SEEK: Detecting GPS Spoofing via a Sequential Dashcam based Vehicle Localization Framework

***Abstract*—GPS spoofing is a great threat to the safety of transportation systems as well as other systems that rely on GPS for navigation. This paper proposes a novel computer vision based approach for GPS spoofing detection, termed SEquential dashcam-based vEhicle localization frameworK (SEEK). SEEK utilizes vehicle dashcam images to identify a vehicle's true location and detects possible GPS spoofing attacks through verifying if the reported GPS locations of the vehicle are correct. However, it is nontrivial to use dashcam images for vehicle localization due to multiple challenges caused by real-world driving, including the complicated lighting/weather conditions, season/timing variations of the images, large blockage ratio in the images, and varying driving speeds. SEEK features a unique design with novel schemes to address complicated lighting/weather conditions, transform images to align with season changes, reduce blockage, and adopt a sequential image matching scheme. The performance evaluation shows that SEEK significantly outperforms the previous GPS spoofing detection scheme, and achieves a detection accuracy of up to 94%.**

***Index Terms*—GPS spoofing, deep learning, vehicle localization, image matching**

I. Introduction

Today the Global Positioning System (GPS) service has been widely used in a variety of applications such as smartphones, wearable devices, and almost all transportation systems, including autonomous vehicles. They all rely on GPS to benefit from location-based services, e.g., navigation, truck/vehicle monitoring, reporting self-location in an emergency or for rescue, etc. While GPS has become widely used today, it is rather vulnerable to spoofing attacks, and the consequence is often catastrophic or life-threatening. GPS spoofing attacks can be easily launched via a portable and low-cost software defined radio, such as HackRF [1], to attack the GPS devices embedded in a wide range of systems [2]–[10]. Attackers can launch various GPS spoofing attacks by broadcasting a crafted GPS signal with a higher power than the authentic signal from GPS satellites. Nearby GPS-enabled devices would then receive the fake GPS signal due to its stronger power and are under the control of the GPS spoofer/attacker. ***An external attacker*** can launch the GPS spoofing attack by tailgating and fooling a target vehicle to a different route toward an unsafe location. Moreover, ***an escape driver***, as shown in Fig. 1, can send a spoofed GPS signal to the GPS tracker on a commercial vehicle or ride-hailing vehicles such as Uber or Lyft, to fake a normal trajectory, while the malicious driver takes the vehicle and/or passengers to an unsafe location [3].

Due to its potentially catastrophic consequence, GPS spoofing attacks have motivated various studies for effective countermeasures. For example, the works in [11]–[13] proposed cryptography techniques to encrypt the signals from satellites with a secret key. The authors in [14]–[18] proposed approaches to detect GPS spoofing attacks by examining the received GPS signal through anomaly detection on the signal waveform, or the calculation of the signal angle of arrival [19]–[21]. However, these approaches either require a massive upgrade to the existing GPS infrastructure or modifications of the GPS receiving devices, which does not seem a practical solution in the near future. Instead of resolving the issue of GPS reception, some works shed lights on using other devices, such as motion sensors in transportation systems, to assist GPS navigation [23]–[25], but did not propose concrete algorithms or schemes. At last, [22] followed this direction and proposed a deep learning based detection method through reconstructing the true vehicle trajectory from inertial sensors such as the accelerometer and gyroscope. Nevertheless, due to the limitation on the accuracy of the commercial accelerometer and gyroscope, it faces challenges in detecting a spoofing attack if the spoofed trajectory is carefully crafted, and thus, the inertial sensor data appear similar to those of the authentic vehicle moving trajectory.

Driven by the lack of a practical and effective solution to GPS spoofing detection, in this paper, we propose a computer vision based framework to detect GPS spoofing by taking advantage of a commonly installed device in vehicles – dashcam. Dashcams are widely used today to provide vehicle and road safety, e.g., documenting accidents for settling insurance claims, challenging traffic tickets, recording thieves to the vehicle, alerting lane departure and possible collision to assist drivers, etc. The proposed framework extracts dashcam images and uses them to identify the real location of the vehicle based on some reference images. Then it detects a GPS spoofing attack by checking if the vehicle is driving on the reported GPS route.

In the literature, there have been some studies recently on a relevant problem, image geo-localization through image retrieval. The *cross-view image matching* (CVM) geo-localization [28]–[33] uses *geo-tagged* satellite images to identify the location of a given ground image, by scanning the satellite image database to find the images that look close to the query image. The geo-location associated with the retrieved satellite image is then used to be the location of the query image. To maintain a good accuracy in CVM, a panorama view of the ground image is typically required to make sure most objects, especially landmark objects, in the satellite image also appear in the query ground image.

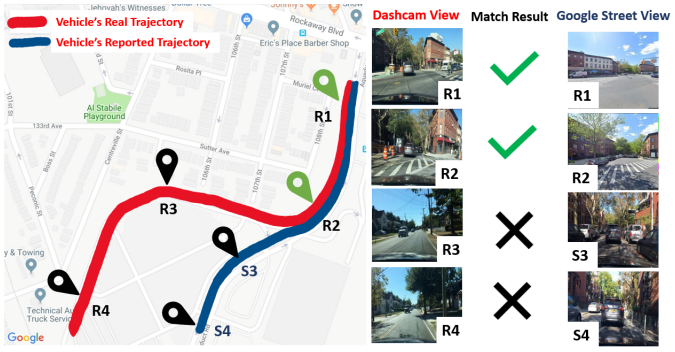


Fig. 1: An illustration of the GPS spoofing attack. A ride-hailing vehicle is driving on the path from R1 to R4 (red line). Meanwhile, the malicious/escape driver spoofs the GPS signal to deceive the monitoring center that the vehicle is driving on the spoofed route (blue line).

For example, [32] mounted 12 fish eye near-infrared (NIR) cameras on the top of a vehicle to obtain the panorama of the ground image, which significantly increases the matching accuracy to the corresponding geo-tagged satellite image.

While the existing CVM studies have demonstrated effective performance in their datasets, we have found that it is rather challenging to directly use the CVM schemes for the dashcam image localization in our GPS spoofing detection framework, due to the following reasons.

1) Modern vehicles usually have only one or two safety cameras. It is inadequate to use those cameras to obtain a panorama ground view at a given location which is required by CVM to achieve a good accuracy [32]. Clearly, a fixed-angle query image (i.e., a dashcam image) has a much narrower view that contains less information compared to the panorama view of the query image in CVM.
2) Dashcam images captured from real-world driving are typically taken at a lower height and hence contain much more blockage, such as due to the vehicles in the front. In contrast, the CVM ground images are usually taken from a 360 Camera installed on the top of the vehicle and hence contain much less blockage. The significant blockage reduces the amount of useful information that appears in dashcam images.
3) The ground view images used in existing CVM studies were typically taken in good lighting conditions. Nevertheless, real-world driving can happen in poor lighting conditions such as rainy and night time. Dashcam images taken under such complicated lighting conditions can significantly degrade the matching accuracy.
4) The existing CVM schemes have been designed to utilize satellite images as references to localize ground view images. However, a satellite image usually covers a large area of thousands of square feet. It cannot provide the sufficiently precise localization needed in GPS spoofing detection for transportation systems.
5) The geo-tagged reference images are usually updated slowly; e.g., Google Street View is usually updated every few years. Hence there can be a significant time/season gap between the reference and dashcam images.

