172*, 107148. [CrossRef]
3. Mademlis, I.; Mygdalis, V.; Nikolaidis, N.; Montagnuolo, M.; Negro, F.; Messina, A.; Pitas, I. High-level multiple-UAV cinematography tools for covering outdoor events. *IEEE Trans. Broadcast.* **2019**, *65*, 627–635. [CrossRef]
4. Feroz, S.; Abu Dabous, S. Uav-based remote sensing applications for bridge condition assessment. *Remote Sens.* **2021**, *13*, 1809. [CrossRef]
5. Outay, F.; Mengash, H.A.; Adnan, M. Applications of unmanned aerial vehicle (UAV) in road safety, traffic and highway infrastructure management: Recent advances and challenges. *Transp. Res. Part A Policy Pract.* **2020**, *141*, 116–129. [CrossRef] [PubMed]
6. Chen, D.; Gao, G.X. Probabilistic graphical fusion of LiDAR, GPS, and 3D building maps for urban UAV navigation. *Navigation* **2019**, *66*, 151–168. [CrossRef]
7. Lu, Y.; Xue, Z.; Xia, G.S.; Zhang, L. A survey on vision-based UAV navigation. *Geo-Spat. Inf. Sci.* **2018**, *21*, 21–32. [CrossRef]
8. Gyagenda, N.; Hatilima, J.V.; Roth, H.; Zhmud, V. A review of GNSS-independent UAV navigation techniques. *Robot. Auton. Syst.* **2022**, *152*, 104069. [CrossRef]
9. Raja, G.; Suresh, S.; Anbalagan, S.; Ganapathisubramaniyan, A.; Kumar, N. PFIN: An efficient particle filter-based indoor navigation framework for UAVs. *IEEE Trans. Veh. Technol.* **2021**, *70*, 4984–4992. [CrossRef]
10. Gao, K.; Wang, H.; Lv, H.; Gao, P. A DL-based High-Precision Positioning Method in Challenging Urban Scenarios for B5G CCUAVs. *IEEE J. Sel. Areas Commun.* **2023**, *41*, 1670–1687. [CrossRef]
11. Eckenhoff, K.; Geneva, P.; Huang, G. MIMC-VINS: A versatile and resilient multi-IMU multi-camera visual-inertial navigation system. *IEEE Trans. Robot.* **2021**, *37*, 1360–1380. [CrossRef]
12. Wang, Z.H.; Qin, K.Y.; Zhang, T.; Zhu, B. An intelligent ground-air cooperative navigation framework based on visual-aided method in indoor environments. *Unmanned Syst.* **2021**, *9*, 237–246. [CrossRef]
13. Arafat, M.Y.; Alam, M.M.; Moh, S. Vision-based navigation techniques for unmanned aerial vehicles: Review and challenges. *Drones* **2023**, *7*, 89. [CrossRef]
14. Jiang, X.; Ma, J.; Xiao, G.; Shao, Z.; Guo, X. A review of multimodal image matching: Methods and applications. *Inf. Fusion* **2021**, *73*, 22–71. [CrossRef]
15. Ramadan, M.; El Tokhey, M.; Ragab, A.; Fath-Allah, T.; Ragheb, A. Adopted image matching techniques for aiding indoor navigation. *Ain Shams Eng. J.* **2021**, *12*, 3649–3658. [CrossRef]
16. Ma, J.; Jiang, X.; Fan, A.; Jiang, J.; Yan, J. Image matching from handcrafted to deep features: A survey. *Int. J. Comput. Vis.* **2020**, *129*, 23–79. [CrossRef]
17. Hai, J.; Hao, Y.; Zou, F.; Lin, F.; Han, S. A visual navigation system for UAV under diverse illumination conditions. *Appl. Artif. Intell.* **2021**, *35*, 1529–1549. [CrossRef]
18. Sarlin, P.E.; DeTone, D.; Malisiewicz, T.; Rabinovich, A. Superglue: Learning feature matching with graph neural networks. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 19–23 June 2020; pp. 4938–4947. [CrossRef]
19. Berton, G.; Masone, C.; Caputo, B. Rethinking visual geo-localization for large-scale applications. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–23 June 2022; pp. 4878–4888. [CrossRef]
20. Krichen, M.; Adoni, W.Y.H.; Mihoub, A.; Alzahrani, M.Y.; Nahhal, T. Security challenges for drone communications: Possible threats, attacks and countermeasures. In Proceedings of the 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH 2022), Riyadh, Saudi Arabia, 9–11 May 2022; pp. 184–189. [CrossRef]
21. Ko, Y.; Kim, J.; Duguma, D.G.; Astillo, P.V.; You, I.; Pau, G. Drone secure communication protocol for future sensitive applications in military zone. *Sensors* **2021**, *21*, 2057. [CrossRef]
