2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

UAV Localization Using Autoencoded Satellite Images

Mollie Bianchi and Timothy D. Barfoot

***Abstract*— We propose and demonstrate a fast, robust method for using satellite images to localize an Unmanned Aerial Vehicle (UAV). Previous work using satellite images has large storage and computation costs and is unable to run in real time. In this work, we collect Google Earth (GE) images for a desired flight path offline and an autoencoder is trained to compress these images to a low-dimensional vector representation while retaining the key features. This trained autoencoder is used to compress a real UAV image, which is then compared to the precollected, nearby, autoencoded GE images using an inner-product kernel. This results in a distribution of weights over the corresponding GE image poses and is used to generate a single localization and associated covariance to represent uncertainty. Our localization is computed in 1% of the time of the current standard and is able to achieve a comparable RMSE of less than 3m in our experiments, where we robustly matched UAV images from six runs spanning the lighting conditions of a single day to the same map of satellite images.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) are being used for more and more applications while still remaining largely reliant on GPS. The disadvantage of a GPS-based localization system is that it is susceptible to dropout, jamming, and interference. In GPS-denied environments, the primary sensor becomes vision due to its low weight and fast computation.

Visual Odometry (VO) is commonly used on UAVs but requires corrections to prevent drift. Visual Simultaneous Localization and Mapping (SLAM) is one solution to this issue, but its use on UAVs has primarily been demonstrated in indoor environments or small areas [2]–[4]. One method [5] for long distance autonomous, outdoor, aerial navigation in GPS-denied environments uses the Visual Teach and Repeat (VT&R) method [6]. By generating a locally consistent visual map on an outbound pass under manual or GPS control, the UAV is then able to return autonomously along that path without GPS.

VT&R is limited in that it requires a manual outbound pass and, because it relies primarily on point-feature matching (e.g., Speeded-Up Robust Features (SURF) [7]), the return pass must be completed shortly after the outbound pass so that the lighting conditions along the path have not significantly changed. It does not allow for the repeated traversal of the path using a map generated much earlier.

A unique opportunity for aerial vehicles is that there is an existing database of satellite images covering the entire world in Google Earth (GE). In many areas, these satellite images have been used to generate a detailed 3D reconstruction of a scene from which it is possible to render an image at any desired pose. In [8], the idea was proposed to replace the manual outbound pass in VT&R with a virtual pass in GE.

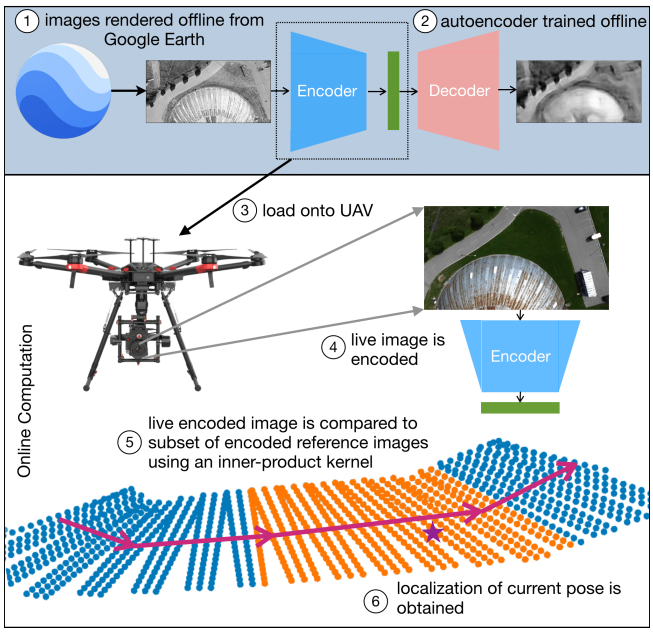


Fig. 1: 1. Offline before flight, images are rendered in a grid pattern around the desired path using Google Earth [1]. An example of this grid pattern is shown at the bottom of the figure. 2. These images are used to train an autoencoder using photometric loss and skip losses. 3. All the encoded training images and the encoder are transferred onto the UAV. 4. The live image captured by the UAV is passed through the trained encoder. 5. The encoded live image is compared with a subset of the encoded reference images using an inner-product kernel outputting a weight for each reference image. 6. These weights are used to compute the localization and covariance.

The largest challenge with this idea is finding a way to accurately and robustly localize real live images captured from a UAV with the artificial images rendered from GE. Since the satellite images used for the reconstruction were captured years ago, there are differences with the live images in terms of lighting, small object movement (e.g. vehicles, trailers), large structural changes (e.g. building additions/demolitions), and unusual object reconstruction, particularly for non-rectangular based objects like trees. This makes it difficult for feature-based methods to obtain accurate and robust results in many cases.

