Automatic Shadow-Based Satellite Feed Verification

David Koplow
MIT
dkoplow@mit.edu

Andrew Cai
MIT
andcai@mit.edu

ABSTRACT

Optical satellites are one of the most valuable sources of surveillance data, yet there do not seem to be any unclassified methods that verify their data in real time. We propose a pipeline that accomplishes this through the frame-by-frame verification of satellite position, image location, and temporal accuracy of the image. We also present and implement a shadow-based method of automatic single-frame image temporal verification. This model performs well on a dataset of satellite images of major cities, even when images are partially obstructed by weather anomalies and cloud cover. It can accurately approximate the time of day an image was taken within an hour, but is not yet robust enough for military applications.

1 INTRODUCTION

With geopolitical tensions rising and a high potential for large-scale military conflict, reconnaissance on enemies is more important than ever. Optical satellites are a significant source of reconnaissance data [6], with the United States possessing at least 52 optical satellites [9] to monitor and influence conflicts such as the War in Afghanistan [12] and the ongoing Russo-Ukrainian War [10]. Officials and experts warn that cyberattacks against optical satellites are an urgent threat [11], with Global Navigation Satellite System (GNSS) signal spoofing already being performed by Russia [8].

Compromised satellites could return spoofed signals with incorrect position, navigation, and timing (PNT) data or modified visual feeds without critical reconnaissance information. For example, adversaries can produce altered visual feeds that depict a different location or the desired location at a different time. Furthermore, the angle of the satellite camera could be altered or the satellite trajectory could be interfered with. Current safeguards against satellite cyberattacks involve encryption, monitoring, and frequency hops [13]. However, we have found no unclassified techniques for automatic satellite feed verification.

We propose a satellite feed verification system that is automatic, real-time, and transparent. This verification system applies only to images taken between dawn and dusk. Our frame-by-frame macro-pipeline can be broken down into four steps: Verifying PNT data, verifying image metadata, verifying the captured image is taken at the correct time (by shadow analysis), and comparing to other the results of prior frames. The macro-pipeline will return a simple "yes, the satellite is compromised" or "no, the satellite is not compromised".

This work has three key contributions: (1) We introduce the idea of automatic temporal satellite visual feed verification. Despite extensive research, we were unable to find other instances of this problem being addressed or proposed. (2) We outline a complete macro-pipeline that can be used to implement automatic satellite feed verification. (3) We implement and evaluate an almost-automatic micro-pipeline using a shadow-based model to verify that an image from a satellite feed is taken at the correct time given correct metadata and satellite location. Indeed, this model, while only part of our macro-pipeline, is critical to our satellite verification algorithm. While our results are not perfect, they are promising and suggest clear avenues for future research to improve performance.

We have introduced the problem and provide background and motivation in this section. In Section 2, we reference prior work, and in Section 3, we explain the design of our model. In Section 4, we describe our dataset and evaluate our results. Finally, Section 5 is our conclusion and includes source code for our model.

2 RELATED WORK

We take inspiration and apply methods from previous researchers for aspects of our macro-pipeline. The first step of our macro-pipeline regards verifying PNT data: We want to verify the coordinates that the satellite claims to be at. Previous researchers have discovered methods to do so: Kalabic, Weiss, and Chiu have devised a scheme to verify self-reported satellite positions [16]. The second step of our macro-pipeline involves verifying image metadata, particularly the geographical region displayed in the image. There are a few existing techniques for doing this, of which we mention two examples: (1) Deep neural networks can analyze patterns in satellite imagery which can be used to determine where our image was taken by comparison with an existing database. In Terrapattern's implementation [19], a lookup table of features and locations is generated from a training set of tagged images. (2) Tang et. al use an adaptive feature contrast (AdaFC) model in order to account for different weights of each feature in similarity measurement [21].

