Detection of Good Matching Areas Using Convolutional Neural Networks in Scene Matching-Based Navigation Systems

Ayham Shahoud [1], Dmitriy Shashev [1] and Stanislav Shidlovskiy [1]

[1] *Tomsk State University, 36 Lenin Ave, Tomsk, 634050, Russia*

Abstract

This paper presents a solution for false matching detection in scene matching-based aerial navigation systems. A navigation system that uses normalized cross-correlation to match a captured image with a reference image was designed. While traditional methods rely on statistical indicators to detect false matchings, this research relied on deep learning using Convolutional Neural Network (CNN). A CNN was trained to online predict the probability of a matching result to be true or false. The training dataset of images was constructed depending on the knowledge of where good matching areas are expected to be. The probability numbers were stored as an assistant map to be used again with the same reference map without classification. The system was implemented and tested in a 3D simulation environment using models for a drone, camera, and flight environment. The Robot Operating System (ROS) and the 3D dynamic simulator Gazebo were used for simulation. The results proved the efficiency of the proposed method in excluding the false matchings. Using the assistant map without classification resulted in an execution time of 41ms and RMS error of position less than 1.2m.

Keywords

Correlation, CNN, false matching, assistant map, potential fields, ROS, Tensorflow.

1. Introduction

Computer vision techniques are commonly used in drone navigation systems. They are also used for mapping, path tracking, and observation [1]. These systems are light, cheap, and do not rely on external systems. They can also offer a good solution to overcome the Global Positioning Systems (GPS) loss of signals. Computer vision navigation measurements can be absolute like in scene matching or relative like in visual odometry. The absolute position can be calculated by matching a captured image with a reference map. Image matching can be done using local features in the images or the cross-correlation function which is adopted in this research, cross-correlation function is a well-studied method for scene matching since the last century. The normalized cross-correlation has been used in many outdoor applications because of its resistance to illumination variation [2]. A lot of problems appear when talking about correlation-based scene matching like execution time, accuracy, and false matching.

The false matchings take place because of the noise, areas that produce a flattened correlation result, or other error sources [3]. In autonomous aerial navigation applications, large errors or error jumps are totally not allowed. These errors must be treated or excluded before being used by the autopilot. False matching may lead to large final errors, mission failures, or human damages. This work presents a solution for false matching detection in correlation-based navigation systems. The system was designed and tested using a drone in the 3D environment shown in Figure 1. A CNN was trained to online classify the areas where good and bad matchings are expected to occur. The classification results were saved to be used with the same reference image without the need to repeat the classification.

The rest of this paper is organized as follows: section 2 for related studies, section 3 for navigation algorithm, section 4 for true and false matching, section 5 for classification using CNN, section 6 for implementation, and section 7 for results analysis and conclusion.

GraphiCon 2021: 31st International Conference on Computer Graphics and Vision, September 27-30, 2021, Nizhny Novgorod, Russia
EMAIL: ayhams86@gmail.com (A. Shahoud); dshashev@mail.ru (D. Shashev); shidlovskiysv@mail.ru (S. Shidlovskiy)
ORCID: 0000-0002-5620-9984 (A. Shahoud); 0000-0002-9533-4577 (D. Shashev); 0000-0002-7541-9637 (S. Shidlovskiy)