In this paper, we propose novel schemes to address the above-mentioned challenges for dashcam image localization in real-world driving and design a framework for GPS spoofing detection, termed ***SE****quential dashcam based v****E****hicle localization framewor****K*** (SEEK) for GPS spoofing detection: (1) To address the lack of panorama view, we propose a trip-level sequential image matching scheme for localization, i.e., group a sequence of dashcam images on a short trip together as a single unit for localization. The proposed sequential approach takes advantage of the spatial correlation of dashcam images and also effectively addresses the varying driving behavior in real world driving. (2) To address the second challenge, we propose a scheme to effectively remove unwanted objects or blockage in dashcam images, to significantly reduce blockage and increase the amount of useful information in dashcam images. (3) To address the third challenge, we propose schemes to address the complicated lighting conditions of dashcam images to improve the performance of localizing them. (4) To address the fourth challenge, we adopt the *Google Street View* (GSV) database as geo-tagged reference images because it provides a higher localization accuracy for a location of interest (within 30 feet of a given location), which can meet the precision requirement of GPS spoofing. (5) At last, to address the challenge of the season/theme gap of the images, we propose a scheme to transform the GSV reference images into the same theme as the dashcam images through autoencoders.

In summary, the main contributions of this paper are as follows:

1) We propose a computer vision based GPS spoofing detection framework, SEEK, which utilizes dashcam images to localize the true vehicle location using GSV reference images. SEEK achieves a detection accuracy of up to 94% for a widely used vehicle driving dataset. To the best of our knowledge, this is the first work that utilizes the computer vision technology to detect GPS spoofing attacks.
2) We propose innovative schemes to address significant challenges to use dashcam images for vehicle localization, including the large blockage area in dashcam images, the poor and complicated lighting conditions, lack of panorama view, and the theme gap between reference images and dashcam images.

The rest of the paper is organized as follows. Section II describes the threat model. Section III presents our proposed techniques for dashcam image localization in real-world driving. Section IV put all techniques together to build the SEEK framework. Section V shows the experimental results. Finally, Section VI concludes the paper.

II. Threat Model

We consider two types of GPS spoofing attacks against the GPS based navigation of transportation systems in real-world

driving, 1) ***random GPS attack*** (RGA), and 2) ***sequential GPS attack*** (SGA). RGA broadcasts a series of random (incorrect) GPS coordinates to the target vehicle with the purpose of confusing the GPS reception in a short period of time. RSA can be launched easily at a road intersection or in the middle of a road to cause a denial-of-service to the navigation system, which may result in catastrophic consequences to the victim vehicles, such as driving on the opposite lane or taking wrong turns, etc.

SGA broadcasts a series of GPS coordinates that form a fake trajectory with the goal of either *hijacking* the vehicle to an unsafe location or *escaping* the tracking of the monitoring center of the vehicle, such as the ride-hailing or commercial trucking company monitoring center. To launch the GPS spoofing attack for vehicle hijacking, an external attacker can tailgate a target vehicle and fool its GPS navigation to a route toward an unsafe location. Meanwhile, a GPS spoofer can also be an "escape" driver of a ride-hailing vehicle or a commercial truck, who broadcasts a set of GPS coordinates forming a fake route to the GPS tracker on the vehicle, with the purpose of hiding the real trajectory of the driver from the monitoring center [3]. Compared to RGA, SGA is a more advanced and complicated attack that can craft a continuous spoofing trajectory to mislead the navigation system of a vehicle. But it requires prior planning of a fake trip, such as start and end points and a route following the city road networks, to cheat the monitoring center, or an existing IMU-based GPS spoofing detection scheme.

III. Proposed Techniques for Dashcam Image Localization in Real World Driving

As discussed in Section I, SEEK utilizes dashcam images to identify the location of a vehicle to detect GPS spoofing. We have also introduced CVM and the challenges of applying CVM directly on dashcam images for geo-localization. In this section, we first conduct some experiments of applying CVM on dashcam images and illustrates the challenges for dashcam image localization. We then present our proposed schemes to address each challenge, to significantly improve dashcam image localization. In the next section, we put those techniques altogether to build the SEEK framework.

The dashcam images in our experiments are from a vehicle driving dataset called BDD**G**4k, with details described in Section V-A1. BDD**G**4K provides a geo-tagged GSV reference image for each dashcam image captured from four thousand driving trajectories. As introduced in Section I, in CVM, the image localization or image matching is through image retrieval. Specifically, given a query image, in order to identify the location of this image, CVM scans the entire reference image database where all images are geo-tagged and returns the top-matched reference image. The location of the returned top image is then used to be the location of the query image.

Due to its dependence on image retrieval, the widely used performance metric to evaluate CVM is the recall accuracy, R@k, which treats it as a success if, among the k nearest reference images returned, one is from the original image pair of the query image. For instance, in our scenario, the dataset BDD**G**4k pairs each dashcam image with a (geo-tagged) GSV image. For a query dashcam image, CVM scans all GSV images in the BDD**G**4k dataset. If the returned top image is originally paired with the query dashcam image, then it is treated as a success.

Table I shows the results of applying state-of-the-art CVM methods on a widely used CVM dataset, CVUSA, and the vehicle driving dataset BDD**G**4k. Note that CVM methods usually apply the polar transformation on the satellite images in CVUSA, which is not feasible for the BDD**G**4k dataset. We observe that two recent CVM methods, TransGeo [33] and L2LTR [30], achieve very impressive R@1 accuracy, 94.08%, and 91.99 %, respectively, on the CVUSA dataset. However, their R@1 performance drops significantly to 46.4% and 52.43% on the BDD**G**4k dataset.

TABLE I: CVM performance on CVUSA and BDD**G**4k datasets by state-of-the-art CVM methods.

Method	CVUSA		BDD**G**4k	
	R@1	R@5	R@1	R@5
CVM-NET [29]	22.47	49.98	10.52	21.45
SAFA [31]	81.15	94.23	35.15	38.42
L2LTR [30]	91.99	98.27	46.4	66.1
TransGeo [33]	94.08	98.36	52.43	65.06

As briefly described in Section I, there are several factors that cause the performance degradation of CVM on the BDD**G**4k dataset, including lack of panorama views, blockage in dashcam images, complicated lighting conditions, and seasonal/theme gap. Next, we discuss each of them in details and present our proposed schemes to address them.

