AtLoc4TOPO: Adapting Attention Guided Camera Localization for the Geodetic Engineering Laboratory (TOPO) at EPFL

Gerber Frédéric, El Alaoui Réda and Reding Simon
Project 2, Machine Learning (CS-433), EPFL

***Abstract*—This project tackles the pose estimation problem by bypassing intermediate steps. Precisely, instead of generating absolute coordinates from an image, and then using these to extract a pose, we adapt the AtLoc paper in order to get the pose directly. Additionally, we provide intermediate saliency maps, which show the most robust points of an image.**

I. Introduction

With the wide spread of electronic devices these days, it has become increasingly important for users to locate themselves in time and space using the Global Positioning System (GPS). Indeed, additionally to being the most important clock source, GPS provides information about the position of a receiver using satellites that orbit around the Earth. Practically, this simple mechanism has many use cases, among which self-driving cars or flying drones.

However, not only is GPS vulnerable to attacks, but in remote locations like mountains, a signal may not even be available. This leads to the idea of using a different way of acquiring precise localization information. For instance, in the drone case we will focus on, two elements are necessary for good orientation: (1) the *position* in three WGS84 coordinates, and (2) the *orientation* in degrees. When bundled together, these two pieces of information are usually called a *pose*.

Compared to other, more specialized sensors, cameras are low-cost and highly available nowadays and usually come as built-in components of drones, it seems practical to use their images as inputs to learn poses with a machine learning model. On the high level, we want to achieve the following approximation:

$$f(\text{image}) \;=\; \text{pose}\;\big[\equiv(\text{position}, \text{orientation})\big].$$

Researchers have already been working on camera localization in aerial contexts, but it is largely impractical to rely on previous photographs in order to train a predictor for poses in unknown locations. As a result, since the community typically trains and tests their models on real datasets, the next step consists in training on *synthetic datasets* and later evaluating them on real images. For instance, Yan [1] from the TOPO laboratory at EPFL achieves this in two steps: (1) create a three-dimensional absolute coordinate from a two-dimensional image, and (2) use these intermediate coordinates to output an estimate of the pose. Overall, the final accuracy is very good, since poses are at most 10 meters away from the ground truth.

Another approach is using data to directly estimate position and orientation. To make this approach more robust, attention guided techniques [2] are used, that is, models which focus on features that are robust across many inputs. Intuitively, the network learns what is most important in the image. This paper, called AtLoc, uses *saliency maps* to visualize the robust features of the input image.

Subsequently, this report will (1) describe the task we have performed in the scope of this machine learning project, (2) detail the way we adapted and ran the code, and (3) discuss the evaluation results of our method.

II. Description of the Task

As mentioned in the Introduction, pose estimation is ideally trained on synthetic images, while the testing and actual predictions happen on real images. Since these types of data items don't look the same, subject to weather or lighting conditions, it is interesting to benefit from the attention guided approach developed in AtLoc [2] to focus only on the lines that matter most in an image. Therefrom, this section will go through a more precise description of the task we are addressing in our work.

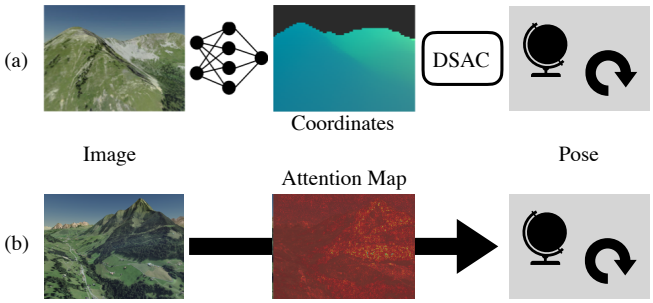

Figure 1. **Previous work (a) *vs* AtLoc (b)**: In (a), the coordinates are generated as an intermediate step, while (b) directly estimates the pose (position, orientation) with internally a visual encoder, an attention module and a pose regressor; see [2] for details.