The third step of our macro-pipeline regards verifying the date and time that a photo was taken by analyzing its shadows. A critical method in this step is shadow isolation

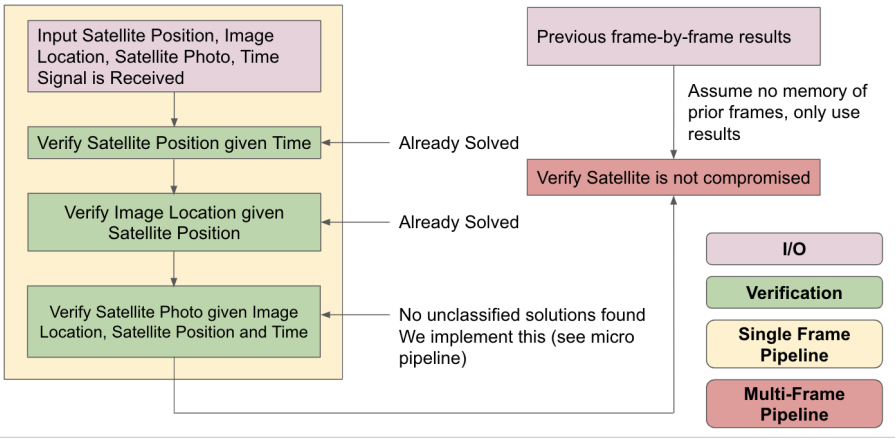

Figure 1: Steps of macro-pipeline used to determine if the satellite is compromised or not.

in satellite images. These use a variety of techniques that are classified into three groups: Machine learning-based research like Khan et. al use CNNs to learn features that represent positions of shadows [18]. Model-based methods focus on geometric modeling of the scene, with Fang et. al's approach focusing on spectral and geometrical models of shadows [5]. Property-based methods, on the other hand, focus on specific features of the image, classifying or segmenting pixels such as in the following two approaches: Salvador et. al use specific low-level features such as the intensity, color, and hue of images to isolate shadows [20]. Silva et. al introduce an effective chromaticity method which uses image segmentation on thresholds determined by features in the image [7].

3 METHODOLOGY

3.1 Overview

Our method aims to independently verify that the satellite is sending correct, real-time information. To do so, we must verify that three pieces of information are not compromised for each image frame: PNT data (satellite position), image metadata (in particular, image location), and the satellite photo itself. We are able to assume that the time that the information was sent is accurate, as a delayed signal would solely result in inaccurate information (which can be deduced from our method).

Our macro-pipeline, as stated earlier, consists of four steps: (1) Verification of satellite position using only the time the signal is received. (2) Verification of image location using the verified satellite position. (3) Verification that the satellite photo is taken at the correct time, given the verified satellite position, verified image location, and time the signal is received. (4) Combining the results received from the previous three steps for the specific frame with results from previous frames in a frame-by-frame, online algorithm. This macro-pipeline is depicted in Figure 1.

For the first step of our macro-pipeline, we verify the satellite position using only the time of the information. We propose the usage of Kalabic et. al's implementation [16]. As stated in Section 2, they have devised a proof-of-location scheme to verify self-reported satellite positions of a constellation with at least three satellites. The second step of our macro-pipeline is to verify the image location given the verified satellite position. We ensure that this desired location matches the true location of the captured image using a method by Tang et. al [21]. They use AdaFC (Adaptive Feature Contrast) on common and distinct features between the captured image and a preprocessed image of the desired location.

The third step of our macro-pipeline is to verify the captured image is taken at the correct time given the verified image location, satellite position, and time. We have not found any unclassified solutions to this step, and implement it in our micro-pipeline, depicted in Figure 2. Our Python-based implementation is almost-automatic (with only one manual operation). This micro-pipeline can be broken down into three segments which are detailed in the following three subsections: Data processing, shadow prediction, and shadow comparison.