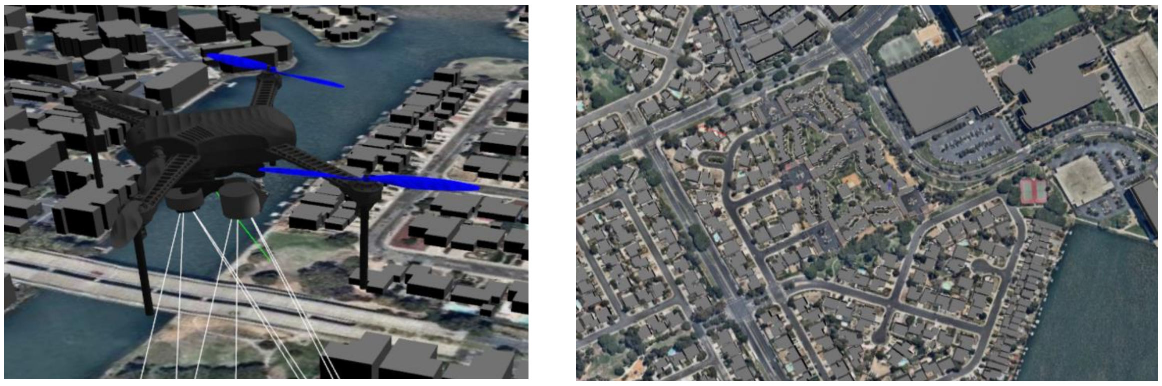

Figure 1: The 3D flight environment in Gazebo to the right. The IRIS drone model to the left with a camera fixed on it.

2. Related studies

Scene matching can be established using methods like local features and cross-correlation function. A lot of researches were done to enhance the reliability and accuracy of navigation systems based on cross-correlation. If accurate and trustable matchings are available, they will be useful in integrated navigation systems. They can be used to compensate for other navigation systems' errors like GPS, inertial systems, or visual odometry.

In reference [3], introduced an implementation of an integrated navigation system using inertial sensors, GPS, and computer vision. The vision navigation system depended on the cross-correlation, and the false matchings were detected and excluded using statistical analysis. An observation of the output variance over a sliding window for the last 30 measurements was done to detect false matchings. Good matchings were expected to occur in areas where road intersections existed. In reference [4], shown an image matching system for an autonomous Unmanned Aerial Vehicle (UAV) based on neural networks. A multi-layer neural network was used to detect the edges, and the position was calculated using cross-correlation. Matching edgy areas produced a reliable correlation result. Using mutual information between images to enhance the correlation results is presented in [5][6]. Map analysis based on image entropy could be used to select the best matching areas as in [7].

The use of an asymmetric two-branch CNN light model to predict the match probability of two images is presented in [8]. The matching network takes a pair of aerial images and road landmarks as input. The outputs are two depth features that were fed into a feature matching layer to compute the correlation map. In reference [9], scene matching between satellite images and online drone images was established using CNN. The contextual information extracted from the scene was used to attain increased localization accuracy and enable navigation without the use of GPS. A semantic shape matching algorithm is subsequently applied to extract and match meaningful shape information from both images.

The statistical solutions for detecting the false matchings may fail during maneuvering, or when the speed changes at a high rate. Relying on other sensors to solve the problem will be directly affected by their errors, and may increase the cost if high accurate sensors are needed.

In this research, an online solution that depends on artificial intelligence to detect good and bad matching areas was studied. Although it rises the execution time, it is more robust and innovative. A CNN was trained to predict the probability of an online captured image to have a true (or false) match on a reference map. The results were simply saved as an assistant map to be used with the same reference map for many purposes in the future. A dataset was constructed and used in training and validation. The navigation system decides according to the output of the CNN whether the correlation-based system output must be accepted or rejected. The resulted assistant map is not only beneficent in excluding false matchings, but it can be used to analyze the flight environment offline.

The assistant map can be considered as a meta-map that contains information about each pixel in the reference map. The "assistant map" name was adopted to refer to the map function, especially in the presented application.

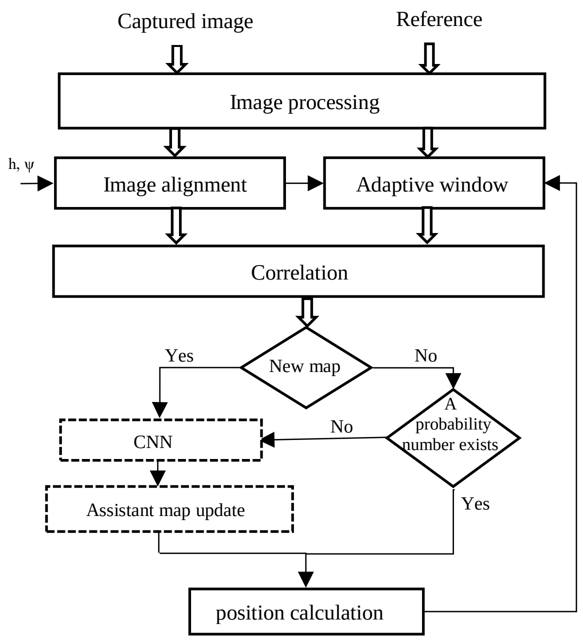

Figure 2: The navigation algorithm. The assistant map values will be updated just in case of no corresponding probability number or in case of a new map.

