Vehicle Offline Localization Based on Computer Vision: an Approach Based on Image Matching Retrieval Algorithms and Implementation

Alexey Kashevnik
SPC RAS
Saint Petersburg, Russia
alexey.kashevnik@iias.spb.su

Ammar Ali
ITMO University
Saint Petersburg, Russia
ammarali32@itmo.ru

Abstract—**Driver assistant systems have approved their essential role in increasing safety in the driving environment. One of their main features is the ability to provide information about location of the vehicle so that the driver can understand how and where to go. Moreover, automatically report the emergency in case of accidents to accelerate the rescue process. Unfortunately, poor Internet connection in many places and the inaccurate GPS information on traveling roads, in particular, make the localization process even harder. In this paper, we propose an offline localization system that can estimate the location of the vehicle in real-time using edge devices. Our proposed system consists of three main components; the first component is the image matching part to estimate the rotation and transition between two images; the second component is the dynamic objects (vehicles, pedestrians, and etc.) segmentation to filter out the matches from them; and finally, an image retrieval system to auto-correct the location using predefined coordinates of famous landmarks.**

I. Introduction

Offline localization or offline localization is an essential feature for driver monitoring systems and driver assistance systems because it helps to localize the vehicle even in places like forests, tunnels, and traveling roads where the GPS has low performance. This could be very critical, especially in case of automatic emergency calls. Multiple researchers were tackling this problem using the Simultaneous Localization and Mapping (SLAM) algorithm, which provides not only the location but also the mapping of the environment. SLAM is widely used in Mechatronics and robotics for indoor and outdoor environments, and it is used in self-driving vehicles. It depends on image matching to extract pairs of points between two images (could be a stereo camera or a monocular camera by taking two frames with a fixed period between them), then using the camera parameters and matches, we can calculate the rotation and transition (movements) as well we can build a 3D point cloud. We can use different optimizations to minimize error and drifting (triangulation, bundle adjustment, and inertial measurement unit reading).

In this paper, we present a method that is derived from the SLAM algorithm only for localization purposes. Our system uses only a single RGB camera image by taking frames with a distance of one second between them. We use image matching between every two sequential frames to extract the pairs of matched points. Using these points, we calculate the rotation and transition at each step. In parallel, we have an image retrieval neural network to extract image descriptors of famous landmarks, and we use this information to correct our constructed path.

The rest of the paper is organized as follows. Section II summarizes the state-of-the-art methods, which were used to estimate location using different algorithms. Section III introduces the proposed method in detail. It includes a general description, datasets used to train the neural network models, an image segmentation model for dynamic objects that can filter out inappropriate matches, a neural network for matched pairs between two frames, how to obtain the rotation and transition mathematically, and image retrieval model for auto-correction of the location if the driver passed by a well-known landmark. We present our current results in Section IV. Finally, the conclusion is in Section V.

II. Related Work

The research for developing offline localization systems or at least a helper system for GPS-denied environments is growing and spreading fast. Authors of the paper [1] proposed a framework for Google Maps to provide an estimated position in GPS-denied environments. The unmanned aerial vehicle (UAV) position is initialized by the correlation between frames, then they use optical flow to track the position through the frames, and a particle filter to obtain a coarse-to-fine search.

In the paper [2], authors proposed a semantic SLAM algorithm for a large-scale outdoor environment. They merged the ORB-SLAM point cloud with a semantic segmentation neural network (PSPNet-101) to get a 3D semantic map from which they were able to create a topological map by matching the real-world landmarks with the point cloud.

In another research [3], authors proposed a robust algorithm for extreme weather conditions. They use pose tracking, local mapping, loop closure, and pose graph optimization with a novel probabilistic point cloud generation and feature matching for radar images.

In the paper [4] authors proposed to use segmentation and object detection neural networks to detect the dynamic objects and remove them to minimize the error of estimating the location by excluding the moving points and considering them as outliers.

Researchers of the paper [5] used a laser scanner to build a 3D point cloud and a camera to detect the loop closure events using a novel appearance-based retrieval system. The frames used for loop closure were processed with their corresponding laser scans for the Euclidean image-to-image transformations to minimize the linearization errors.

Authors of the paper [6] proposed an improved correlative scan matching algorithm to enhance the performance and robustness of the scan matching module, an Adaboost-based loop closure detection, and a light-weight graph optimization algorithm.

Authors of the paper [7] provided a qualitative comparison between different SLAM algorithms (visual SLAM, Lidar SLAM, and sensor fusion SLAM). The explored different approaches for grid mapping (Dempster-Shafer, Bayesian, and Fuzzy Logic), and presented different localization estimation methods (probabilistic likelihood, edge or point features based direct scan matching techniques, particle filter).