The fourth step of our macro-pipeline uses a sequence of boolean results, one from each frame. Given a continuous stream of input data, we find if "No, the satellite is compromised" results are outputted at a significant rate for a consistent reason. After sufficient testing of the first three steps, we determine the false negative rate of each. Then, we examine input data for the last n frames, where n is

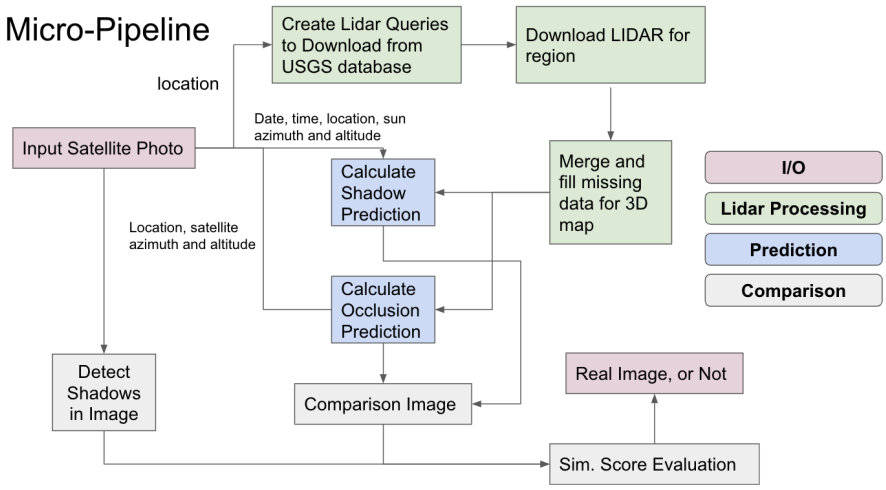

Figure 2: Steps of micro-pipeline used to determine if the satellite image was taken at the correct time.

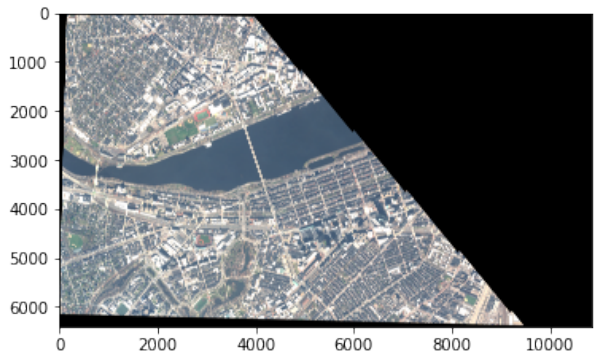

Figure 3: Unedited satellite image of a selected region of Boston and Cambridge.

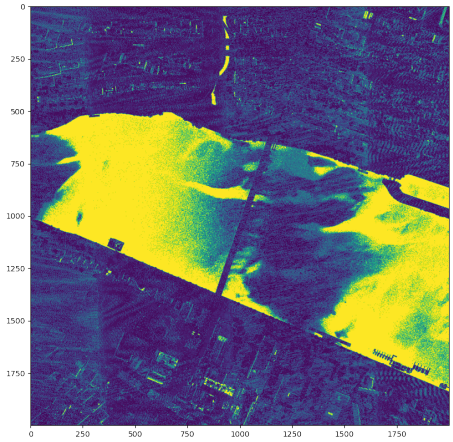

Figure 4: LiDAR data sparsity on a selected area of Boston and Cambridge. Blue and yellow areas represent high and low measurement densities, respectively.

determined after sufficient testing. The proportion of "no" results derived from each step is measured, and compared to the measured false negative rates. If any of these three proportions is statistically significant, then the satellite is completely compromised.

3.2 Data Processing

Our micro-pipeline begins with data processing. The data processing step takes in a satellite image as input and outputs a height map of the area photographed in the satellite image. In addition, it transforms satellite image data into a generalized format.

This step begins with the processing of the satellite image. Each satellite image in our dataset consists of two files: A .json file containing image metadata and a GeoTiff file containing the image itself (such as in Figure 3). We process the GeoTiff file by determining the possible longitudes and latitudes of each point (encoded in the WGS84 standard coordinate system), and storing the RGB representation of the image.

We are now able to retrieve the necessary LiDAR data given our processed satellite image. LiDAR (Light Detection and Ranging) is a remote sensing method that measures the altitudes of points using discrete projections of an aircraft-mounted laser onto points on Earth's surface [1]. Each $1m \times 1m$ square typically receives 15 point measurements, but variance is high [14], as seen in Charles River in Figure 4.