3. Navigation algorithm

A navigation system that depends on cross-correlation was designed. The position of the drone corresponds to the patch that produces the highest correlation value with the reference image. Before calculating the cross-correlation, the captured image must be aligned with the reference image i.e., rotated and scaled to match the reference image scale and orientation. The alignment process was done using the heading angle 'ψ' from the compass and the height 'h' from an altimeter. A detailed explanation of the navigation system implementation is presented in [10]. Let '*T*' be the captured image and '*I*' the reference image, the normalized cross-correlation is given in the following equation:

$$R(x,y) = \frac{\sum_{\grave{x}\grave{y}}(T(\grave{x},\grave{y}).I(x+\grave{x},y+\grave{y}))}{\sqrt{\sum_{\grave{x}\grave{y}}T(\grave{x},\grave{y})^2.\sum_{\grave{x}\grave{y}}I(x+\grave{x},y+\grave{y})^2}} \quad (1)$$

To reduce matching time, only a cropped window of the map was matched with the captured image. The size of the cropped window changes adaptively with the last movement amplitude and the scaled image size [10]. The cross-correlation process consumes less time than the classification process. That's why the classification was done after correlation, so maybe there will be no need for the classification step, for more details refer to section 5.4. The navigation algorithm is shown in Figure 2.

4. True matchings and false matchings

False matching means that for some reasons the maximum correlation value occurs in the wrong position on the reference image. False matching between an image and the reference image leads to jumps or large errors in positioning [3]. These errors may have dangerous outcomes in the case of autonomous vehicles. Destruction of the vehicle, mission failures, or damages to the surrounding environment might occur. There are many reasons why false matching occurs like image noise and areas that produce a flattened correlation such as soft intensive jungle.

The normalized cross-correlation is a mathematical operation that always will have results. Even if the images (matrices) are not identical or not similar to each other, the correlation will produce a result. The navigation system must decide if the matching result or the final position value is accepted or not. Artificial Intelligence (AI) offers a good innovative solution depending on the knowledge of where good and false matchings are expected to occur.

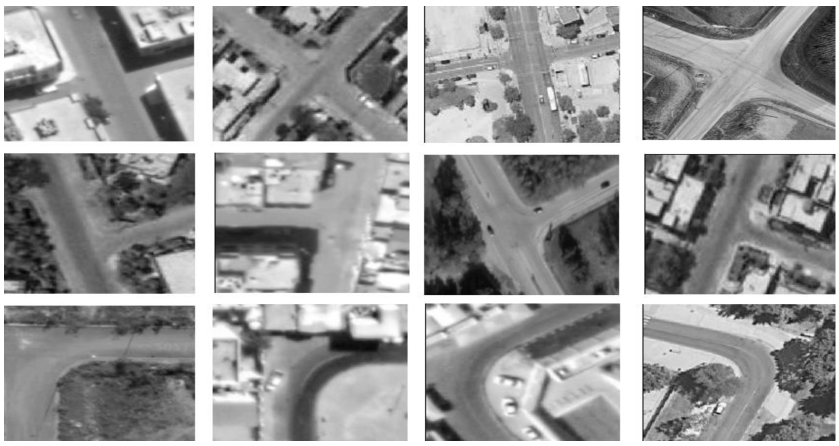
Figure 3: Examples for good matching areas in the constructed dataset from google maps.