Patel et al. [8] used multiple GE images rendered around the desired flight path and used mutual information (MI)

The authors are affiliated with the University of Toronto Institute for Aerospace Studies (UTIAS): mollie.bianchi@robotics.utias.utoronto.ca, tim.barfoot@utoronto.ca

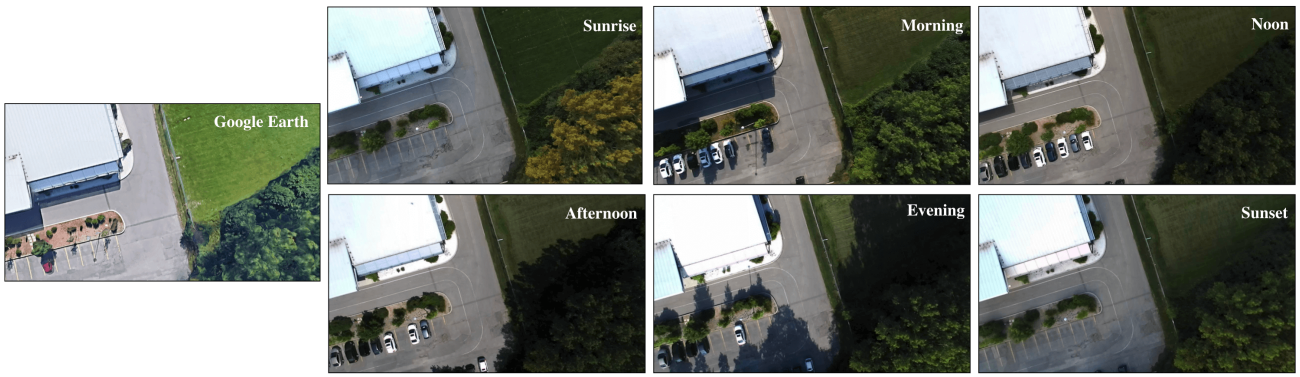


Fig. 2: An image from each of the six lighting conditions in the dataset and a corresponding image rendered from GE [1] is shown. The shadows present in the GE images most closely resemble those present in the morning lighting condition. The shadows in the afternoon and evening datasets appear on the opposite side of objects as compared to GE.

to search for the best alignment with the live image. This approach was computationally expensive and would require storing thousands of full-sized images on board the UAV. It is not capable of running in real time.

This work introduces a new method to use prerendered satellite images that is fast and storage efficient. As in [8], images are rendered around the desired flight path in GE. An autoencoder is trained on these path-specific images to compress them to a much smaller vector representation. The same autoencoder is used to compress the live images as well. The compressed live image vector is compared to all nearby compressed GE image vectors through an inner-product kernel. This results in weights associated with each of the corresponding GE image poses. From these weights, a localization for the longitude, latitude, and heading with accompanying covariance is computed. This method has been demonstrated on a real UAV dataset of images along a 1.1km path at six different times of day covering several lighting conditions. In comparison with [8], we are able to achieve the same accuracy performance on image registration and run in 1% of the computation time.

The rest of this paper is organized as follows. Section II reviews the related work from the literature. Section III discusses our methodology. Sections IV and V provide our experimental results on a real UAV dataset. Section VI wraps up with our conclusions and suggestions for future work.

II. Related Work

A. Aerial Visual Localization

Early works in visual aerial localization looked at using edge detection [9] or a combination of classical techniques with learned semantic segmentation [10]. Both these approaches perform better at high-altitude flights where more structure is present in the images and suffer in areas comprising mainly grass and trees. There have been more recent feature-based methods that match street view images to images from a ground robot [11] and a UAV [12], [13]. Place recognition is performed using a visual bag-of-words technique and then followed by image registration using Scale Invariant Feature Transform (SIFT) [14] keypoints. However, feature matching has been shown to contain significant numbers of outliers due to large changes in appearance and viewpoint.

The current best method for localization using satellite images is the MI based approach presented in [8]. This was largely inspired by [15], [16] in which MI had been used to localize monocular camera images within a textured 3D model of the environment. Adapting this idea to a UAV, Patel et al. [8] were able to achieve less than 3m and 3° Root Mean Square Error (RMSE) on low-altitude flights at six different times of day. In their work, images were rendered from GE beforehand every 3m along the path and around the path at intervals of 6m. The Normalized Information Distance (NID), which is a MI based metric, was computed between the live image and all images within 4m of the prior pose estimate (e.g., from filtering) to select the best-matching image. The alignment between this geo-referenced image and the live image was then computed by a series of coarse and refined optimizations of the warping parameters. Each step of the optimization required numerically computing the Jacobian of the NID with respect to the warping parameters. This process was quite slow making this method incapable of running in real time. As well, the images were stored in their full 560×315 resolution resulting in large storage costs for longer paths. We build on the idea of prerendering images around a desired path in GE, but improve upon [8] by eliminating the costly optimization step, improving runtime, and decreasing storage requirements.

B. Autoencoders

One of the core limitations with [8] is that the images are large, each is 176,400 pixels making the MI computation slower. A common learning-based method for compressing images is autoencoders [17]. One neural network acts as the encoder, compressing images down to some low-dimensional bottleneck. A second network upsamples this bottleneck vector back to an image with the same size as the original image. Minimizing some loss function between the original image and the recreated image, the network can learn to retain the key features in the bottleneck. There has been lots of work in this area, including new loss functions [18], [19],

adversarial autoencoders [20], and combining autoencoders with neural autoregressive models [21]. In this work, we use an autoencoder architecture based on [18] to compress our images.

C. Kernels

Kernels, such as the inner-product or exponential kernel, are often used for matching patches between images, such as in [22], [23]. They provide a measure of the similarity between the two patches, but they are not commonly used for comparisons of whole images due to the high number of pixels involved. We use kernels on the autoencoded representations of whole images. Since these autoencoded images are small enough to quickly compute a kernel between them, it eliminates the need for extracting and matching patches from an image.