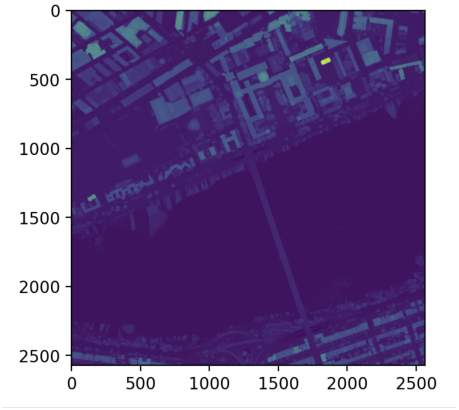

Figure 5: Height map of a selected area of Boston and Cambridge. Yellower areas represent larger Z-values.

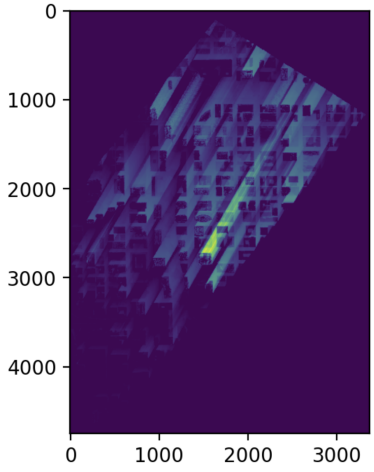

Figure 6: Shadow prediction (without removing occluded areas) on a selected area of Chicago. Yellower areas represent more intense shadows.

LiDAR data is measured in surveys, each in different formats and spanning varying regions. To extract the relevant LiDAR data, we proceed through a three-step process: (1) Extract LiDAR squares that cover the geographical region contained in our captured image. (2) Merge and prune the points in these squares. (3) Transform the data into a usable point cloud format.

We first extract LiDAR squares that cover our captured image. Using the NationalMap API, we are able to search for LiDAR squares that cover a specific point [4]. We iterate through the coordinates of our captured image using the Python geopy library, beginning from the lower left and ending when all areas are covered. Each LiDAR survey encodes its information with a modified coordinate system (CRS) [3], which we must convert into the WGS84 standard coorindate system. While we must manually aggregate them, we could create a database of them after enough data is processed. Besides this search, all steps of our pipeline are automatic. To account for data sparsity and generate a usable height map, we apply a weighted 4-NN. Each point in our area that lacks a measurement receives an “implied height" generated as the distance-weighted average of the heights of its four nearest neighbors, such that the closer a point is, the more it impacts the height value. If the distances for the four points are too large, then the point is not given a measurement.

3.3 Shadow Prediction

The second step of our micro-pipeline is shadow prediction. Given metadata of an image and the region’s height map (as seen in Figure 5), this step outputs the predicted shadows in the resulting image. Our shadow prediction method relies on three steps: (1) Project the position of the Sun onto each point in the map generated in Step 1 to create a shadow map, and project the position of the satellite to create an occlusion map. (2) Find regions that are occluded regions from the shadow map as a result of the scene geometry and position of the satellite. (3) Process both images to remove occluded regions.

We fetch the altitude and azimuth of the sun at a specific date, time, and location using the Python pysolar library, and verify this against image metadata. Using these angles, we project the height of each point of the map onto ground level. We assume projection is orthographic and that the angle of the Sun is constant for all points in an image. A ray can then be drawn between each “projecting point" and its “shadow endpoint". Computationally, this was approximated for each pixel by collecting the height values for the first 2000 pixels in the direction of a sun beam, then comparing the maximum height of an object that would be in shadow at that point to the actual height of the point (as seen in Figure 6).