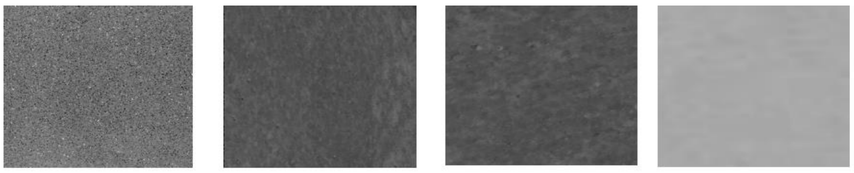
Figure 4: Examples for bad matching areas in the constructed dataset from google maps.

4.1. Good matching areas

Good matchings areas are places where the cross-correlation between a captured image and the reference image is expected to have a true matching. According to previous studies, these areas have a large probability to occur with the street (path) intersections. In general, they exist in relatively large geometrical intersection areas [8]. These areas are more stable and robust against the noise, weather, and illumination variance compared to small details in the image.

This research is interested in the city environment. The intersections in a city could be similar to the following shapes ╬, ╠, ╔, or may have other shapes with more arms like stars and polygonal. In real situations, they might be rotated, cropped, or warped under different models. The CNN must be able to recognize them by using a suitable training dataset. A set of good matching areas are shown in Figure 3. All pictures were taken from google maps.

4.2. Bad matching areas

They are areas with almost flattened correlation results i.e., no robust peak. Areas covered with snow, lake, large asphalt squares, intensive green jungle or grass, and homogeneous roofs might produce false matchings. In real work, the situation will be worse because of noise, illumination, weather, and sensor errors. A set of bad matching areas are shown in Figure 4.

5. Good and bad matching areas classification using CNN

5.1. CNN structure

Deep learning that depends on the convolutional neural network is very efficient in image classification. It is used for many applications like object detection, visual path tracking, and many other classification problems. The pre-processing required in a CNN is much lower compared to other classification algorithms. CNN takes an input image and assigns learnable filters to various features in the image, CNN has the ability to learn these filters [9][11]. A structure of a CNN is shown in Figure 5, it is used to classify the bad and good matching areas. A CNN consists of convolutional layers, pooling layers, fully connected layers, and activation functions. The feature extraction is done using convolution and some non-linear activation functions in the convolution layers.

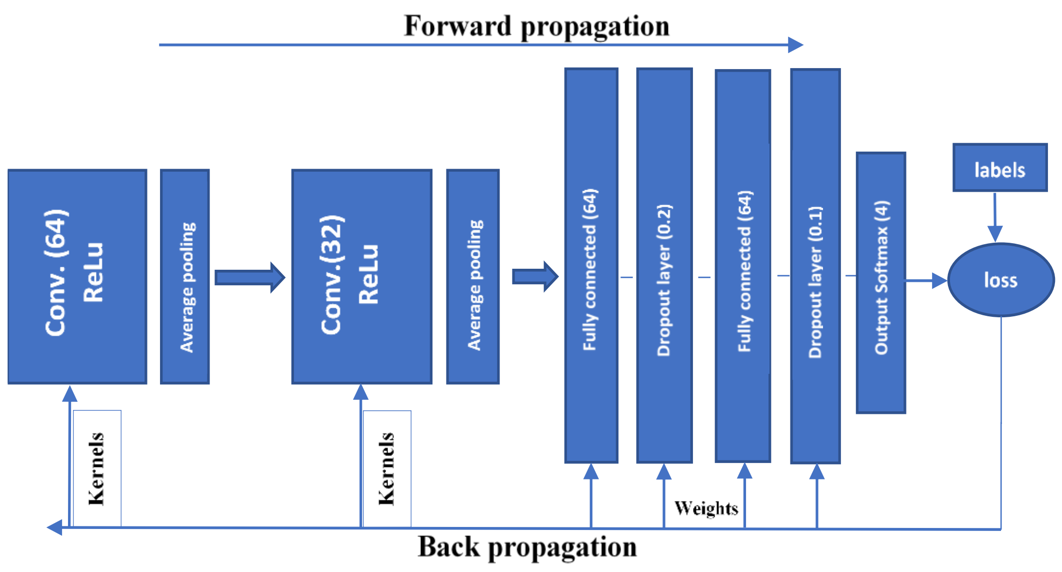

Figure 5: The designed CNN structure, conv. refers to a convolution layer.