D. Learned Pose Estimation

Using learned methods to directly compute the poses of objects in images, or the relative pose change between two images has been the focus of many works [24]–[26]. However, these approaches are limited by the available training data. Real data is expensive to collect and label, and synthetic data does not typically generalize directly to the real world. Alternatively, Sundermeyer et al. [27] use a similar method to what is proposed in this work to perform 6D Object Detection. Instead of explicitly learning from 3D pose annotations during training, they implicitly learn representations from rendered 3D model views. Using a denoising autoencoder, they generate a codebook containing the encoded representations of tens of thousands of rendered images of the desired object at uniformly distributed poses. The same autoencoder is used to encode a live image and a cosine similarity metric is used to match the live image with the closest poses from the codebook.

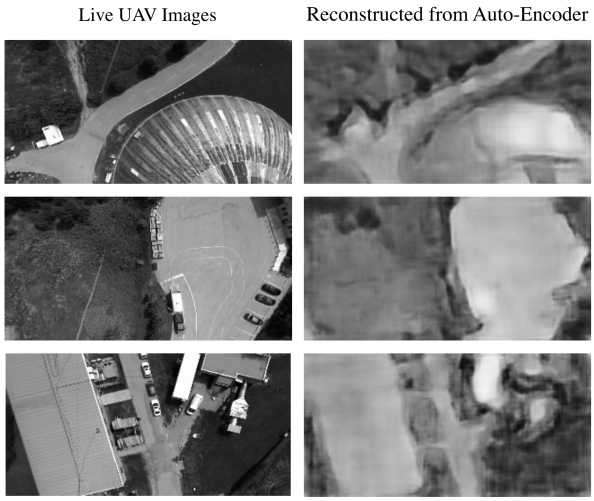


Fig. 3: On the left are real images collected by the UAV along the path. On the right is the corresponding image after passing through the autoencoder and decoder which were only trained on GE images.

III. Methodology

The proposed approach can be divided into several steps as depicted in Figure 1. Offline, images are rendered from GE around the desired flight path. Then an autoencoder is trained for this specific path using these images. These images are encoded and saved after passing through the trained network. While the offline processing is significant, it only needs to be completed once per path and would eliminate the need for manual mapping flights before each autonomous flight as in [5]. In the online component of the pipeline, weights for a subset of these autoencoded reference images are computed using an inner-product kernel computation with an autoencoded live image. The localization and corresponding covariance are then computed from these weights. Finally, outlier rejection is performed based on the covariance estimate from the previous step.

A. Pre-Flight Image Collection

Using a desired path, images are rendered from GE [1] at the intended orientation every 0.5m along the path. Additional images are rendered at 0.5m lateral offsets out to 5m to either side of the path. This requires 42 images for each meter of the path. This coverage could be modified based on expected performance of the UAV. For example, if you are expecting the UAV to operate in windy conditions more images could be rendered further from the path. Regardless, this leads to a high number of images for non-trivial path lengths. Storing and comparing these images in full size would be infeasible. The next step and key aspect of this method is to use an autoencoder to compress the images to a low-dimensional representation while maintaining the key features of each image such that comparing the compressed images using a kernel yields sensible results.

B. Autoencoder

With a focus on place-specific excellence, the autoencoder is trained solely using the precollected images from GE for that desired path. A new autoencoder would need to be trained for each path. The autoencoder architecture is based on [18] as implemented in [28]. The input is a 320×160 greyscale GE image. The encoder is composed of six layers. Each of the first five layers perform a 2D convolution with a stride of two followed by a batch normalization layer. The number of channels double as indicated in Figure 4. Finally, a linear layer maps the output of the final convolution layer to the bottleneck vector. Different sizes were experimented with for the dimension of the bottleneck. A bottleneck of dimension 1000 was selected as it was the smallest size that could still achieve the desired accuracy.

The decoder behaves opposite the encoder. A linear layer first maps the bottleneck variable to 1024 channels. This is then passed to the first of five layers, each of which performs upsampling by a factor of two, followed by convolution with a stride of three, and batch normalization. The number of channels is halved in each layer until an output greyscale image with the same dimension as the input image is

generated. To obtain the compressed image vector, the output after only the encoder part of the network is used.