However, we must also consider occlusion areas: Regions that might be in shadow but are hidden by a building as a result of the angle the satellite is taking an image from. Due to the nature of camera projection, predicted shadows that are in occlusion areas do not typically appear on an image due to the true height of each occlusion area being higher than the predicted height pulled from the height map. Though we could correct for the occlusion shadow boundary, we find that the computational power needed to check which parts of occluded areas will be shadowed is impractical and would only marginally improves our model, as the shadow detection method employed is inaccurate on sides of buildings due to their texture and windows. The occluded areas are calculated in a similar manner as the shadowed areas, and can be viewed in Figure 7. To generate our shadow prediction map, we

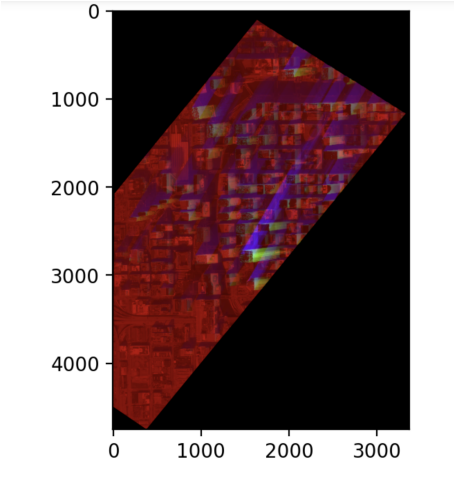

Figure 7: Shadow prediction (in blue) and occlusion prediction (in blue) overlaid with satellite image (in red).

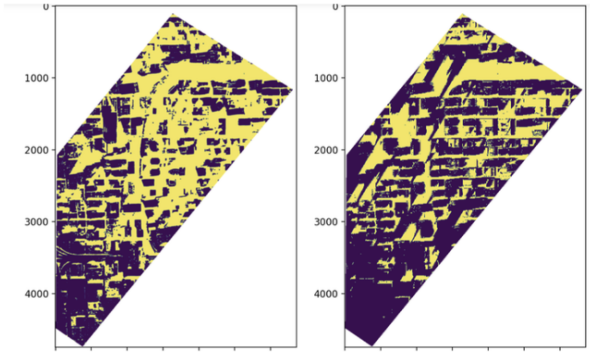

Figure 8: Isolated shadows from satellite image (left) and shadow prediction map (right) of a region of Chicago. Yellow areas represent shadows while black areas represent lack of shadows.

remove the areas in the occlusion prediction regions from the shadow prediction regions.

3.4 Shadow Comparison

The third step in our micro-pipeline is shadow comparison. Given the transformed satellite image data from the data processing step and the shadow prediction map, this step outputs a boolean result determining if the captured image was taken at the correct time. There are three main steps to determining this value: (1) Isolate shadows from the transformed satellite image. (2) Generate a similarity score by comparing these isolated shadows and the shadow prediction map. (3) Determine if the image was taken at the correct time by comparing the previous similarity score with similarity scores generated by comparing isolated shadows with shadow prediction maps at altered times.

We first isolate shadows from the transformed satellite image using an algorithm by Silva et. al [7]. As stated previously, Silva et. al employ a property-based chromaticity method to segment shadows. In our micro-pipeline, we use Hong's implementation of Silva et. al's model (with Otsu's Method of Thresholding replaced by the significantly quicker K-Means clustering) [15].

Next, we compare the isolated shadows and shadow prediction map, as shown in 8. To determine a similarity score, we determine the fraction of shadowed pixels in the shadow prediction map that are also shadowed in the captured image (which must be in the unit interval). Finally, we determine if the satellite image was taken at the correct time. For each image, we feed the shadow prediction step described in Subsection 3.3 eleven pieces of altered data, where the time in the metadata is modified to every hour multiple before and after the unaltered time. Because shadow prediction isn't effective at night, any times before 6AM or after 6PM are disregarded. Thus, we retrieve a shadow prediction map for every hour of the day. We compare each of these shadow prediction maps with the isolated shadows found in the satellite image to produce eleven altered similarity scores. Of the twelve total similarity scores, if the similarity score generated by unaltered data is the highest, then the captured image is verified to be taken at the correct time. Otherwise, the captured image is compromised.