Only kernels' elements (filters) that are used in the convolution operation are trained in the training phase. In the designed CNN, two convolution layers (64 and 32 kernels respectively) with a kernel size of 3x3 and the nonlinear ReLu activation function were used. Down-sampling operation which reduces the in-plane dimensionality is done in pooling layers. In this research, the features are not small details in the image, so an average-pooling of 2x2 was used after each convolution layer, which helped in the suppression of undesired small details in the images.

The outputs of the convolution layers are flattened and connected to a fully connected layer. To prevent overfitting, two dropout layers were used in addition to two fully connected layers. The final output layer used the Softmax function as an activation function. More details are in Figure 5.

5.2. Dataset creation

Using a down-looking camera simplified the task of collecting and creating the dataset from google maps. We chose 3 classes representing 3 types of intersections, and one class for bad matching. Increasing the number of classes that represent the good matchings may improve the results, but that needs a huge effort for building the dataset and training the CNN. Four classes gave acceptable results in a normal city environment. For cities with very complex geometrical characteristics, maybe we should think of increasing the number of classes.

The first class: bad matching areas similar to images shown in Figure 4 (for more about this class, refer to 5.4). The second class: good matching areas with intersections similar to ╬ i.e., a node with 4 arms. The third class: good matching areas with intersections similar to ╠ i.e., a node with 3 arms. The fourth class: good matching areas with intersections similar to ╔ i.e., a node with 2 arms. Of course, these shapes are ideal just for explanation. In Figure 3, shown real examples for them from the dataset.

A dataset of 3920 images was constructed. 1500 images were cropped from google maps, then augmented to 3920. The images were cropped from google maps for real intersections in cities. They were converted into grayscale, resized to 200x200 pixels, and stored as a labeled dataset.

5.3. CNN training and validation

A huge development was achieved in artificial intelligence tools that are used in training CNN, like Tensorflow and Keras. These tools were used in this research to train the CNN. The dataset was divided into a training dataset and a validation dataset. 80 % of the data used for the training and the rest for the validation. The accuracy and loss in training and validation are shown in Table 1. The final results proved the efficiency of the selected CNN structure and dataset.

Table 1
CNN training results

Parameter	Accuracy	Loss
Training	0.96	0.08
Validation	0.92	0.1

5.4. Navigation decision

In integrated navigation, error compensation does not require continuous scene matching. It is enough to have a number of trustable matchings according to the application. In standalone vision navigation, losing a matching will be a problem, and that is out of the scope of this work [8].

After the correlation process, the image is fed into the CNN as shown in Figure 2. If it is classified as a bad matching area (class number 1), the navigation system will not rely on the vision navigation system. The first class (bad matching areas) was used to reinforce the exclusion of such areas from being used in fixing integrated navigation solutions or from being used by the autopilot. These areas must be treated as "obstacles" for the drone. If a matching got a probability greater than 0.5 to be class 1, then no matter what the overall CNN result is, the patch will be excluded and considered false.

If the image is classified as a good matching area (class number 2, 3, or 4), and the probability to be class 1 was less than 0.5, then the navigation system will rely on the vision navigation system. The system must associate the probability (of having a true matching) to a predefined patch of the map that corresponds to the drone position, as shown in Figure 6.

The previous association could be done by building a corresponding map that contains the probability number of the pixel in the reference map. Each pixel on the constructed map (the assistant map) corresponds to the same pixel in the reference map. The corresponding patch size on the assistant map was fixed practically to be equal to twice the last movement amplitude in pixels (proportional to drone speed). In this work, the average speed of the drone was 10 m/s, and the execution time during classification was 140ms, so the patch size was approximated to 5 pixels (2.65m) centered by the drone pixel position (for more about the map specification, refer to 6.1).

Association overlaying might occur continually if the map is new, although that is not bad (just a recalculation depending on a new classification result), it is time-consuming. Overlaying can be avoided when using new maps by using a well-defined stride in the association, that's not adopted in this work.