A. OneShot Camera Pose Estimation

More precisely, Figure 1 shows a comparison between prior work and our adaptation of AtLoc for pose estimation with the TOPO datasets. In fact, Yan regresses in his work

[1] a set of absolute coordinates from a $480 \times 640 \times 3$ RGB image with a deep neural network. After that, he applies a custom RANSAC solver called DSAC [3] to get the final pose, *i.e.* a position and an orientation.

B. Attention Guided Camera Localization

On the other hand, AtLoc internally creates attention maps that can be visualized using *saliency maps*, whereby the most important features are highlighted. This construction is only an intermediate step though, so pose regression happens directly with an image once the model has been trained[1]. Consequently, geometrically robust features are more likely to need "attention", which is the reason why the TOPO lab is interested in this work. As a result, our work consists in the following steps: (1) run the AtLoc code as provided in its official repository, (2) adapt its dataloader and run scripts to plug some custom TOPO datasets into it, and (3) interpret/optimize the results. Finally, the ultimate goal is to find out whether AtLoc is any good in pose estimation.

III. Running the Code with a New Dataset

The AtLoc Paper shows its performance on indoor and outdoor datasets with pictures and cameras attached to a car. In those images, stable features include edges of house walls, pieces of furniture and the contour of the horizon. Moreover, unstable features which AtLoc successfully avoids in these scenes are cars, pedestrians and texture-less surfaces in general. Since our use case focuses on drones, we will have to test whether AtLoc successfully identifies other stable features in more rural environment and from different perspectives in new datasets provided by the TOPO lab at EPFL.

A. The Datasets

We have customized dataloaders for three datasets:

- Comballaz Synthetic Archive;
- Comballaz (Air 2);
- EPFL (Phantom).

The first dataset is, as the name indicates, a large dataset of synthetically generated images, while the other two contain real images recorded on drones, combined with matching synthetic images at the exact same poses, as compared on Figures 2 and 3. Notable here is that different drones capture scenes differently: while the DJI Phantom has its camera recording close to vertically, recording features from above (top down), the Air 2 records images horizontally, capturing ridgelines and other stable features on the horizon, but in some images also a large portion of sky. Such empty spaces contain no features at all in the synthetic case, but in the real data they may contain a large amount of unstable features, like clouds, planes or the sun...

[1]The authors of the AtLoc paper have tested their method on the 7Scenes and RobotCar datasets.

Figure 2. **Seasons change features even in seemingly optimal weather conditions**: In addition to color changes in the whole image, there are several trees in the center of the image where the canopy is missing due to the lost leaves, revealing the texture of the ground behind it. The fog in the valley as well as the angle of the sun block additional texture in the distance: only the silhouette of the main ridgelines are visible. Observe also the larger shadows cast which are caused by the time of day but also amplified due to the season.

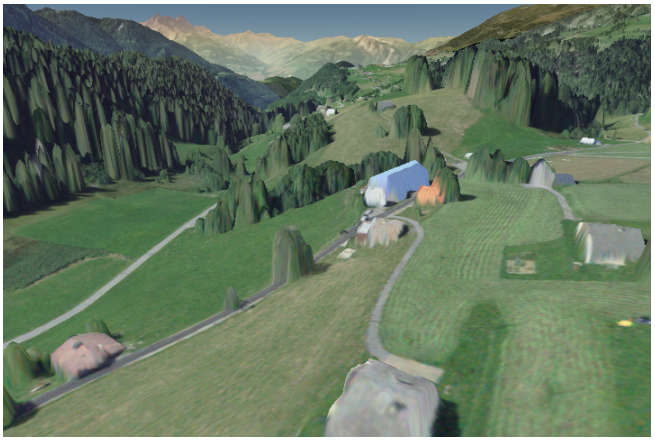

Figure 3. **Synthetic counterpart to the real image**: We can see that evergreen and leaved trees are indistinguishable. In addition, some solid, stable features such as power lines are missing. Additionally, the better lighting pronounces the texture along the forest edge and the texture of the alps in the distance.