4 EXPERIMENTAL RESULTS

4.1 Evaluation

We evaluate our micro-pipeline, which returns the status of an image based on the similarity score between an image and the projected shadows at different times. The micro-pipeline is evaluated on selected photos from nine large American cities. These urban regions have complex shadows but are also able to return the most precise results given the larger shadow sizes from buildings commonly found in cities. We use a test dataset of 24 images from Planet Labs's Skysat 50cm database [2], but our access plan limits the scope of our evaluation. Each image is checked to be between dawn and dusk, and almost all images are free of significant cloud cover and weather anomalies. Three exceptions are included to test the robustness of our model: Two images of Philadelphia under light cloud cover and moderate cloud cover, and one additional image of Boston covered with snow. While this data-set is small, we were limited by the number of square km we could pull under our free agreement.

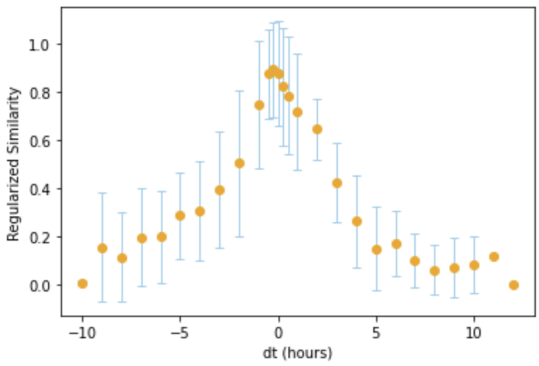

Figure 9: Mean and standard deviations of the regularized similarities at each time difference. Only points with three or more dt values were plotted.

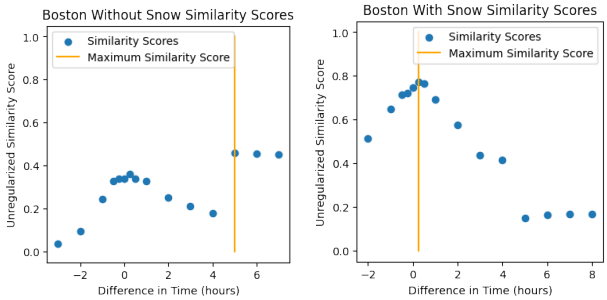

Figure 10: Similarity score vs change in time for Boston without snow cover (left) and Boston under snow cover (right). Blue dots are local similarity scores and orange lines show maximum score.

We assume that our dataset is error-free. Hence, the visual feed, metadata, and PNT information we use will be assumed to be correct. We feed our model these three variables in our 24 unaltered examples. The model's performance on these examples determines the true negative rate, defined as the proportion of non-compromised photos that the model verifies. Recall our metric of comparison: We determine if the similarity score generated with unaltered time data is higher than the similarity scores generated by each altered time. In addition to generating similarity scores with modified times at every hour multiple before and after the unaltered time, we also generate similarity scores of times 15 and 30 minutes before and after the unaltered time. These four "close" times, while not used to determine if a satellite image is compromised, are used to better analyse the accuracy of our model. In no examples where the highest similarity score occurred at a dt within an hour but not 0, was the similarity score at 0 lower than one above an hour. They are measured in addition to the 12 hour-by-hour similarity scores, and are used in the analysis of the regularized similarity score.

We have two quantitative methods to evaluate our results: First, we determine the success rate of our model on the 24 unaltered examples. Due to our metric of comparison, it is not effective to measure the true positive rate (defined as the proportion of compromised photos that the model fails to verify), as introducing random adversarial examples would bias our results. Second, we calculate the regularized similarity score (of both altered and unaltered data). This is performed by assigning the time generating a predicted shadow with the lowest similarity score for an image (of the 16 measured) a regularized value of 0, the highest similarity score a regularized value of 1, and every other datapoint a regularized value between 0 and 1 such that the ratio between its distances to the highest and lowest similarity scores remains invariant. Each similarity score is then assigned a "time difference", define as the difference between the unaltered time that the image was taken and the altered time that produces the specific similarity score. We then aggregate this data for each of the 24 unaltered examples to determine the mean and standard deviations of the regularized similarity scores for each time difference.