ORB-SLAM is one of the most famous SLAM algorithms. Authors of the paper [8] released an algorithm that outperforms the state-of-the-art in more than 29 datasets. The proposed system works in real-time on CPUs and can handle monocular, stereo, and depth cameras. The proposed algorithm is based on bundle adjustment with monocular and stereo observations with a lightweight localization approach that allows zero-drift localization.

DSP-SLAM algorithm [9] proposed to enhance the joint-map of 3D dense models. It takes the 3D point cloud reconstructed by feature-based SLAM and provides dense reconstructions of detected objects. DSP-SLAM is a very suitable algorithm that could be integrated into our 3D vehicle detection, and segmentation system [10], but the negative side is the running time.

In the paper [11] authors proposed a novel approach for image matching by using cross-attention layers to obtain descriptors. Instead of the multi-stage method by performing feature extraction then matching. They establish pixel-wise dense matches at a coarse level at first, then refine matches.

Multiple researchers added more enhancements to the LoFTR architecture by proposing a quadtree attention layer [12] that was able to achieve better results not only for image matching but also for other tasks like (image classification, object detection, and segmentation). Another enhancement [13] used a transformer-based encoder, unlike Loftr, which uses a convolutional neural network (ResNet) as a feature extractor.

In the scope of the standard method that uses descriptors to find the matching pairs the descriptor could be extracted using basic image processing algorithms like scale-invariant feature transform (SIFT) or lightweight neural networks (like super-point). One of the most valuable matching neural networks is Super-Glue [14]. This neural network architecture based on the attention mechanism can reason about the underlying 3D scene and feature assignments jointly.

In the paper [15] authors proposed leveraging principles from reinforcement learning to optimize end-to-end a high number of correct feature matches.

III. Method

In the section we consider the general description of the proposed method for vehicle offline localization, present a dataset we used to train neural network models, and discuss three neural network models for image segmentation, matching and retrieval.

A. General Description

The proposed method for monocular localization depends on three neural networks, as shown in Fig. 1. We decided to take two sequential frames from our monocular system with a specific period between them. We use our segmentation models to generate masks of the dynamic object and pass the images with their masks to our image-matching system to remove the matched pairs from the dynamic objects. After that we calculate the rotation and transition between these two frames using Epipolar Geometry , and auto-correct the location if the second frame includes a landmark from our dataset.

1) The first neural network is used for image segmentation to remove dynamic objects like vehicles, pedestrians, and bicycles, if the matched pairs were taken from dynamic objects, then the rotation and transition will be calculated concerning that moving object leading to an enormous error in the estimated location.
2) The second neural network is used for image matching. It is responsible to find the matched points between two subsequential frames to calculate the rotation and transition for these points and construct the path using epipolar geometry.
3) The third neural network is used for image retrieval. It generates descriptors for landmarks and uses these descriptors to auto-correct the location and path.

B. Data

For model training, we used open-source datasets. The first dataset is Cityscapes [20] for image segmentation. It includes scenes for 50 cities with high-quality pixel-level annotations of 5000 frames. The second one is Megadepth dataset [21] for image matching. It consists from 196 different locations reconstructed from COLMAP SfM/MVS. The third one is Google Landmarks 2021 [22] for image retrieval. It includes more than 1.5 million images for famous landmarks. Together with that we have tested each developed component on our own dataset, which includes videos of a length of 20 seconds with five frames per second recorded using the developed Drive Safely system [17], [18]. To improve performance, we propose to prepare a ground truth for our data and to finetune these models on the ground truth dataset, rather than directly on our data.