Since drones move in the 3D space, we further modify the poses to better fit our use case. In particular, instead of $\text{pose}_{\text{AtLoc}} = (\text{lat}, \text{long}, a, b_i, c_j, d_k)$ with latitude/longitude WGS84 coordinates and quaternions, we use $\text{pose}_{\text{TOPO}} = (x, y, z, \text{pitch}, \text{yaw}, \text{pitch}, \text{roll})$ where x, y, z are ECEF coordinates derived from the WGS84 and height information provided in the datasets. This allows us to account also the error in height. We compute the distance d between the target t and predicted p as:

$$d = \sqrt{(x_t - x_p)^2 + (y_t - y_p)^2 + (z_t - z_p)^2}$$

using the 3D Pythagorean theorem.

B. In Search of Stable Features

Keeping the applications of drones in mind, we have to choose stable features over seasons and time of day that the mechanism can 'pay attention' to:

- **Ridgelines**: These are perhaps the most stable features, especially when considering distinct features of the Alps of our datasets. However, they may not always be available due to cloud cover or, in the case of the Phantom dataset, completely absent due to the camera angle. It should be noted that this feature will not be extensible to other regions: in many places, mountains are too far away to be visible in plain sights, or the ridgelines not distinct enough[2].
- **Buildings**: They are relatively stable, but may not always be available or distinct enough (rooftops often have similar shape and texture).
- **Forests**: Those are stable in location but change their form during the season. It is important here that leaf trees can change the landscape significantly throughout the seasons: the missing canopy can reveal additional features such as roads and streams that would not be visible in the summer.
- **Roads**: While in extreme conditions (during or immediately after heavy snowfall) or particular circumstances (when they are closed thus unrecognizable from the air), they tend to be a stable feature with a distinct shape. But more importantly, their texture won't change during the day as they are flat on the ground: they will be less affected by shadows.

C. Running Code on the IZAR Cluster

Now that we are familiar with the datasets, let's dive into the actual code. Fortunately, the repository provided by the AtLoc authors[3] has clear instructions regarding Python dependencies on `numpy`, `torch` and other specific modules. However, it relies on `Conda`, whereas the IZAR cluster[4] (where the TOPO datasets are stored) only supports `pip`. Therefore, we created a custom virtual environment called `atloc` containing the necessary modules.

Additionally, we wrote several batch scripts, which both unzip the datasets to temporary directories and run either training or testing on a GPU. All in all, the process of writing and debugging those scripts probably took much longer than the ten hours that were actually needed for an their execution, but it was very instructive nonetheless.

Regarding the code, each dataset[5] led to three scripts: if we take for instance Comballaz Air, we have (1) `prepare_air`, that copies the data from the VNAV archive into our repository file system; (2) `train_air`, whose purpose lies in offloading the lengthy model training to a GPU; and (3) `eval_air`, in which the previously trained model is tested on a small fraction of the data. Similar scripts are available for the EPFL dataset we tested, as described in the README file[6].

D. Initial Models on the TOPO Dataset

Figure 4 provides a first impression from a reduced version of the Comballaz Air dataset: the attention module manages to draw more attention to stable features with texture such as the road and away from the sky, which seems promising. However, we have noticed that here and in some other images, there is also some attention drawn to unstable features in this context (*e.g.* forests that change with seasons). In addition, some stable features such as ridgelines aren't that prominent.

IV. Model Evaluation

A. Performance Discussion

In this Section, we are interested to find out how good the AtLoc-trained model performs on the Comballaz and EPFL datasets, regarding pose estimation and saliency maps.

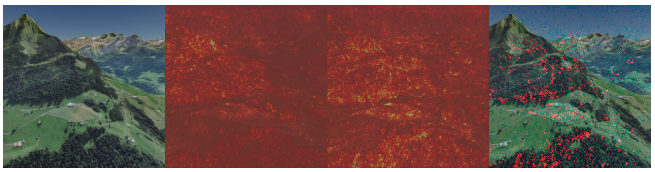

Figure 4. **Saliency Map Difference Before/After Training on Synthetic Image**: The first image is the original, the second represents the state at the beginning of training (output of the "visual encoder"), the third image is after 95 epochs of training (locally) using the first 210 pictures of the Comballaz Air dataset. The last image is a difference between the second and third, so it shows where training has been effective.