There is no need to repeat CNN classification on the same map in the future. If the same map is used, then after the correlation, a check will be done to see if the probability number that corresponds to the drone position (on the map) exists on the assistant map. If the probability number does not exist, then classification is needed. The adopted method in saving results was easy to implement and the obtained assistant map was easy to analyze and visualize as an image or a matrix.

Analyzing the assistant map will be efficient to classify areas where robust visual navigation results are expected to be. To make the processing and visualizing easier, the results could be saved as "zeros" for the false matchings and "ones" for the true matchings according to a threshold of 0.5.

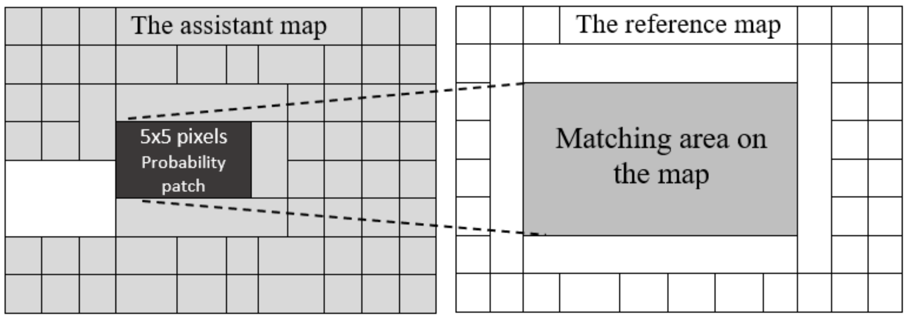

Figure 6: Saving the classification results. The black color on the assistant map corresponds to pixels in which if the drone is located, the camera is expected to capture a false matching, and the white color corresponds to the expectance of a true matching. Gray color refers to non-visited areas.

6. Implementation

6.1. The equipment

A computer with CPU i5-10300 - 2.5Ghz was used. The chosen camera model has a 60˚ field of view and an image size of 200x200. Tensorflow 2.4.1 and Keras 2.4.3 were used in training the CNN. All programs were written using Python 3.7 under Linux. OpenCV 3.4 library was used for image processing. A reference image with a resolution of 0.53 m/pixel and a size of 900x1000 pixels was used as a reference map. It was taken from a height of 500m in the environment.

6.2. Simulation environment

IRIS drone model from Ardupilot was selected, it was equipped with a camera, inertial sensors, compass, and GPS. A 3D city environment from Gazebo was used as a flight environment for the drone. Software In The Loop (SITL) was used to launch and control the drone trajectory. Subscribers were written to the camera images, compass, and to navigation solution published by ROS with MAVROS communication protocol. MAVROS is a middle protocol that translates the messages of the models into ROS messages. The navigation solution published by ROS was used as a reference path. The simulation environment was flexible in visualization, changing parameters, and repeating tests with zero cost.

6.3. Experiments

The drone was guided on a path in the 3D environment with an average speed of 10 m/s and a height of 100 m. Three flights were done on the same path. The first flight using online classification, the second flight using the created assistant map during the first flight, and the third flight using a statistical indicator to detect the false matchings. The adopted statistical method is similar to that used in [3], the variance of the position was observed over a sliding window of 30 measurements. The matching was considered false if the variance exceeded a predefined threshold.

The chosen environment intentionally contained areas where bad matchings are expected, as shown in Figure 1. The captured images were processed online to calculate the drone position. The position output was saved to ‘csv’ file and plotted using Octave. In the following figures, “scene matching” refers to the calculated path and “ref” to the reference path. In Figure 7, shown the position results in the first flight i.e., with online classification. In Figure 8, shown the statistical indicator with a threshold of 1.4 to indicate a false matching for variance values greater than it.

7. Results analysis and conclusion

7.1. Results analysis

A summary of the results is shown in Table 2. With online CNN classification, the execution time was 140ms while the RMS error of the position was 1.2m (disregarding the false matchings). The position error almost maintains itself using the assistant map, while the execution time becomes 41ms. Without false matching detection, the RMS error of position was 3.1m and the execution time was 40ms. The statistical method resulted in 1.5m RMS error of position and 41ms execution time.