4.2 Analysis

Our model is successful on 20 of 24 obstructionless images, giving an 83.3% true negative rate. Out of these 24 images, only one has an error of greater than 1 hour. This datapoint was an outlier due to the failure of the shadow segmentation algorithm, and will be further discussed later.

Our results are statistically significant: We note that the mean error is 0.29 hours and the standard deviation of the error is 0.42 hours. Hence, by using a t-statistic, the 95% confidence interval of the mean error is $[0.108, 0.472]$, ensuring that the average error is likely less than 0.5 hours. Assuming our data follows a normal distribution, this error will be under 1.13 hours 95% of the time.

We also measure the mean and standard deviation of the regularized similarity score for each time difference, as shown in Figure 9. The mean of the regularized similarity score is high and extremely similar for the time differences of 0, -15, and -30 minutes. Furthermore, the mean regularized similarity score for time difference 0 is over 10% greater than the mean regularized similarity score for every other integer hour time difference.

It is important to note that we are unable to use a fixed threshold on similarity score to determine if images are verified. From our examples, the maximum similarity score of an image can range from 5.4% to 86.4%. This difference is due to the different ratio geometries of each satellite image, and

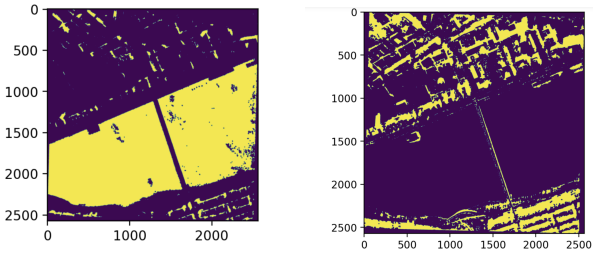

Figure 11: Isolated shadows of an image of Boston without snow cover (left) and Boston under snow cover (right). The yellow areas represent shadows.

can only be corrected by examining regularized similarity scores like our analysis in Figure 9.

Our model is surprisingly robust: it performs well on images with significant cloud cover and weather anomalies. Our image of Philadelphia with light cloud cover has an error of zero, while our image of Philadelphia with heavy cloud cover has an error of 0.25. We conjecture that this is due to our comparison metric, which is robust because no matter the true shadow level, the relative best representation of the shadows will correspond to the relative maximum similarity score. We did not find statistically significant difference in performance against time of day.

Finally, we qualitatively analyze a pair of images that represent our outlier datapoint. First, we note that the image of Boston under snow cover has an error of 0.25, while the image of Boston without snow cover has an error of 5. The isolated shadows for each image can be viewed in Figure 11. The darker Charles River is believed to be completely shadowed by our shadow isolation algorithm in the image without snow cover, but the light snow covering the river in the other image leads the algorithm to believe that there the river is clear of shadows. As shown in Figure 10, the true time was still near a local minimum.

5 CONCLUSION

In this paper, we have devised a method of automatic satellite feed verification that independently verifies each piece of information that an imaging satellite sends. We have also successfully implemented a shadow-based micro-pipeline that verifies that a satellite image was taken at the correct time given a verified image location and satellite position. This micro-pipeline performs quite well given the complexity of the task, resulting in an 83.3% true negative rate and with a mean error in the predicted the time an image was taken of 28 minutes under 95% confidence. It is also quite robust, with predictions accurate to 15 minutes (on small sample size) even when images are obfuscated by cloud cover or snow cover.

While results are promising, our micro-pipeline is not nearly robust enough to be applicable in real-world situations. We currently require LiDAR data in order to generate the shadow prediction map, which can be sparse in foreign areas and doesn't account for new developments. Our micro-pipeline is also simply an approximation; the one-hour precision level makes it possible for signals to be compromised by an adversarial agent with knowledge of the system, as they could retrieve a photo of the location taken at an approximately correct time. Furthermore, the performance of our micro-pipeline on types of satellite images such as rural areas and mountainous terrain is unclear. Finally, our micro-pipeline also only functions on images taken during daylight hours and ones without large bodies of water.