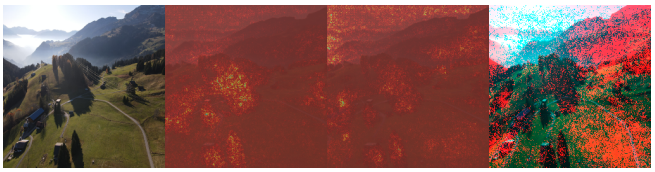

Figure 5. **Saliency Map Difference Before/After Training on Real Image**: The red channel in the last image represents the difference made by training, here, after 95 epochs.

When comparing Figures 4 and 5, it becomes apparent that training has much more effect on real images: this may be partly because we have trained the model with both real and synthetic images in this case, but real images are much more influenced by specific conditions, while synthetic images are much more stable. However, it appears that the

[2]This is the case, for example, in the Jura mountain range.

[3]https://github.com/BingCS/AtLoc

[4]IZAR is a brand new cluster with 136 GPUs as part of the SCITAS resources for HPC at EPFL.

[5]Given the time restrictions, we decided to focus on Comballaz Air and one EPFL flight from the 24th September 2020.

[6]https://github.com/CS-433/cs-433-project-2-eightyears

sky is considered too important, and this is also the case in other images. Nevertheless, we can observe that forests are still taken into account, but less, which is good since their appearance may vary across the seasons.

Finally, we would like to see the ridgelines gain in importance for the attention map, which is not the case here[7]. However, since quite some attention is directed towards the sky and not the rest, it still defines good contours for the mountains, which is a desirable property.

EPFL Phantom: What's about the EPFL images, the data is really different. For example, we have no sky or mountain edges, and since the pictures have been taken by a Phantom drone, they are all from above. As a result, the images contain mostly edges of buildings, grass areas or gray asphalt. As can be observed on Figure 6, the AtLoc model seems to focus its attention on areas with grass, while it *loses* the focus from buildings.

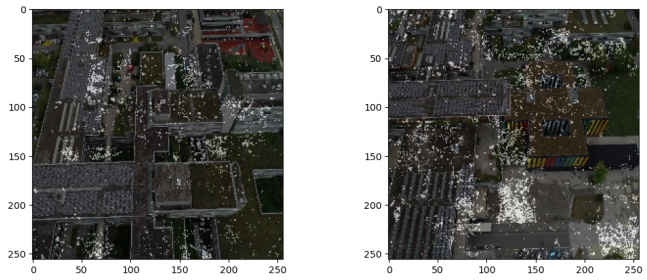

Figure 6. **Saliency Map Differences For EPFL Flight**: These differences correspond to the last image in Figures 4 and 5, *i.e.* the difference made by training. In particular, the highlighted points seem to be paid *less* attention to after training the model.

B. Results

Moving on to more quantitative results, we summarize in the Tables on Figures 7 and 8 the accuracies in different testing cases.

Training	**Synthetic Only**		**Real Only**	
Testing	**Synthetic**	**Real**	**Synthetic**	**Real**
	Comballaz Air			
Distance [meters]	51.68	255147.98	NA	2.04
Orientation [degrees]	118.22		NA	82.27

Figure 7. **Quantitative Results For Unmixed Training**: Medians for two possible scenarios: (1) training on synthetic (archive) data only, and (2) training on real data.

In these Tables, we observe that when training on both types of images, synthetic predictions are much better, while real data suffers from large inaccuracies. This is probably due to the small portion of real images with respect to the huge amount of synthetic data that can be generated.