A. Trip Level Matching to Address Lack of Panorama View and Impact of Slow Driving

In Section I, we have discussed the issue of the lack of a panorama view in dashcam images, which is critical to achieving good performance in traditional CVM. Moreover, there is another unique issue raised in real world driving – there can be multiple *similar* images collected from a set of nearby locations when the vehicle drives at a low speed or at a complete stop. In this case, a nearby GSV reference image with a different location ID may be returned by CVM when it is applied to BDD**G**4k. When such a result is returned, it is considered a failure in the R@1 performance metric, as it represents a different GPS location, although it is very close to the true location of the query image.

To illustrate how the slow driving behavior impacts CVM performance, we consider a top-M (Meter) metric, which treats it a success if the returned image is within M meters of the query image, instead of requiring the GPS location IDs be the same for the two images as in Table I. Table II illustrates the top-M matching accuracy when M is set to 10 meters, 20 meters, and 50 meters, respectively. We notice the accuracy of the best-performing method, TransGeo, improves from 52.43% to 68.83% from R@1 to top-10 meters. This verifies our

TABLE II: The accuracy of image matching on BDD**G**4k dataset by the distance to the query image.

Method	BDD**G**4k		
	M=10m	M=20m	M=50 m
CVM-NET	30.4	45.31	48.3
SAFA	44.2	50.42	56.92
L2LTR	51.3	66.1	72.71
TransGeo	68.83	75.2	80.1

speculation since, with the top-M metric, a returned image from a nearby location may also count as a success without requiring the image at the exact location of the query image.

To address the above challenge, as well as the lack of panorama views, we propose to conduct dashcam image matching at the trip level, which includes a sequence of contiguous images along a vehicle's trajectory. This will mitigate the impact of slow driving on image matching to eliminate the interference introduced by the repeated or nearby images during vehicle stopping or slow driving. It also helps to compensate for the lack of multiple panorama images at a given location. To process an image sequence from a vehicle's trajectory, we utilize the *recurrent neural network* (RNN) to exploit the spatial and temporal dependencies in a sequence. The Long Short-Term Memory (LSTM) [34] and Gated Recurrent Unit (GRU) [35] are two popular variations of RNN. However, we adopt GRU because it requires less GPU memory while achieving comparable performance to LSTM.

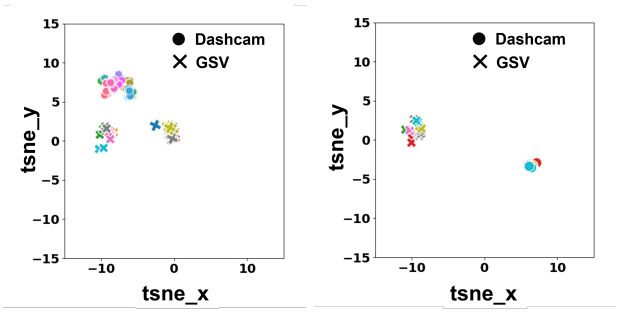
B. Image Normalization to Address Lighting/Weather Conditions

1) Impact of Lighting Condition on Image Matching: We have found that the complicated lighting/weather condition is one of the main factors that affect the performance of CVM on driving datasets. To illustrate its impact, we sort the trips in BDD**G**4k into four groups, Sunny, Cloudy, Rainy, and Dark/Night, based on lighting and weather conditions.

To examine the impact of lighting conditions on the images, we first randomly select ten trips under the Sunny condition and ten trips under the dark lighting condition. Then we use the feature extractor of a recent CVM method, TransGeo [33], to obtain the feature description for both dashcam images and GSV reference images corresponding to the trips. The t-SNE visualization of the feature vectors of those images is plotted in Fig. 2. We can see that the dashcam images and GSV images have quite different clustering behaviors under different lighting/weather conditions. Note that GSV images are always captured in good lighting conditions. In contrast,

TABLE III: Trips distribution under various lighting/weather conditions.

	Sunny	Cloudy	Rainy	Night	Total
Train set	1409	247	68	1276	3000
Val set	475	120	22	383	1000
Total	1884	367	80	1659	4000



(a) (b)

Fig. 2: Impact of lighting condition on the images illustrated in the feature space, a) images randomly sampled from the daylight condition, and b) images randomly sampled from the dark/night condition.

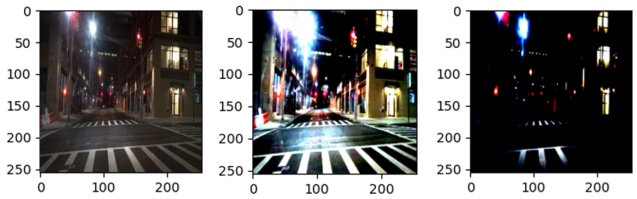
dashcam images are captured in diverse lighting conditions depending on the trips. When the lighting is dark (Fig. 2(b)), dashcam images are clustered together and far away from the GSV reference images. When the lighting is good (Fig. 2(a)), we notice the distance between the GSV images cluster and the dashcam images cluster is smaller. Due to the shorter distance, the matching between dashcam images and GSV images of the daylight trips is more likely to succeed than for the trips in the dark/night condition.

2) Normalization for Low-Light Images: We propose to use the image normalization technique to address the impact of lighting/weather conditions and improve the performance of image matching. Image normalization is a technique in computer vision that changes the range of pixel intensity values to a certain range, e.g., 0 to 1. In the literature, most studies adopt the mean and standard deviation (std) of ImageNet for normalization, i.e., mean = [0.485, 0.456, 0.406] in the three channels (RGB), and std = [0.229, 0.224, 0.225].

TABLE IV: Mean and Std of dashcam and GSV images under different lighting conditions.

Lighting Condition	View Source	Mean	Std
Mixed Daylight	Dash	[0.3383, 0.3688, 0.3790]	[0.2493, 0.2600, 0.2694]
	GSV	[0.5228, 0.5431, 0.5502]	[0.2204, 0.2261, 0.2554]
Night	Dash	[0.1380, 0.1113, 0.0944]	[0.1579, 0.1420, 0.1335]
	GSV	[0.5293, 0.5488, 0.5538]	[0.2163, 0.2213, 0.2554]

However, due to the unique features of the images caused by the diverse lighting/weather conditions in real-world driving, the images from the vehicle driving dataset have quite different mean and std from ImageNet. Table IV illustrates the mean and std of the images in a widely known driving dataset, Berkely BDD100K [36], under daylight (mixed weather) and night condition, which are significantly different from the mean and std of ImageNet. In addition, GSV images have similar mean and std in both daylight and night trips as they are actually independent of the trips and all taken at good lighting conditions (and different times) at the GPS coordinates of the trips. Dashcam images have a significantly lower mean/std



(a) (b) (c)

Fig. 3: Normalization of an image in BDD**G**4k, a) the raw image that was captured at a low lighting condition, b) the normalized image with the mean and std from Table IV, and c) the normalized image with mean/std of ImagetNet

in corresponding lighting conditions. The Berkely BDD100K driving dataset is a highly diverse driving dataset. We expect their mean and std are typical for vehicle driving datasets. Hence we will adopt this mean and std for the normalization of the images in our driving dataset.