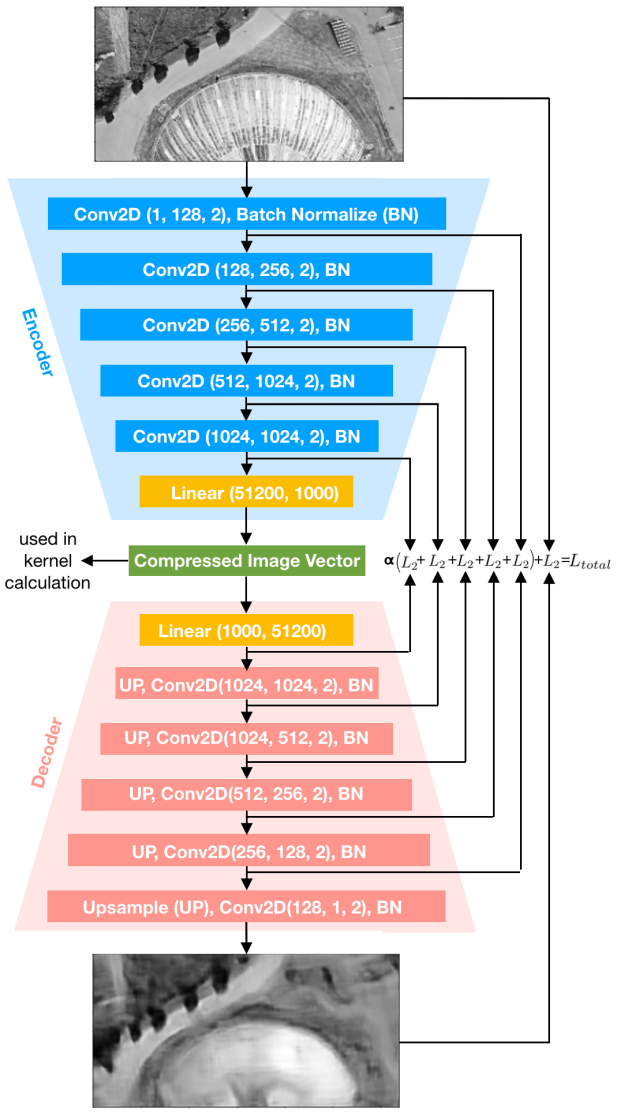


Fig. 4: The autoencoder architecture is based on [18]. It uses traditional photometric loss between the input image and the reconstructed image as well as skip losses between the corresponding layers in the encoder and decoder.

The loss function used to train the network is a combination of photometric loss between the input and output images, i.e., L = $(I_{input} - I_{output})^2$, and L2 loss between the outputs of corresponding layers referred to as skip losses. These additional skip losses are weighted with a value of 0.01 and encourage the decoder to learn the reverse behaviour of the encoder and was found to improve performance on the real image validation sets. The network was trained for 20 epochs with a learning rate of 1e-4. For the path used in the experiments, the network was trained with approximately 48,000 images. On a Nvidia DGX Station using a single Tesla V100 GPU training took around 20 hours to complete.

It is important to stress that the network is trained only on images from GE, but it is still able to generalize to real-world images with different lighting conditions. Thus it is able to be trained before having to actually fly the path. Figure 3 shows some examples of real images reconstructed after passing through the autoencoder. While the reconstructions are not as sharp as the reconstruction of the training data as seen in Figure 4, the main structures in each image are preserved. For our application of comparing the encoded live image to encoded reference images, the network achieves sufficient generalization performance. It may be interesting for future work to look into using data augmentation during training to further improve generalization performance similar to [27].

C. Localizing Using Kernels

To minimize time spent loading the autoencoded GE reference images, all image vectors are stacked and loaded into a $1000 \times N$ dimensional matrix denoted $\mathbf{Y}_{\text{ge}}$, which is loaded onto the GPU. Then, based on a prediction of the current live image pose, $\mathbf{Y}_{\text{ge}}$ is indexed to include only the reference images that are 4m ahead and behind along the path, which is the same search area used in [8]. The live image is passed through the trained autoencoder and the resulting compressed 1000×1 vector is denoted as $\mathbf{y}$. The weights, $\mathbf{w}$, are computed for the subset of autoencoded GE reference images using a basic inner-product kernel:

$$\mathbf{w} = \mathbf{Y}_{\text{ge}}^T \mathbf{y}. \tag{1}$$

$\mathbf{w}$ contains a similarity measurement between the live image and each of the reference images. Since there are 336 images being used for comparison, many of these images have low, but non-zero, weights. These weights tend to pull the mean value towards the centre of the area covered by the reference images. To prevent this and get a result closer to the images with the highest weights, a new set of weights, $\mathbf{w}_{\text{th}}$, is created by setting all values of the weights less than one standard deviation of the max weight to zero. The thresholded weights are then normalized:

$$\bar{\mathbf{w}}_{\text{th}} = \frac{\mathbf{w}_{\text{th}}}{\sum_i w_{\text{th},i}}. \tag{2}$$

The longitude and latitude coordinates of each reference image are stacked in a $2 \times N$ matrix $\mathbf{X}_{\text{ge}}$. The thresholded weights are used to compute the localization for the longitude and latitude according to:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = \mathbf{X}_{\text{ge}} \bar{\mathbf{w}}_{\text{th}}. \tag{3}$$

The covariance is computed using the original weight values:

$$\mathbf{P} = \sum_i w_i (\mathbf{x}_{\text{ge},i} - \hat{\mathbf{x}})(\mathbf{x}_{\text{ge},i} - \hat{\mathbf{x}})^T. \tag{4}$$

Some examples showing the localization, covariance, and weights generated for each nearby GE reference image are shown in Figure 8.

Instead of rendering reference images at multiple headings and including them in the previous computation, the heading computation is performed after the above step. The reference image with the largest weight is selected for comparison,

$\mathbf{y}^{\star}_{\text{ge}}$. The uncompressed live image is then rotated in 1° increments between -5°and 5°. All these rotated images are autoencoded and stacked into an 1000×11 matrix, $\mathbf{Y}_{\theta}$. The kernel computation from (1) is repeated:

$$\mathbf{w}_{\theta} = \mathbf{Y}_{\theta}^{T} \mathbf{y}^{\star}_{\text{ge}}. \tag{5}$$

These weights are normalized, $\bar{\mathbf{w}}_{\theta}$, and then used to compute a heading measurement following the same procedure as in (3) using a stacked vector of the rotation values, $\mathbf{x}_{\theta}$:

$$\hat{\theta} = \mathbf{x}_{\theta}^{T} \bar{\mathbf{w}}_{\theta}. \tag{6}$$

This mean heading value, $\hat{\theta}$, by which the live image has been rotated is then subtracted from the heading of the selected reference image, β, to get a global heading, $\beta - \hat{\theta}$. While not included here, a similar process could be used to get an estimate for altitude without having to render additional reference images. The full localization is then:

$$\hat{\mathbf{p}} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \beta - \hat{\theta} \end{bmatrix} \tag{7}$$

One of our earlier approaches for localization was to use the position of the image with the highest similarity measurement. This yielded fairly similar results to the weighted average approach except that it was limited by the grid spacing of the reference images and more susceptible to outliers.