[7]Better training could include more epochs, which could be explored by further research. We had to clamp our experiments due to the lengthy GPU script preparation and the ten-hour long execution…

Training	**Both Synthetic & Real**		**Both Synthetic & Real**	
Testing	**Synthetic**	**Real**	**Synthetic**	**Real**
	Comballaz Air		*EPFL 2020-09-24*	
Distance [meters]	69.74	29094.23	24.33	21724.34
Orientation [degrees]	144.80		110.76	

Figure 8. **Quantitative Results for Mixed Training**: Medians for training on both real and synthetic images, where we're mostly interested in the results for real data, given the drone use case.

Moreover, when training on 30'000 synthetic images from the Comballaz Archive, the synthetic accuracy is slighlty better than with mixed training, but not perfect either, probably since the testing images are taken at different poses. Additionally, the real accuracies drop as expected, since there are no real images in the training set.

This seems to imply that, at least with only 100 epochs, AtLoc is not able to extract common attention maps for real and synthetic images. Also, the split between real and fake images matters a lot, but it takes time to generate as many real images than there are fake ones.

So unfortunately, extrapolating to the real data does not seem so promising, at least in the Comballaz (Air) and EPFL (2020-09-24) datasets illustrated here. However, testing on real data when the model has been trained by real data yields very good accuracy, which seems nice, but requires that the drones only be put in 'known' areas, that is, locations for which PhD students from TOPO have manually created images.

C. Further Improvement

In order to have a closer match between real and synthetic data, the general idea was to have similar saliency maps, so that the attention is focused on the same features. To achieve this goal better than in this work, it should be possible to generate data, for instance by adding gray overlays to simulate clouds on synthetic landscapes, or by changing the colors from green to white to create a snow effect.

Some other ideas, as proposed by the TOPO lab, would be to use GANs to generate some (fake) real images to train from; or to dummify the images in order to keep only a low level of detail including the most important shapes.

V. Conclusion

To conclude, our end-to-end AtLoc network seems to produce more accurate results[8] than two-step networks as in [1], and our use case is no different. However, while we have had success in training Atloc on new datasets, either real or synthetic, we were unable to produce meaningful results using mixed training sets. We suspect that the chasm between the two datasets is currently too large but we believe that there is indeed potential to use data augmentation on the synthetic images as well as some preprocessing on the real images to make the two datasets more alike.

[8]But such networks have other drawbacks, like behaving as black-boxes.

ACKNOWLEDGEMENTS

The authors thank Iordan Doytchinov for his supervision and helpful suggestions during our work, and Jiannong Fang for his patient support when debugging our scripts for GPUs. Also, thanks a lot to the whole TOPO team, in particular Qi Yan for his previous work and Théophile Schenker for providing the dataset on IZAR.

REFERENCES

[1] Q. Yan, "OneShot Camera Pose Estimation for Synthetic RGB Images in Mountainous Terrain Scenes," 2020.

[2] B. Wang, C. Chen, C. X. Lu, P. Zhao, N. Trigoni, and A. Markham, "AtLoc: Attention Guided Camera Localization," *arXiv preprint:1909.03557*, 2019.

[3] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother, "DSAC – Differentiable RANSAC for Camera Localization," *arXiv eprint arXiv:1611.05705*, 2018.

[4] R. Silva and N. Richart, "Python Virtual Environments," 2020. [Online]. Available: https://scitas-data.epfl.ch/confluence/display/DOC/Python+Virtual+Environments

[5] V. Rezzonico and R. Silva, "File Systems," 2020. [Online]. Available: https://scitas-data.epfl.ch/confluence/display/DOC/File+systems

[6] V. Rezzonico and J.-C. De Giorgi, "Frequently Asked Questions," 2020. [Online]. Available: https://scitas-data.epfl.ch/confluence/display/DOC/FAQ

[7] N. Varini and N. Richart, "How to use Tensorflow on the Deneb GPU nodes," 2020. [Online]. Available: https://scitas-data.epfl.ch/confluence/display/DOC/How+to+use+Tensorflow+on+the+Deneb+GPU+nodes