Fig. 3 illustrates an example of image normalization for a dashcam image from BDD**G**4k, using the mean and std of ImageNet and the ones of Berkely BDD100K. We can see that the normalized image in Fig. 3 (b) has a much higher contrast; the buildings/landmarks and the crosswalk pattern on the road are highlighted, which are crucial to improve performance. In contrast, the original image captured in the low lighting condition loses the detailed texture and information of buildings and landmarks. At last, the normalized image with ImageNet mean/std exhibits poor performance due to the significant difference in the capturing context of the images.

C. Season Alignment of Reference Images

Due to the fact that GSV reference images are slowly updated, typically after several years, the GSV image at the same location of a dashcam image is often taken at a different time/season, i.e., they have quite different themes. To bridge the gap in the seasonal change on these two views, we propose a season alignment technique to transform the GSV images. We use two autoencoders to import the latent view from the dashcam image into the GSV reference image to mimic the theme/season of the dashcam image, while still producing a realistic transformed image. This improves the matching performance between the two views under various season changes.

Fig. 4 illustrates the process of transferring season features between two views. Two CNN-based autoencoders are trained to learn the hidden style features from dashcam and GSV images, respectively. The latent view is the encoded feature vector that can be used to reconstruct the same image. For the dashcam view, we use the dashcam decoder to decode the latent view to reconstruct its own image in a smaller size without the loss of critical information. In the decoding process of the GSV reference view, we replace the latent view from the GSV image with the one from the dashcam image to reconstruct the GSV reference image. The output of the reference decoder has a smaller size and, most importantly,

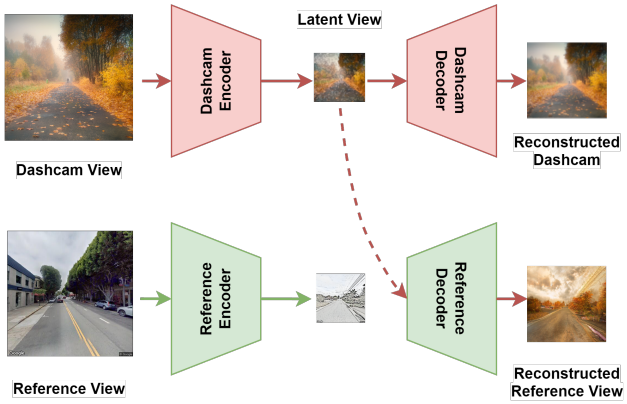


Fig. 4: Season alignment for GSV reference images.

combines the styles from both views and essentially aligns the GSV image to the same season as the dashcam image.

D. Dashcam Image Blockage Removal

In a real-world driving scenario, the dashcam view can be easily blocked by the front vehicle, objects on the windshield, or the reflection of items on the dashboard. Therefore, compared to the GSV images, which are sanitized after capturing, dashcam images usually have a much higher blockage ratio.

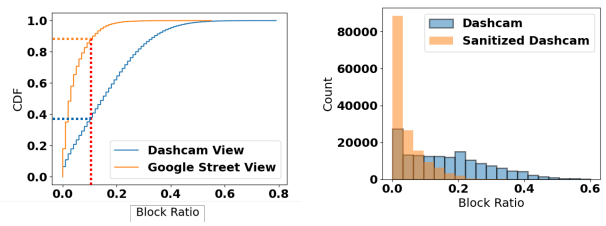
We define the blockage ratio of an image as follows:

$$BlockageRatio = \frac{\text{\# of pixels classified as ``obstacles''}}{\text{Total \# of pixels}}. \quad (1)$$

(a) (b) (c)

Fig. 5: Blockage removal: a) original dashcam image showing segmentation, (b) dashcam image after blockage removal, (c) GSV reference image at the same location.

To summarize the composition of the images captured while driving, we apply the image segmentation technique [37] to annotate each pixel in both the dashcam image and the corresponding GSV reference image. Fig. 5 shows a dashcam image and the corresponding GSV image taken at the intersection where the target vehicle is waiting behind a silver SUV. The blue overlay covers the pixels classified as "obstacles". With the assistance of image segmentation, we calculate the blockage ratios for both the dashcam image and GSV reference image at the same location using (1). As a result, the dashcam view in Fig. 5 has a blockage ratio of 22%, whereas the blockage ratio of the GSV reference image at the same location is only about 2%. Fig. 5(a) shows the CDF



(a) (b)

Fig. 6: Blockage ratio: a) CDF of blockage ratio before blockage removal, b) blockage ratio histogram of dashcam images before and after removing blocking objects.

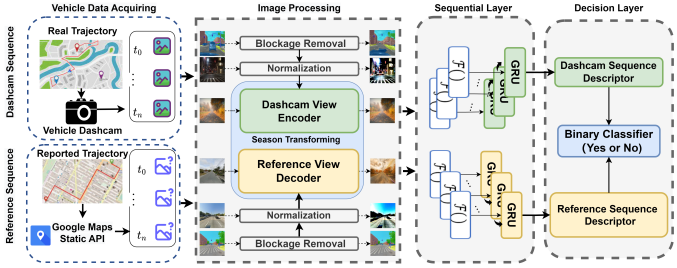


Fig. 7: The architecture of the SEEK framework.

of the blockage ratio of dashcam images and GSV reference images after analyzing the whole dataset, which indicates that more than 90% of GSV images have less than 10% blockage ratio. On the other hand, only 40% of dashcam images have less than 10% blockage ratio.

To address the challenge of large blockage areas in dashcam images, for each dashcam image, we use a tool called *automated object remover* [38] that combines Semantic segmentation and EdgeConnect architectures to remove specified objects in images. By filtering out the objects that are considered "obstacles" such as vehicles, vans, and trucks, we can provide a cleaner image with less blocking area. Fig. 6(b) illustrates the histogram of blockage ratio in the dashcam images before and after blockage removal. Clearly, the blockage in dashcam images can be significantly removed. For instance, in Fig. 5(b), we can see that the silver SUV that blocks the center of the image has been removed, which restored the yellow building at the corner and is expected to improve the dashcam-GSV matching performance.

IV. System Architecture of SEEK

After introducing the proposed techniques for dashcam image localization, in this section, we put them all together to describe the proposed GPS spoofing detection framework SEEK, which is built upon the proposed schemes in the preceding section.