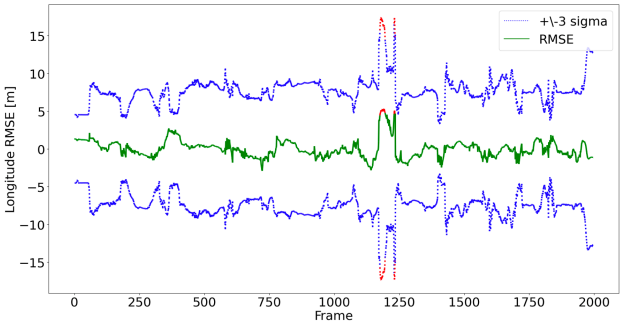


Fig. 5: Plots of the RMSE for the longitude coordinates of the sunrise test are shown here with accompanying 3σ uncertainty envelope. The registrations that were rejected due to either σ_{long} or $\sigma_{\text{lat}} > 5$ are shown in red.

D. Outlier Rejection

Since we compute a covariance with our localization based on the weights, we can also use this to reject outliers. When the weights have a single narrow peak away from the edges of the area covered by the reference images, the σ^2_{long} and σ^2_{lat} values are small. When the weights are more spread out with a less-well-defined peak, when there are multiple peaks, or when the peak occurs very close to the edge of the reference area, this results in larger values for σ^2_{long} and σ^2_{lat}. We reject localizations that have either σ_{long} or σ_{lat} greater than 5. Figure 5 plots the RMSE for the longitude coordinates with accompanying covariance. Rejected registrations are indicated in red.

IV. EXPERIMENTAL SETUP

A. Image Registration on UTIAS Dataset

Image registration to obtain the longitude, latitude, and heading was performed on the same dataset as in [8]. We do not focus on estimating the roll, pitch, or altitude of the vehicle as those can be measured by complementary sensors to vision. The data was collected at UTIAS using a DJI Matrice 600 Pro multirotor UAV with a 3-axis DJI Ronin-MX gimbal (see Figure 6). A StereoLabs ZED camera provides stereo images at 10FPS. The RTK-GPS system and IMU provide the vehicle pose for ground truth.

This dataset consists of six traversals of a 1132m path over built-up areas with roads and buildings as well as large areas of grass and trees. Each run captures the distinctive lighting condition at different times of day: sunrise, morning, noon, afternoon, evening, and sunset. These lighting conditions are shown in Figure 2. The UAV flies at an altitude of 40m with a constant heading and the camera pointed in the nadir direction.

There is an unknown offset between the RTK-GPS frame and the GE frame. So 10% of the successful image registrations are used to align the frames. These registrations are then omitted from all error calculations.

V. RESULTS

Most vision-based localization methods rely on features. Previously, Patel et al. [8] evaluated the ability of SURF features to match between GE rendered images and live images on the same path used here under various lighting conditions using the VT&R framework in [5]. Using the GE images for the teach pass and the live images for the repeat, features were only capable of producing less than 7% successes per repeat if a successful registration is defined as having greater than 30 Maximum Likelihood Estimation Sample Consensus (MLESAC) inliers. We are not currently aware of any works localizing GE images to real UAV images at a similar altitude and orientation as our flight path other than [8].

We are able to achieve comparable results with the MI-based approach from [8] on the same dataset in 1% of the computation time. In Table II, we present our RMSE for the

Fig. 6: The dataset was collected by a 3-axis gimballed stereo camera on a multirotor UAV.

TABLE I: Comparison of Errors for Successful Registrations

Lighting Condition	Registration Success [%]			Successful Registrations RMSE								
				longitude [m]			latitude [m]			heading [degree]		
	Ours A	Ours B	MI [8]	Ours A	Ours B	MI [8]	Ours A	Ours B	MI [8]	Ours A	Ours B	MI [8]
Sunrise	98.8	**99.9**	94.7	**1.05**	1.17	1.10	0.97	0.97	**0.71**	**0.35**	**0.35**	2.28
Morning	**100**	**100**	95.1	**0.90**	**0.90**	1.02	0.95	0.95	**0.58**	**0.31**	**0.31**	2.57
Noon	**100**	**100**	97.8	1.04	1.04	**0.78**	0.87	0.87	**0.61**	**0.36**	**0.36**	1.82
Afternoon	**98.0**	**98.6**	96.0	**1.64**	1.67	1.69	**0.88**	**0.87**	0.92	**0.34**	**0.34**	1.17
Evening	90.9	**96.0**	81.3	**2.16**	2.17	3.03	1.18	**1.14**	1.32	**0.42**	**0.41**	2.49
Sunset	97.4	**98.7**	87.5	**1.37**	1.48	1.95	0.96	**0.94**	1.12	**0.36**	**0.36**	2.64

longitude, latitude, and heading for all registrations on each of the six runs. We achieve lower errors as compared to the results presented in [8]. We use the same search area as in [8] to select our subset of reference images, which correspond to a maximum RMSE of 6.4m.