There is a multitude of directions we can pursue for our future steps. Possibly the most imperative is to conduct additional testing of our micro-pipeline: A larger and more robust dataset could more firmly confirm our results. For example, we can test our model on multiple images of the same region at different times, or on multiple image sizes focused on the same region. Testing with smaller time difference intervals, such as 5 to 10 minutes rather than an hour, would allow for tighter bounds on estimates of the time an image was taken. In addition, we can optimize Silva et. al's shadow isolation algorithm to improve the robustness of our model on specific types of images such as those containing bodies of water. Finally, to replace the inconsistent and often-lacking LiDAR data, we may be able to infer the height data of a scene using only its satellite image by employing a U-net architecture following the work of Karatsiolis et. al [17].

The code for our micro-pipeline is available on github.

REFERENCES

[1] 3dep lidarexplorer. https://prd-tnm.s3.amazonaws.com/LidarExplorer/index.html#/. Accessed: 2022-04-26.
[2] High-resolution imagery with planet satellite tasking. https://www.planet.com/products/hi-res-monitoring/. Accessed: 2022-04-26.
[3] Spatial reference list. https://spatialreference.org/ref/epsg/.
[4] Tnmaccess api. https://apps.nationalmap.gov/tnmaccess/#/.
[5] A method to segment moving vehicle cast shadow based on wavelet transform. *Pattern Recognition Letters 29*, 16 (2008), 2182–2188.
[6] 21st century reconnaissance, battlefield key component of maneuver warfare. *Marine Corps Gazette* (2017).
[7] Near real-time shadow detection and removal in aerial motion imagery application. *ISPRS Journal of Photogrammetry and Remote Sensing 140* (2018), 104–121.
[8] Above us only stars, exposing gps spoofing in russia and syria. *C4ADS innovation for peace* (2019).
[9] Ucs satellite database. https://www.ucsusa.org/resources/satellite-database, 2022. Accessed: 2022-04-26.
[10] Barnes, J. E., Cooper, H., and Schmitt, E. U.s. intelligence is helping ukraine kill. *New York Times* (May 4, 2022).
[11] Brooks, C. The urgency to cyber-secure space assets. *Forbes* (Feb 27, 2022).

[12] Campbell, D. Us buys up all satellite war images. *The Guardian* (Oct 17, 2001).

[13] Eddy, M. Want to hack a satellite? it might be easier than you think. *PC Magazine* (2019).

[14] GISGeography. A complete guide to lidar: Light detection and ranging. *GISGeography* (October 29, 2021).

[15] Hong, W., and Wang, T. Shadow detection algorithm for aerial and satellite images. https://github.com/ThomasWangWeiHong/Shadow-Detection-Algorithm-for-Aerial-and-Satellite-Images, 2019.

[16] Kalabić, U., and Weiss, A. Distributed small sat location verification. *Integrated Communications Navigation and Surveillance Conference* (2021).

[17] Karatsiolis, S., and Kamilaris, A. Focusing on shadows for predicting heightmaps from single remotely sensed RGB images with deep learning. *CoRR abs/2104.10874* (2021).

[18] Khan, S. H., Bennamoun, M., Sohel, F., and Togneri, R. Automatic shadow detection and removal from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence 38*, 3 (2016), 431–446.

[19] Levin, G., Newbury, D., McDonald, K., Alvarado, I., Tiwari, A., and Zaheer, M. Terrapattern. https://github.com/CreativeInquiry/terrapattern, 2016.

[20] Salvador, E., Cavallaro, A., and Ebrahimi, T. Cast shadow segmentation using invariant color features. *Computer Vision and Image Understanding 95*, 2 (2004), 238–259.

[21] Tang, H., Gong, A., Li, S., Yi, W., and Yang, C. Similarity measures of satellite images using an adaptive feature contrast model. *International Journal of Geosciences 4*, 2 (2013), 329–343.