A. System Overview

Fig. 7 illustrates the SEEK framework, which is composed of two Siamese-type pipelines. Each pipeline is composed of the four component schemes we have developed in the preceding section, Image Normalization, Blockage Removal, Season Alignment, and Trip Level matching. The fundamental idea of SEEK is to compare the vehicle's real-time dashcam image sequence with the GSV reference image sequence queried at the GPS locations reported by the vehicle GPS receiver. The two pipeline designs can effectively project the spatial features learned from each pipeline into a shared space. SEEK can be deployed on different platforms to detect possible GPS spoofing attacks. For example, it can be implemented on the vehicle or the remote monitoring center of ride-hailing apps, such as Uber or Lyft. It is carried out by the following steps: 1) a driving vehicle records its GPS location from the GPS receiver, and its dashcam takes video, which is essentially a sequence of images, 2) the GPS locations and the dashcam images are synchronized by the time, 3) the vehicle either sends the GPS locations and dashcam images/video to a local or remote spoofing detector, 4) the spoofing detector queries the GSV reference images by the reported GPS locations using Google API, 5) the spoofing detector feeds both dashcam image sequence and GSV reference image sequence to SEEK and obtains the detection result; *if the binary classifier outputs Yes, i.e., dashcam images match the GSV images, then there is no spoofing, and otherwise, there is spoofing*.

B. Image Transformation

As discussed in the previous section, images need to be properly processed/transformed to address the challenges of complicated lighting conditions, large blockage areas, and season/theme gaps between dashcam and GSV images. Therefore, the three image transformation techniques proposed in the preceding section are applied to the dashcam and GSV images coming into the two pipelines. Specifically, we first normalize the images by using the mean and std extracted from the driving dataset. Then we apply image segmentation [37] to decompose the image and identity the objects in the image that need to be removed. After the unwanted objects have been removed, we further feed the dashcam image into an autoencoder ("Dashcam Encoder") to compress the image into the feature space (encoding), from which it can be reconstructed to the original image (decoding) but has a smaller dimension. The GSV images follow a similar procedure, but it takes the latent space of the dashcam view into the "Reference Decoder" to reconstruct an image with the seasonal theme of the dashcam view as well as a smaller size. After the images are properly transformed by those techniques, they are ready to be fed to the sequential layers for trip level matching.

C. Sequential Trip Level Image Matching

The proposed trip level image matching can effectively utilize the spatial and temporal dependencies among an image sequence of vehicle driving and compensate for the lack of panorama view. We first feed the transformed images to a feature extractor that learns the spatial feature of each individual image in a sequence. Let $\mathbf{x}_n^d$ and $\mathbf{x}_n^g$ ($\mathbf{x}_n^d, \mathbf{x}_n^g \in \mathbb{R}^{|\mathbf{L}_n| \times D}$) denote the output of the feature extractor for a

given transformed dashcam image sequence and GSV image sequence, where D is the embedding size which is 1024 in this paper. They represent the stack of feature vectors from the two image sequences, respectively. In this paper, we test three candidates for the feature extractor: VGG-16 [40], and two feature extractors from L2LTR [30] and TransGeo [33]. Note that CVM models are usually structured in a Siamese-like network. One of the sub-networks can be used as a feature extractor.

SEEK uses a recurrent neural network (RNN) to keep track of the feature representation learned in a sequential manner while the vehicle is in motion. As discussed in the preceding section, SEEK adopts a popular variant of RNN, GRU, due to its more efficient design. SEEK uses a stacked GRU structure with two layers, and the hidden unit is chosen to be 512. The dropout layer [41] and batch normalization [42] are adopted to reduce the internal covariate shift among time steps.

D. Binary Classification Layer

The outputs of the two pipelines of SEEK, $r^d = \mathcal{R}(\mathbf{x}_n^d)$ and $r^g = \mathcal{R}(\mathbf{x}_n^g)$, are the fixed-length vectors that represent the features learnt from each sequence. The final objective is to identify if the series of reference images obtained from the reported GPS locations match the same series of real-time vehicle dashcam images from the same trip. Hence, the last layer of SEEK is a binary classification layer. We adopt the BCELoss defined below as the loss function.

$$\mathbf{L}_{BCE} = -[y_n \cdot \log(\Delta_i) + (1 - y_n) \cdot \log(1 - \Delta_i)], \quad (2)$$

where $i \in \{1, 2, ..., \mathcal{N}\}$, and $\Delta_i = |r_i^d - r_i^g|$ represents the difference between two feature vectors r_i^d and r_i^g, and y_n is the label of the input sequence pair. BCELoss creates a criterion that measures the Binary Cross Entropy between the target and the output. We also add a sigmoid layer in the network to limit the output to a range between 0 and 1. The classification output y_n = 1 indicates the dashcam image sequence and the GSV image sequence are from the same trip. Otherwise, if y_n=0, the two sequences are not from the same trip, i.e., there is a *GPS spoofing attack*.

V. Performance Evaluation

In this section, we carry out experiments to evaluate the performance of the proposed SEEK framework on GPS spoofing detection. We first introduce the dataset we use for the experiments, then discuss the experiment settings, and at last present the results.

A. Dataset

1) Vehicle Driving Dataset: In the literature, a widely used vehicle driving dataset is the Berkley diverse driving video database, BDD100K [36], which consists of 100,000 vehicle driving videos from diverse locations under different weather conditions and different times of the day. Each video records a driving trajectory of about 40 seconds long, 720p, and 30 fps. The videos also include GPS/IMU information to show approximate vehicle driving trajectories.

As can be seen from the SEEK architecture, unlike CVM, SEEK does not utilize image retrieval from the reference image database for localization of the query image. Instead, SEEK compares the query image (sequence) and the reference image (sequence), and finds if they match each other to detect GPS spoofing attacks. Hence, in addition to the dashcam images in BDD100K, we also need the corresponding GSV reference images at the reported GPS locations of the dashcam images. To this end, we expand the BDD100K dataset by adding the GSV reference images.

Specifically, for the dashcam images, we sample the driving video clips in BDDK100k [36] at the sampling rate of 1 Hz, which matches the sampling rate of the GPS information in BDD100K. Then we use the Google Street-View (GSV) Static API [43] to obtain the geo-tagged GSV reference image roughly aligned with the GPS location of each dashcam image in the same heading direction. Note that both the alignment of a dashcam image with the GPS location recorded by cell-phones and the alignment of a GSV image to this GPS location are approximate. Usually, the closest GSV images are a few meters or a few tens of meters away from the referenced GPS location. We have sampled 4 thousand video clips from the BDD100k dataset, i.e., 4000 trips. Each trip lasted about 40 seconds, as described earlier. In total, from those trips, we obtain 152,667 image pairs with various weather and lighting conditions. We call the resulting dataset as BDD**G**4k.

2) Spoofing data generation: The BDD100K dataset does not have GPS spoofing samples. We also have not found other practical public driving datasets with GPS spoofing samples. To bridge this gap, we manually generate GPS spoofing data samples for BDD**G**4k. Given different GPS spoofing attack models, the generated GPS spoofing samples need to accurately represent the attack behaviors based on the existing sequences in BBD**G**4k. We introduce a parameter $\alpha \in [0, 1]$ to describe the strength of the GPS spoofing attack or the percentage of spoofed GPS coordinates in a trip. When $\alpha = 0$, there is no spoofing attack at all. If $\alpha = 1$, then all GPS coordinates in a trip trajectory are spoofed by the attacker.