Table 2

Navigation results

False matching exclusion method	RMS position error(m)	Execution time (ms)	Detected false matchings/overall (%)
Online CNN	1.2	140	98.7
With the assistant map	1.15	41	98.7
With Statistical indicator	1.5	41	112
Without false matching exclusion	3.1	40	-

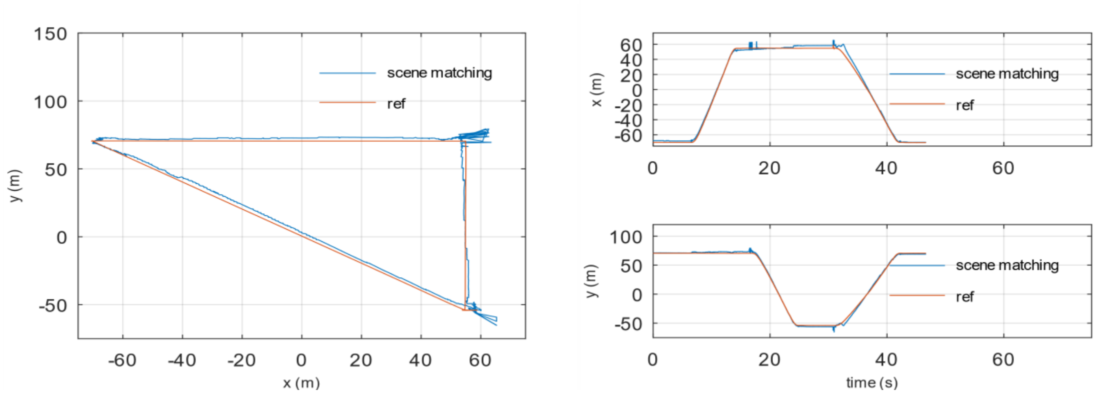

Figure 7: The calculated path in (x,y) plane with the reference path using online classification to the left. The calculated and reference position on the x-axis and y-axis to the right.

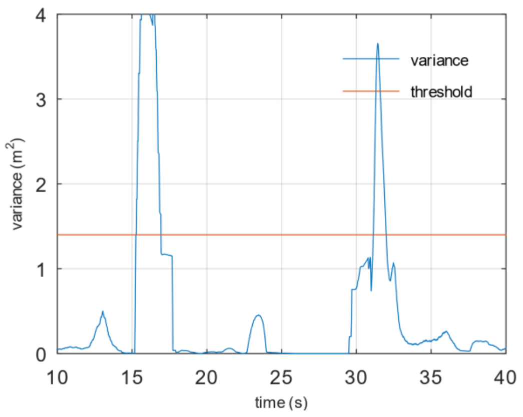

Figure 8: The calculated variance with a threshold of 1.4 to indicate false matchings.

The results were expected since CNN consumes a large time. Using the assistant map proved its efficiency regarding the execution time and accuracy. Excluding false matching suppressed the position error jumps and enhanced the reliability. Compared to statistical methods, an advance to CNN was noticed, the percentage of false matchings detected by CNN was more convenient. In the statistical method, a jump in the calculated variance corresponds to the false matchings was noticed. A latency appears in observing the false matchings because of the observation window size. This means that when bad matchings finish or start, the statistical indicator will not respond immediately. The 112% means that extra (not real) false matchings were detected because of the previous reason (latency). That happened even with the selection of small window size, to be able to sense the variance variation rapidly.

The RMS error of position was expected to be small (using any method) when excluding the false matchings. Losing a matching because it has a large probability to be false, has fewer negative outcomes than being accepted and used in navigation or control. In the CNN-based method, in addition to the trustable false matching exclusion, the assistant map was built which offers beneficent information about the environment.

7.2. Conclusion

This paper presented an implementation of a navigation system based on normalized cross-correlation. A new approach to online detection of good and bad matching areas was introduced and compared to a traditional statistical method. A CNN was trained to learn good and bad matching areas. A dataset of 3920 images was created to train and validate the CNN. Images for geometrical intersections from google maps were selected and divided into 4 classes, one class for bad matchings and the others for good matchings. The CNN classification results were saved as an assistant map with the same size as the reference map.