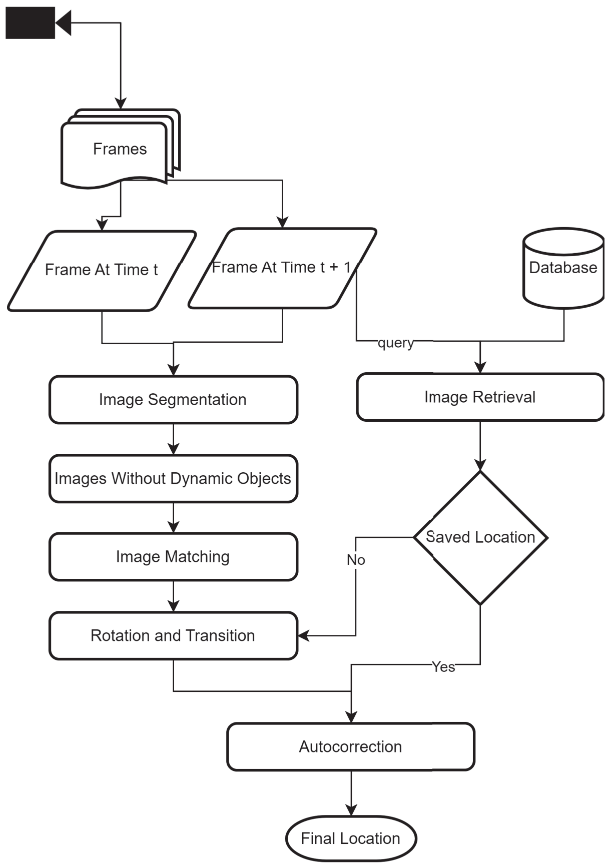


Fig. 1. Proposed method for vehicle offline localization

C. Image segmentation

For image segmentation, we used the current state of the art over multiple datasets OneFormer [19]. OneFormer architecture is based on a DiNAT_L transformer as a backbone to extract the features from the image, then a pixel decoder for mapping the embeddings to a multi-scale features domain. The model is trained to solve universal segmentation tasks (semantic segmentation, instance segmentation, and panoptic segmentation). Therefore the model takes two inputs. The first input is the image that we want to segment; the second is a text to define the required task. In our case, we are more interested in the instance segmentation task, because we want to remove the dynamic object from the image to exclude its effect on the matching process. Fig. 2 illustrates how OneFormer works on our dataset. These results are for instance segmentation (segmenting vehicles and removing them to exclude the points during the matching process).

D. Image Matching

For image matching, we are using LoFTR architecture [11], even though LoFTR QuadTree and match former outperform LoFTR in accuracy. Choosing LoFTR architecture was a suitable trade-off between the running time, cost, and accuracy. LoFTR does not follow the sequential approach for image matching, which involves detecting and describing

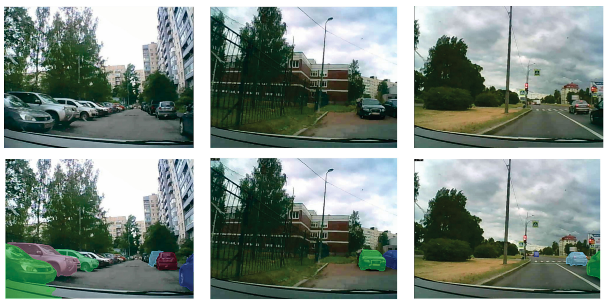


Fig. 2. Instance segmentation using OneFormer on our dataset

image features, and then matching them between images. Instead, LoFTR first establishes pixel-wise dense matches at a coarse level. It then refines these matches to identify the best matches between corresponding image regions. LoFTR also uses attention mechanisms to force the feature descriptors to be conditioned on both images. Fig. 3 illustrates the main four stages for LoFTR to obtain the matches between two images.

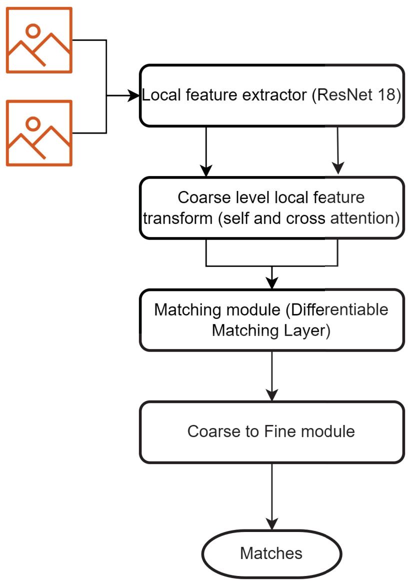

Local feature extractor (ResNet 18)
Coarse level local feature transform (self and cross attention)
Matching module (Differentiable Matching Layer)
Coarse to Fine module
Matches

Fig. 3. Main steps for LoFTR matching method

E. Geometric Model Estimation

After getting the matches from the LoFTR model and removing the points that are located on the dynamic objects, we used these matches to express the motion of the vehicle. We calculated the rotation and transition between frame_t1 and frame_t2. To do so, we firstly to calculated the Fundamental matrix.

1) Fundamental matrix: Let us denote a set of points on frame_t1 as X and the matches of these points on frame_t2 as X' then the Fundamental matrix F is a 3X3 matrix that satisfies:

$$XFX' = 0. \tag{1}$$

Unfortunately, the matches we get from LoFTR are not 100% perfect. Therefore to obtain the fundamental matrix, we solved this equation numerically to find the F matrix that satisfies the previous equation on the maximum number of pairs and considers them as inliers. Other matches will be considered outliers and will be removed. This problem could be solved using different methods, one of the most popular methods is RANSAC [23]. In our research, we have decided to use MAGSAC++ [16] because LoFTR showed the best results using MAGSAC++ algorithm on the Image Matching Challenge 2022 [24].