In our driving dataset, a trip includes both dashcam images and the recorded GPS locations. Given a trip j, we use different approaches to generate a GPS spoofed trip that simulates RSA and GSA, respectively. To begin with, let the number of GPS coordinates to be spoofed be $N_{sp} = \lceil \alpha \times \mathcal{L}_j \rceil$, where α is described above, and $\mathcal{L}_j$ denotes the length of trajectory j. Then we generate faked GPS locations to replace the N_{sp} GPS locations of trip j and, accordingly, update the GSV images for those spoofed GPS locations. In RSA, N_{sp} GPS locations are randomly selected from trip j and are replaced with random GPS points, i.e., there is no relationship between the spoofed GPS locations. However, in SGA, we replace the last N_{sp} GPS points in trip j using a segment of continuous GPS points from a random trip (that trip needs to be longer than N_{sp}). With both approaches, we generate a spoofing trip j'. *The GSV and dashcam image sequences of trip j' form a negative pair*. Note that the dashcam images of trip j' are the same as the original trip j. To eliminate any bias introduced by the imbalanced

TABLE V: Performance of SEEK compared with DeepPOSE in different lighting conditions.

Method	Attack	Sunny				Cloudy				Rainy				Dark			
		Acc	Precision	Recall	F1	Acc	Precision	Recall	F1	Acc	Precision	Recall	F1	Acc	Precision	Recall	F1
SEEK	RGA	**0.940**	0.923	0.960	**0.941**	0.915	0.895	0.940	0.917	0.827	0.816	0.844	0.830	0.852	0.798	0.942	0.864
	SGA	0.920	0.894	0.953	0.923	0.890	0.860	0.931	0.894	0.808	0.803	0.816	0.809	0.825	0.780	0.906	0.838
DeepPOSE	RGA	0.828	0.775	0.923	0.843	0.828	0.780	0.913	0.841	0.827	0.780	0.910	0.840	0.827	0.775	0.920	0.841
	SGA	0.785	0.743	0.869	0.801	0.780	0.750	0.839	0.792	0.800	0.754	0.888	0.816	0.780	0.742	0.856	0.795

data in the training process, we generate the same number of negative samples as positive samples. Here *a positive sample is a pair of dashcam and GSV image sequences of a trip without spoofing*. To guarantee the generality of the model working in various lighting conditions, we balance the positive and negative pairs with the same number of samples from various lighting conditions. We generate 10k positive pairs and 10k negative pairs for each experiment setting.

B. Experimental Settings

1) Implementation Details: SEEK is implemented in PyTorch [44]. Both dashcam and reference GSV images have the original size of 256x256 and are resized to 128x128 after performing season alignment. The model is trained and evaluated on NVIDIA A6000 GPU with 64GB GPU memory. A two-step training process is adopted. We first train the feature extractor on BDD**G**4k dataset. Then, we fix the feature extractor and train the entire system of SEEK. The batch size is 32, and the Adam optimizer is used with a learning rate of 0.0001 based on the cosine scheduling. The whole training process takes 150 epochs, in which 50 epochs are used to train the feature extractor first, and the rest 100 epochs are used to train the entire SEEK system for GPS spoofing detection.

2) Evaluation Metrics: We use the widely used standard performance metrics for classification, accuracy, precision, recall, and F1 score. As a comparative study, we compare SEEK with DeepPOSE [22], which is the only other machine learning based approach that can be applied to the Berkeley BDD100K dataset. Other previous GPS spoofing detection studies [19], [45] use totally different approaches that are not comparable and rely on different assumptions, such as requiring multiple GPS receivers per system and/or ground-based sensors/infrastructure, which are not applicable in the scenario of the BDD100K vehicle driving database we adopt.

C. Performance of GPS Spoofing Detection

1) Performance under different lighting conditions: Table V illustrates the classification results of the proposed GPS spoofing detector, SEEK, including accuracy, precision, recall, and F1 score under different lighting/weather conditions, with the attack strength $\alpha = 1$, trip length $\mathcal{L} = 20s$, and the feature extractor from TransGeo. The classification accuracies of SEEK for RGA and SGA are 94% (F1 score: 0.941) and 92% (F1 score: 0.923), respectively, under the sunny lighting/weather condition. The performance decreases if the lighting condition degrades. For example, under the dark lighting condition, the detection accuracy drops to 85% for RGA, and 82% for SGA. Nevertheless, SEEK outperforms DeepPOSE under all lighting/weather conditions. One may note that the performance of DeepPOSE is similar under all lighting conditions. This is because DeepPOSE utilizes the data from motion sensors which are not affected by the weather/lighting. One can also observe that the performance of SEEK for SGA is only slightly lower than the one for RGA. This demonstrates SEEK is robust to detect smart GPS spoofing attacks such as SGA which carefully crafts a spoofing trajectory that follows real road networks, speed limits, etc., and hence is actually a legitimate trajectory.

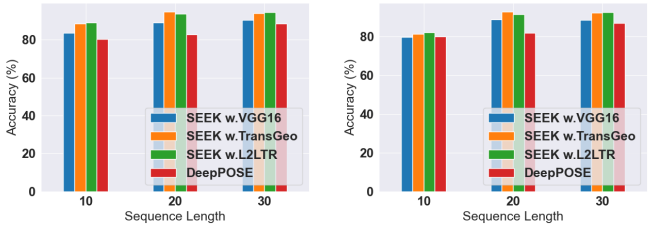
TABLE VI: Performance impact of different image processing techniques/schemes

Schemes	Attack	Sunny				Dark			
		Acc	Precision	Recall	F1	Acc	Precision	Recall	F1
TL only	RGA	0.723	0.719	0.733	0.726	0.601	0.599	0.612	0.605
	SGA	0.711	0.708	0.717	0.713	0.596	0.595	0.602	0.598
TL + NR	RGA	0.803	0.783	0.840	0.810	0.723	0.712	0.750	0.730
	SGA	0.770	0.772	0.767	0.769	0.702	0.687	0.742	0.713
TL + BR	RGA	0.815	0.784	0.870	0.825	0.680	0.661	0.738	0.697
	SGA	0.795	0.767	0.847	0.805	0.658	0.641	0.715	0.676
TL + SA	RGA	0.802	0.782	0.836	0.808	0.710	0.693	0.755	0.722
	SGA	0.768	0.751	0.803	0.776	0.703	0.685	0.753	0.717
All together	RGA	0.940	0.923	0.960	0.941	0.852	0.798	0.942	0.864
	SGA	0.920	0.894	0.953	0.923	0.825	0.780	0.906	0.838

2) Performance impact of individual components of SEEK: As discussed in Section III, to overcome the challenges encountered in real-world driving scenarios, we have proposed four schemes to transform the images, Trip Level matching (TL) of dashcam images, Image Normalization (NR), Blockage Removal (BR), and Season Alignment (SA). We present how each of these proposed schemes improves the performance of GPS spoofing detection. Table VI shows the results of applying those schemes. In the table, the scheme "TL only" means we only use the trip level matching. That is, we treat a sequence of images on a trip in the last $\mathcal{L} = 20$ seconds as a basic unit for classification. As discussed in Section III, the image matching accuracy using a single image is around 52%, which would be closely relevant to the GPS spoofing detection accuracy. On the other hand, Table VI indicates that using the TL scheme significantly increases the performance, with accuracy above 70% in the sunny weather/lighting condition and about 60% even in the dark lighting condition.