22. Rezwan, S.; Choi, W. Artificial intelligence approaches for UAV navigation: Recent advances and future challenges. *IEEE Access* **2022**, *10*, 26320–26339. [CrossRef]
23. Couturier, A.; Akhloufi, M.A. A review on absolute visual localization for UAV. *Robot. Auton. Syst.* **2021**, *135*, 103666. [CrossRef]
24. Wan, X.; Liu, J.; Yan, H.; Morgan, G.L. Illumination-invariant image matching for autonomous UAV localisation based on optical sensing. *ISPRS J. Photogramm. Remote Sens.* **2016**, *119*, 198–213. [CrossRef]
25. Zhang, X.; He, Z.; Ma, Z.; Wang, Z.; Wang, L. Llfe: A novel learning local features extraction for uav navigation based on infrared aerial image and satellite reference image matching. *Remote Sens.* **2021**, *13*, 4618. [CrossRef]
26. Ren, K.; Ding, L.; Wan, M.; Gu, G.; Chen, Q. Target localization based on cross-view matching between UAV and satellite. *Chin. J. Aeronaut.* **2022**, *35*, 333–341. [CrossRef]
27. Ding, L.; Zhou, J.; Meng, L.; Long, Z. A practical cross-view image matching method between UAV and satellite for UAV-based geo-localization. *Remote Sens.* **2020**, *13*, 47. [CrossRef]
28. Xu, C.; Liu, C.; Li, H.; Ye, Z.; Sui, H.; Yang, W. Multiview Image Matching of Optical Satellite and UAV Based on a Joint Description Neural Network. *Remote Sens.* **2022**, *14*, 838. [CrossRef]

29. Fu, C.; Li, B.; Ding, F.; Lin, F.; Lu, G. Correlation filters for unmanned aerial vehicle-based aerial tracking: A review and experimental evaluation. *IEEE Geosci. Remote Sens. Mag.* **2021**, *10*, 125–160. [CrossRef]
30. Li, S.; Liu, Y.; Zhao, Q.; Feng, Z. Learning residue-aware correlation filters and refining scale for real-time uav tracking. *Pattern Recognit.* **2022**, *127*, 108614. [CrossRef]
31. An, Z.; Wang, X.; Li, B.; Xiang, Z.; Zhang, B. Robust visual tracking for UAVs with dynamic feature weight selection. *Appl. Intell.* **2022**, *53*, 3836–3849. [CrossRef]
32. Wang, Y. Hybrid efficient convolution operators for visual tracking. *J. Artif. Intell.* **2021**, *3*, 63.
33. Hou, Z.; Wang, Z.; Pu, L.; Ma, S.; Yang, Z.; Fan, J. Target drift discriminative network based on deep learning in visual tracking. *J. Electron. Imaging* **2022**, *31*, 043052. [CrossRef]
34. Lin, H.; Zhou, J.; Gan, Y.; Vong, C.M.; Liu, Q. Novel up-scale feature aggregation for object detection in aerial images. *Neurocomputing* **2020**, *411*, 364–374. [CrossRef]
35. Bellavia, F.; Colombo, C. Is there anything new to say about SIFT matching? *Int. J. Comput. Vis.* **2020**, *128*, 1847–1866. [CrossRef]
36. Chen, Y.; Chen, R.; Liu, M.; Xiao, A.; Wu, D.; Zhao, S. Indoor visual positioning aided by CNN-based image retrieval: Training-free, 3D modeling-free. *Sensors* **2018**, *18*, 2692. [CrossRef]
37. Ha, I.; Kim, H.; Park, S.; Kim, H. Image retrieval using BIM and features from pretrained VGG network for indoor localization. *Build. Environ.* **2018**, *140*, 23–31. [CrossRef]
38. Arandjelovic, R.; Gronat, P.; Torii, A.; Pajdla, T.; Sivic, J. NetVLAD: CNN architecture for weakly supervised place recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 1437–1451. [CrossRef]
39. Milford, M.J.; Wyeth, G.F. Mapping a suburb with a single camera using a biologically inspired SLAM system. *IEEE Trans. Robot.* **2008**, *24*, 1038–1053. [CrossRef]
40. Hu, S.; Lee, G.H. Image-based geo-localization using satellite imagery. *Int. J. Comput. Vis.* **2020**, *128*, 1205–1219. [CrossRef]
41. Zheng, Z.; Wei, Y.; Yang, Y. University-1652: A multi-view multi-source benchmark for drone-based geo-localization. In Proceedings of the 2020 28th ACM International Conference on Multimedia, Seattle, WA, USA, 12–16 October 2020; pp. 1395–1403. [CrossRef]