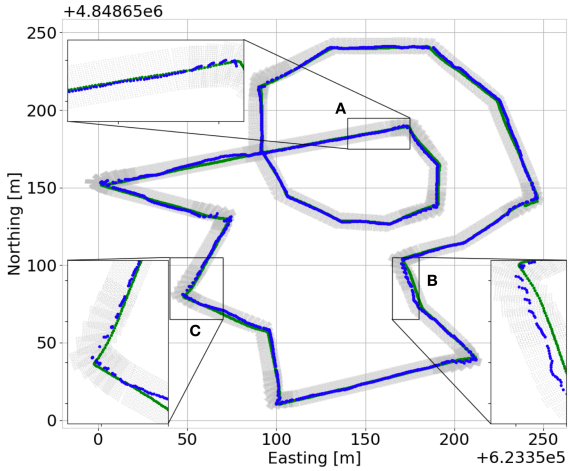
In Table I, the error results from only successful registrations are compared against the errors from successful registrations in [8]. For "Ours A", we use the outlier rejection scheme described in Section III-D to reject registrations with a high covariance estimate. For "Ours B", we use the outlier rejection from [8] along with our localization method. In [8], registrations are deemed unsuccessful if the localization is too far from the previous estimate. There is minimal difference in the performance between "Ours A" and "Ours B". The benefit of our outlier rejection scheme is that it is based purely on the covariance of the localization result and does not require a prior estimate.

In comparison with [8], for all but the noon lighting condition we achieve lower RMSE error on the longitude coordinate. For the latitude coordinate, we have lower RMSE error for three of the runs and for the other three runs we are an average of 0.3m higher. Our success rate of registrations is higher for all the lighting conditions. Particularly in the evening and sunset runs, we see an increase of ~10% in the success rate and a decrease in RMSE.

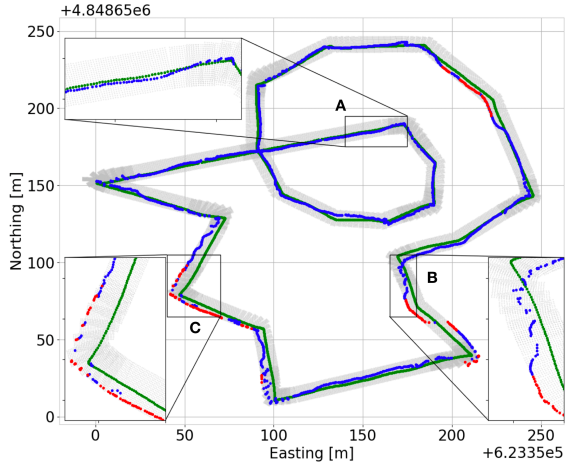
The most significant advantage of our method over [8] is the substantial reduction in runtime. Both methods were run on a Lenovo P52 laptop with an Intel i7 8th generation core, a Nvidia Quadro P2000 GPU, and 32 GB of RAM. Most of the MI registrations took between 5 to 35 seconds per frame, whereas our approach only took between 0.09 and 0.15 seconds. We were able to eliminate the costly optimization component from [8], which requires warping the image and recomputing the MI up to 150 times per registration. Instead, by rendering more reference images at a finer grid spacing and using the mean of the kernel weights to interpolate between them, we were able to achieve similar results at greatly reduced computation time.

TABLE II: Comparison of Errors for All Registrations

Lighting Condition	All Registrations RMSE					
	longitude [m]		latitude [m]		yaw [degree]	
	Ours	MI [8]	Ours	MI [8]	Ours	MI [8]
Sunrise	**1.18**	1.87	**0.98**	1.47	**0.35**	2.80
Morning	**0.90**	2.24	**0.95**	1.39	**0.31**	2.97
Noon	**1.04**	1.26	**0.87**	1.02	**0.36**	2.70
Afternoon	**1.84**	2.14	**0.90**	1.57	**0.35**	2.63
Evening	**2.53**	4.09	**1.19**	3.63	**0.42**	5.25
Sunset	**1.64**	3.03	**0.97**	1.95	**0.37**	3.06



(a) Morning



(b) Evening

Fig. 7: Registration results showing our best (morning) and worst (evening) localization results. Grey dots indicate the reference image positions. Green dots indicate the ground truth live image positions. Blue dots indicate accepted localizations and red dots indicate rejected localizations. Shadows on the opposite sides of objects as compared to the GE reference images cause higher errors and more rejected registrations in the evening run.

Both approaches are still limited by the storage available on the UAV. By autoencoding the reference images, we only need 1000 numbers to represent each image. Recording these numbers as half precision floats only requires 4.2 kB per image. In [8], the reference images are stored as 560×315