Adding the NR (image normalization) technique to the TL scheme further improves the performance. For instance, in the low-lighting (dark) condition, the accuracy improves 12.2% for RGA and 10.7% for SGA, respectively. The table also indicates that NR brings higher performance lift for detection in the dark lighting condition than in the better lighting condition.

In contrast to the NR technique, the BR (blockage removal)



(a) (b)

Fig. 8: Accuracy of SEEK with regard to sequence lenth $\mathcal{L}$: (a) for RGA, and (b) for SGA. $\alpha = 1$.

technique brings a better improvement for the GPS spoofing detection in good lighting conditions. For instance, it improves the accuracy by 7.9 % and 6.1% (TL+BR vs. TL only) in low-lighting conditions for RGA and SGA, respectively, whereas the accuracy improvement is 9.2% and 8.4%, respectively, in the sunny weather. This is because, in the latter case, the objects in an image are easier to be identified, and thus removed. Therefore, the quality of the image is critical before applying block removal. Thus this observation guides the design of SEEK, i.e., applies BR after NR in SEEK as shown in Fig. 7.

Table VI also shows that the SA (season alignment) technique also effectively improves the performance of SEEK in both low lighting and good lighting conditions. In the former case, the accuracy of SEEK is improved by 10.9% and 10.7% for RGA and SGA, respectively, while in the latter case, the accuracy is improved by 7.9% and 5.7%, respectively. At last, as a result of applying all four techniques, TL, NR, BR, and SA, the generalization ability of SEEK is significantly improved to address various lighting/weather conditions, seasonal changes, and blockage to dashcam images.

3) Impact of feature extractor and sequence length: Fig. 8 shows the performance of SEEK in terms of the sequence length ($\mathcal{L}$), under the good lighting condition. We also examine the performance of SEEK under three different feature extractors, VGG16 and two additional ones from L2LTR [30], and TransGeo [33], respectively. Among the three feature extractors, the ones from L2LTR and TransGeo have slightly better performance than VGG16. While L2LTR and TransGeo feature extractors result in a similar performance, the latter is slightly preferred as it uses less GPU memory. With regard to the sequence length, Fig. 8 indicates that a longer sequence length improves the performance. SEEK approximately reaches the maximum accuracy when the sequence length is 20 seconds. A longer sequence of 30 seconds can slightly improve the performance, but it also makes the neural network more complicated, taking more time to train the model. At last, one can observe that in all scenarios, SEEK outperforms DeepPOSE. Note that the sequence length, or trip length, also affects DeepPOSE, which utilizes the motion sensor data over a period of time to detect GPS spoofing.

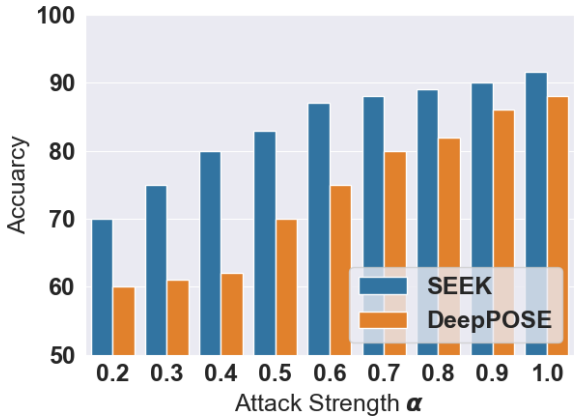


Fig. 9: Accuracy of SEEK compared with DeepPOSE v.s. the attack strength α, with $\mathcal{L} = 20$, TransGeo feature extractor.

4) Resistance to Stealthy GPS Attacker: Next, we test the performance of SEEK with the presence of stealthy attackers who do not spoof the GPS signal until the vehicle drives on a route that is similar enough to the spoofing route. To simulate the stealthy GPS spoofing attack, we control the attack strength α and change it from 0.2 to 1, which indicates the fraction of the number of GPS locations in a vehicle trajectory to be spoofed. Fig. 9 shows the accuracy of SEEK with different values of α, assuming the sequence length $\mathcal{L} = 20s$. From the figure, when the attack strength is higher, the detection accuracy of SEEK is also higher. When the attack strength is lower, i.e., the target vehicle is lightly attacked, the detection accuracy decreases. Nevertheless, a low attack strength would also likely not succeed in achieving the objective of the attackers, such as hijacking a target vehicle. We also compare SEEK with DeepPOSE in Fig. 9. It is clear SEEK outperforms DeepPOSE, especially when the attack strength is lower.

VI. Conclusion

In this paper, we have proposed a novel computer vision based framework for GPS spoofing detection, SEEK, which utilizes dashcam images for vehicle geo-localization and hence detects GPS spoofing attacks. To address the challenges raised by real world driving, such as the lack of panorama view, complicated lighting conditions, large blockage in dashcam images, and seasonal changes, we have proposed multiple novel schemes to transform the images, including trip level (TL) image matching, image normalization (NR), blockage removal (BR), and season alignment (SA). We have extensively evaluated the performance of SEEK compared with a recent GPS spoofing detection scheme DeepPOSE in various settings. The experiment results indicate that SEEK effectively detects GPS spoofing and significantly outperforms DeepPOSE. It achieves a detection accuracy of up to 94% in the good lighting condition.

REFERENCES

[1] K. Wang, S. Chen, and A. Pan, "Time and position spoofing with open source projects," *Black Hat Europe*, vol. 148, pp. 1–8, 2015.

[2] C. L. Krishna and R. R. Murphy, "A review on cybersecurity vulnerabilities for unmanned aerial vehicles," in *Proc. IEEE SSRR*, 2017.

[3] S. Narain, A. Ranganathan, and G. Noubir, "Security of GPS/INS based on-road location tracking systems," in *IEEE S&P*, pp. 587–601, 2019.

[4] K. C. Zeng, S. Liu, Y. Shu, D. Wang, H. Li, Y. Dou, G. Wang, and Y. Yang, "All your GPS are belong to us: Towards stealthy manipulation of road navigation systems," in *Proc. 27th USENIX security*, 2018.

[5] Q. Luo, Y. Cao, J. Liu, and A. Benslimane, "Localization and navigation in autonomous driving: Threats and countermeasures," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 38–45, 2019.