The created assistant map contains the probability numbers i.e., the probability of an image captured from the drone at that position to have true or false matching. Probability numbers were used to help the navigation system to decide if the vision navigation system output should be accepted or rejected. The final system resulted in a more accurate and robust navigation system compared to traditional false matching methods, but as expected with a larger execution time of 140ms in online detection mode. In the offline detection (with the assistant map) the execution time was the same as in traditional methods.

The stored assistant map can be used for many purposes. It can be used for the same previous purpose i.e., navigation decision assistant, without the need for classification. It can be used also to predict where a visual navigation system will be more effective by visualizing the assistant map before the mission. Analyzing the assistant map will helpful also in choosing the type of vision navigation system. For example, it might be found up that cross-correlation is not efficient and other methods like local features must be adopted. Sometimes it might be found up that standalone vision navigation is efficient without the need for integration with other systems.

In a standalone vision navigation system or generally, in critical missions, it might be necessary to consider the bad matching areas on the assistant map as obstacles. Instead of excluding these areas, they can be avoided by the drone guidance system using the potential fields method for example.

Classification of the best loitering and landing areas, topics we shall be focusing on in future work. Employing the assistant map in path planning is an interesting topic for future work also.

8. References

[1] Belmonte, L.M.; Morales, R.; Fernández-Caballero, A. Computer Vision in Autonomous Unmanned Aerial Vehicles—A Systematic Mapping Study, Applied Sciences. 2019, 9, 3196.

[2] Matuszewski, Jan & Grzywacz, Wojciech. (2017). Application of Discrete Cross-Correlation Function for Observational-Comparative Navigation System. Annual of Navigation. 24. 10.1515/aon-2017-0004. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp. 68–73.

[3] J. R. G. Braga, H. F. C. Velho, G. Conte, P. Doherty and É. H. Shiguemori, "An image matching system for autonomous UAV navigation based on neural network," 2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), Phuket, Thailand, 2016, pp. 1-6, doi: 10.1109/ICARCV.2016.7838775.

[4] A. Yol, B. Delabarre, A. Dame, J. Dartois and E. Marchand, "Vision-based absolute localization for unmanned aerial vehicles," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 2014, pp. 3429-3434, doi: 10.1109/IROS.2014.6943040.

[5] Tsvetkov, Oleg & Tananykina, L.V. (2015). A preprocessing method for correlation-extremal systems. Computer Optics. 39. 738-743. 10.18287/0134-2452-2015-39-5-738-743.

[6] X. Zhang, Z. He, Y. Liang and P. Zeng, "Selection Method for Scene Matching Area Based on Information Entropy," 2012 Fifth International Symposium on Computational Intelligence and Design, 2012, pp. 364-368, doi: 10.1109/ISCID.2012.98.

[7] G. Conte and P. Doherty, "An Integrated UAV Navigation System Based on Aerial Image Matching," 2008 IEEE Aerospace Conference, 2008, pp. 1-10, doi: 10.1109/AERO.2008.4526556.

[8] Y. Zhao and T. Wang, "A Lightweight Neural Network Framework for Cross-Domain Road Matching," 2019 Chinese Automation Congress (CAC), 2019, pp. 2973-2978, doi: 10.1109/CAC48633.2019.8996270.

[9] A. Nassar, K. Amer, R. ElHakim and M. ElHelw, "A Deep CNN-Based Framework For Enhanced Aerial Imagery Registration with Applications to UAV Geolocalization," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2018, pp. 1594-159410, doi: 10.1109/CVPRW.2018.00201.

[10] A. Shahoud, D. Shashev and S. Shidlovskiy, "Design of a Navigation System Based on Scene Matching and Software in the Loop Simulation," 2021 International Conference on Information Technology (ICIT), 2021, pp. 412-417, doi: 10.1109/ICIT52682.2021.9491778.

[11] Salman Khan; Hossein Rahmani; Syed Afaq Ali Shah; Mohammed Bennamoun; Gerard Medioni; Sven Dickinson, A Guide to Convolutional Neural Networks for Computer Vision, Morgan & Claypool, 2018.