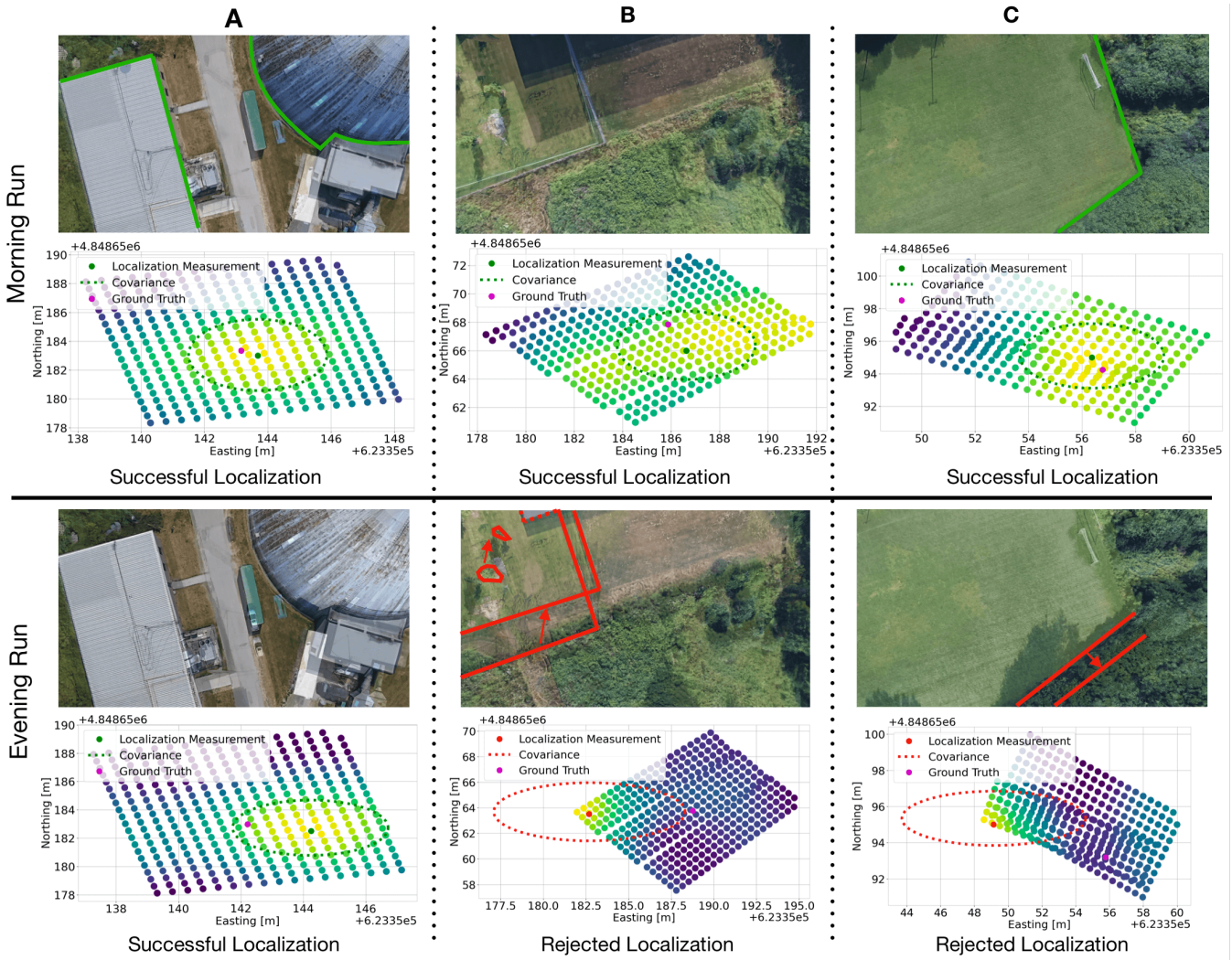


Fig. 8: An example frame from each of the three areas indicated in Figure 7 for the morning lighting condition is shown at the top of this figure and for the evening lighting condition on the bottom. For each lighting condition, the top row shows the overlay between the live image and the GE reference image closest to the localization. The heat maps plotted in the bottom row show the value of the weights for each of the nearby GE reference images with yellow indicating a higher weight. The resulting localization and covariance is shown in green for successful registrations and in red for rejected registrations. The ground truth is shown in magenta.

4-bit greyscale images requiring approximately 11 kB of storage which is almost three times as large. As a result of this reduction, we are able to render more images per meter of the path while still having a lower per meter storage cost, 0.241 Mb compared to 0.722 Mb. Encoding the images also makes the base comparison computation faster. An inner-product computation between two 1000 dim vectors takes on average 0.26ms, whereas a MI computation between two 176,400 pixel images takes on average 109ms. The disadvantage of our approach is that in addition to the autoencoded reference images, we must also store the weights for the trained neural networks in order to encode the live image on board. However, this is a fixed cost that does not increase with the length of the path. So for the 1.1km path in the dataset used, our total storage is still less than what is used in [8]. Computation and storage comparisons are summarized in Table III.

We plot our registrations on the run with our best localization results, morning, in Figure 7a and on the run with the worst localization results, evening, in Figure 7b. The ground truth from the RTK is shown by the green dots. Successful registrations are indicated by blue dots and rejected registrations are indicated by red dots. The grey dots show the positions of the GE reference images. The localizations on the morning lighting condition likely perform the best because the shadow conditions match those on the GE images. In the evening dataset, the shadows are on the opposite sides of the objects as compared to the GE images. Figure 2 shows an example image from each of the six lighting conditions for comparison.

We show an example from three different areas along the path for the morning and evening lighting conditions in Figure 8. The top row for each lighting condition shows an overlay of the live image and the GE reference image

TABLE III: Runtime and Storage Requirements Comparison

Comparison Method	Kernel (Ours)	MI [8]
Average Runtime for 1.1 km Path	**221 s**	18422 s
Average Time per Frame	**0.11 s**	9.23 s
Average Time per Comparison Computation	**0.26 ms**	109 ms
Storage Cost per Image	**4.2 kB**	11 kB
Storage Cost per m of Path	**0.241 Mb**	0.722 Mb
Fixed Storage Cost	158 Mb	**0 Mb**
Total Cost for 1.1km Path	**423 Mb**	794 Mb

closest to the localization. The bottom row shows a heat map of the value of the weights of the nearby GE reference images computed from an inner-product with the live image. The resulting localization from the mean of the thresholded weights and the covariance estimate are plotted as well. For successful registrations, the covariance envelope is smaller. For the rejected localizations (i.e., evening B and C), shadows cause the highest weights to occur on misaligned images at the very edge of the reference images. In these cases, the covariance that results from the weights is larger than in the successful registrations, very elongated, and does not pass our threshold for outlier rejection.