[6] S. P. Arteaga, L. A. M. Hernández, G. S. Pérez, A. L. S. Orozco, and L. J. G. Villalba, "Analysis of the GPS spoofing vulnerability in the drone 3DR Solo," *IEEE Access*, vol. 7, pp. 51 782–51 789, 2019.

[7] M. L. Psiaki, T. E. Humphreys, and B. Stauffer, "Attackers can spoof navigation signals without our knowledge. Here's how to fight back GPS lies," *IEEE Spectrum*, vol. 53, no. 8, pp. 26–53, 2016.

[8] S. Shane and D. E. Sanger, "Drone crash in Iran reveals secret us surveillance effort," *The New York Times*, vol. 7, 2011.

[9] K. C. Zeng, Y. Shu, S. Liu, Y. Dou, and Y. Yang, "A practical GPS location spoofing attack in road navigation scenario," in *Proc. 18th ACM HotMobile*, 2017.

[10] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful GPS spoofing attacks," in *Proc. 18th ACM CCS*, 2011.

[11] Y. Bardout, "Authentication of gnss position: An assessment of spoofing detection methods," in *Proc. 24th ION GNSS*, 2011.

[12] B. W. O'Hanlon, M. L. Psiaki, J. A. Bhatti, D. P. Shepard, and T. E. Humphreys, "Real-time GPS spoofing detection via correlation of encrypted signals," *NAVIGATION: Journal of the Institute of Navigation*, vol. 60, no. 4, pp. 267–278, 2013.

[13] K. Wesson, M. Rothlisberger, and T. Humphreys, "Practical cryptographic civil GPS signal authentication," *NAVIGATION: Journal of the Institute of Navigation*, vol. 59, no. 3, pp. 177–193, 2012.

[14] D. M. Akos, "Who's afraid of the spoofer? GPS/GNSS spoofing detection via automatic gain control (AGC)," *NAVIGATION: Journal of the Institute of Navigation*, vol. 59, no. 4, pp. 281–290, 2012.

[15] K. Jansen, N. O. Tippenhauer, and C. Pöpper, "Multi-receiver GPS spoofing detection: Error models and realization," in *Proc. 32nd ACM ACSAC*, 2016.

[16] M. R. Manesh, J. Kenney, W. C. Hu, V. K. Devabhaktuni, and N. Kaabouch, "Detection of GPS spoofing attacks on unmanned aerial systems," in *Proc. 16th IEEE CCNC*, 2019.

[17] E. Shafiee, M. R. Mosavi, and M. Moazedi, "Detection of spoofing attack using machine learning based on multi-layer neural network in single-frequency GPS receivers," *The Journal of Navigation*, vol. 71, no. 1, pp. 169–188, 2018.

[18] M. Sun, Y. Qin, J. Bao, and X. Yu, "GPS spoofing detection based on decision fusion with a k-out-of-n rule," *Int. J. Netw. Secur.*, vol. 19, no. 5, pp. 670–674, 2017.

[19] K. Jansen, M. Schäfer, D. Moser, V. Lenders, C. Pöpper, and J. Schmitt, "Crowd-GPS-Sec: Leveraging crowdsourcing to detect and localize GPS spoofing attacks," in *Proc. IEEE SP*, 2018.

[20] P. F. Swaszek, R. J. Hartnett, M. V. Kempe, and G. W. Johnson, "Analysis of a simple, multi-receiver GPS spoof detector," in *Proc. ION ITM*, 2013.

[21] P. F. Swaszek, R. J. Hartnett, and K. C. Seals, "Using range information to detect spoofing in platoons of vehicles," in *Proc. 30th ION GNSS+*, 2017.

[22] P. Jiang, H. Wu, and C. Xin, "DeepPOSE: Detecting GPS spoofing attack via deep recurrent neural network," *Digital Communications and Networks*, vol.8, 2021.

[23] J. Farrell and M. Barth, *The global positioning system and inertial navigation*. Mcgraw-hill New York, vol. 61, 1999.

[24] J. Wendel, O. Meister, C. Schlaile, and G. F. Trommer, "An integrated GPS/MEMS-IMU navigation system for an autonomous helicopter," *Aerospace science and technology*, vol. 10, no. 6, pp. 527–533, 2006.

[25] D. Titterton, J. L. Weston, and J. Weston, *Strapdown inertial navigation technology*, vol. 17, 2004,

[26] M. Schäfer, V. Lenders, and J. Schmitt, "Secure track verification," in *Proc. IEEE S&P*, 2015.

[27] D. Moser, P. Leu, V. Lenders, A. Ranganathan, F. Ricciato, and S. Capkun, "Investigation of multi-device location spoofing attacks on air traffic control and possible countermeasures," in *Proc. 22nd ACM MobiCom*, 2016.

[28] S. Zhu, T. Yang, and C. Chen, "VIGOR: Cross-view image geo-localization beyond one-to-one retrieval," in *Proc. IEEE CVPR*, 2021.

[29] S. Hu, M. Feng, R. M. Nguyen, and G. H. Lee, "CVM-NET: Cross-view matching network for image-based ground-to-aerial geo-localization," in *Proc. IEEE CVPR*, 2018.

[30] H. Yang, X. Lu, and Y. Zhu, "Cross-view geo-localization with layer-to-layer transformer," *in Proc. NIPS*, 2021.

[31] Y. Shi, L. Liu, X. Yu, and H. Li, "Spatial-aware feature aggregation for image based cross-view geo-localization," *in Proc. NIPS*, 2019.

[32] S. Hu and G. H. Lee, "Image-based geo-localization using satellite imagery," *International Journal of Computer Vision*, vol. 128, no. 5, pp. 1205–1219, 2020.

[33] S. Zhu, M. Shah, and C. Chen, "TransGeo: Transformer is all you need for cross-view image geo-localization," in *Proc. IEEE CVPR*, 2022.

[34] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[35] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[36] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving dataset for heterogeneous multitask learning," in *Proc. IEEE CVPR*, 2020.

[37] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ADE20K dataset," in *Proc. IEEE CVPR*, 2017.

[38] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi, "EdgeConnect: Structure guided image inpainting using edge prediction," in *Proc. IEEE ICCVW*, 2019.

[39] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. IEEE CVPR*, 2005.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[41] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[42] T. Cooijmans, N. Ballas, C. Laurent, Ç. Gülçehre, and A. Courville, "Recurrent batch normalization," *arXiv preprint arXiv:1603.09025*, 2016.

[43] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver, "Google street view: Capturing the world at street level," *Computer*, vol. 43, no. 6, pp. 32–38, 2010.

[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *in Proc. NIPS*, 2019.

[45] A. Eldosouky, A. Ferdowsi, and W. Saad, "Drones in distress: A game-theoretic countermeasure for protecting uavs against GPS spoofing," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2840–2854, 2019.