VI. CONCLUSIONS AND FUTURE WORK

We presented a method for localizing live images captured from a UAV under six different lighting conditions to pre-rendered images from GE. Compared to the best existing method, we are able to achieve a similar level of accuracy at 1% of the computation time. Our method also has a lower storage requirement per length of path making it an ideal candidate for running on board the UAV in future work. All preprocessing can be completed offline and grants the UAV the ability to traverse new areas without having to manually fly and map them first. Since our method is able to match images across large periods of time (the GE images are at least two years older), it could also be used for repeated traversals of the same path over large periods of time. Future work aims at integrating our method into a filtering pipeline such that it can be used in the loop on board the UAV.

ACKNOWLEDGMENT

This work was funded by NSERC Canada Graduate Scholarship-Master's, Defence Research and Development Canada, Drone Delivery Canada, the Centre for Aerial Robotics Research and Education, University of Toronto, and the Vector Scholarship in Artificial Intelligence.

REFERENCES

[1] Google Earth. Image Credits: Google, Landsat/Copernicus. Location: UTIAS, North York, Ontario, Canada. Accessed: Aug 24, 2020.

[2] M. Blöesch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based mav navigation in unknown and unstructured environments," in *Proc. Int. Conf. Robot. Automat.*, 2010, p. 21–28.

[3] S. Shen, N. Michael, and V. Kumar, "Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft mavs," in *Proc. Int. Conf. Robot. Automat.*, 2015, pp. 5303–5310.

[4] S. Weiss *et al*., "Monocular vision for long-term micro aerial vehicle state estimation: A compendium," *J. Field Robot.*, vol. 30, no. 5, pp. 803–891, 2013.

[5] M. Warren, M. Greeff, B. Patel, J. Collier, A. P. Schoellig, and T. D. Barfoot, "There's no place like home: Visual teach and repeat for emergency return of multirotor uavs during gps failure," *IEEE Robot. Automat. Lett.*, vol. 4, no. 1, pp. 161–168, 2019.

[6] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long range rover autonomy," *J. Field Robot.*, vol. 27, no. 5, pp. 534–560, 2010.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[8] B. Patel, T. D. Barfoot, and A. P. Schoellig, "Visual localization with google earth images for robust global pose estimation of uavs," in *Proc. Intl. Conf. Robot. Automat.*, 2020.

[9] G. Conte and P. Doherty, "An integrated uav navigation system based on aerial image matching," in *Proc. IEEE Aerosp. Conf.*, 2008.

[10] A. Nassar, K. Amer, R. ElHakim, and E. M., "A deep cnn-based framework for enhanced aerial imagery registration with applications to uav geolocalization," in *Proc. IEEE. Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018.

[11] P. Agarwal and L. Spinello, "Metric localization using google street view," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 3111–3118.

[12] A. L. Majdik, Y. Albers-Schoenberg, and D. Scaramuzza, "Mav urban localization from google street view data," in *Proc. Int. Conf. Intell. Robots Syst.*, 2013, pp. 3979–3986.

[13] A. L. Majdik, D. Verda, Y. Albers-Schoenberg, and D. Scaramuzza, "Air-ground matching: Appearance-based gps-denied urban localization of micro aerial vehicles," *J. Field Robot.*, 2015.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[15] G. Pascoe, W. Maddern, and P. Newman, "Robust direct visual localisation using normalised information distance," *Proc. British Mach. Vis. Conf.*, pp. 70.1–70.13, 2015.

[16] G. Pascoe, W. Maddern, A. D. Stewart, and P. Newman, "Farlap: Fast robust localisation using appearance priors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 6366–6373.

[17] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. Int. Conf. Learn. Representations)*, 2014.

[18] X. Hou, L. Shen, K. Sun, and G. Qiu, "Deep feature consistent variational autoencoder," in *Proc. Appl. Comput. Vis. IEEE Winter Conf.*, 2017, pp. 1133–1141.

[19] K. Ridgeway, J. Snell, B. Roads, R. Zemel, and M. Mozer, "Learning to generate images with perceptual similarity metrics," in *Proc. Int. Conf. Image Process.*, 2015.

[20] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow, "Adversarial autoencoders," *Proc. Int. Conf. Learn. Representations*, 2015.

[21] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel, "Variational lossy autoencoder," *Proc. Int. Conf. Comput. Vis.*, 2016.

[22] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2005, pp. 1458–1465.

[23] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schomid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *Int. J. Comput. Vis.*, vol. 73, no. 2, pp. 213–238, 2007.

[24] I. Melekhov, J. Kannala, and E. Rahtu, "Relative camera pose estimation using convolutional neural networks," in *Proc. Int. Conf. Adv. Concepts Intell. Vis. Syst.*, 2017, pp. 675–687.

[25] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proc. IEEE Intl. Conf. Comput. Vis.*, 2015, pp. 2938–2946.

[26] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 292–301.

[27] M. Sundermeyer, Z. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientationlearning for 6d object detection from rgb images," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 699–715.

[28] H. Bai, "Variational Autoencoder for face image generation in PyTorch", Github Repository. [Online]. Available: https://github.com/bhpfelix/Variational-Autoencoder-PyTorch. Accessed: Jul. 10, 2020.