2) Essential matrix: The essential matrix is a special case of the fundamental matrix. So it can be expressed by:

$$YEY' = 0, \tag{2}$$

where E is the essential matrix, but the condition here applies to the coordinates. Y and Y' should correspond to the same 3D space. In general, we use the 3D camera space. So, let us denote the camera matrix as K then:

$$E = (K')^T FK. \tag{3}$$

3) Pose Estimation: The camera pose has 6-DoF (degree of freedom) rotation (roll, pitch, and yaw) and transition (x, y, z). We can calculate the camera pose from the essential matrix and the camera parameters, but mathematically we will get four different correct solutions. Unfortunately, only one of them is correct in the real world. Let us use Singular Value Decomposition (SVD) to write the essential matrix as the following:

$$E = UDV^T, \tag{4}$$

and let us define W as the rotation matrix around z axis with 90 degrees.

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Then we can calculate the four possible solutions as follows:

$$S1 = U(:,3) \ and \ R1 = UWV^t, \tag{6}$$

$$S2 = -U(:,3) \ and \ R2 = UWV^t, \tag{7}$$

$$S3 = U(:,3) \ and \ R3 = UW^tV^t, \tag{8}$$

$$S4 = -U(:,3) \ and \ R4 = UW^tV^t. \tag{9}$$

Then we substitute the previous possible values for S and R to get the possible solutions:

$$Pose = KR[I_{3X3} \ \ -S], \tag{10}$$

where S is the center of the camera. To choose the unique solution in reality, we need to check the chirality condition. To do so, we can use triangulation with the fact that the reconstructed points should be in front of the camera. A 3D point X is in front of the camera in case:

$$D > 0 \ where \ D = r3(X - S), \tag{11}$$

where $r3$ is the third row of the rotation matrix.

F. Image Retrieval

The idea beyond using such a system is to use predefined coordinates to correct the location. Image retrieval neural networks are responsible for representing the images as a set of features. So, when we pass a new picture of the particular object to the model, its representation on an N-dimensional space will be as close as possible to the feature extracted from the old image. These systems are popular and widely used, especially for face recognition tasks. We trained a neural network to extract features from famous landmarks. This way, we can auto-correct the location whenever the driver passed by a landmark on our database.
We have trained EfficientNetB3 architecture with noisy-student initial weights over the Google landmarks retrieval dataset. The neural network is responsible for extracting 1280 features from each image. We compare the extracted embedding with the database, and If we find a match, then, we correct the location (see Fig. 4).

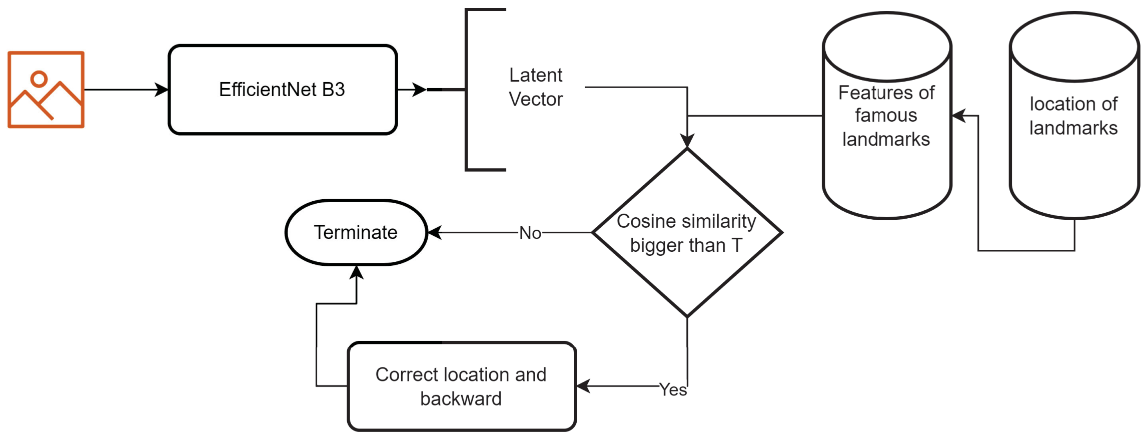
We used the Arcface layer while training instead of the standard Softmax layer because our goal is to maximize the margin between different landmarks as much as possible, which is the main difference between the proposed approach and any other classification problem. Instead of using Euclidean distance, Arcface uses geodesic distance on a hyper-sphere, and the goal is to maximize the angular (arc) margin between different landmarks.

$$L = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s\cdot(cos(\theta_{yi}+m))}}{e^{s\cdot(cos(\theta_{yi}+m))} + \sum_{j=1,j\neq yi}^{N} e^{s\cdot cos(\theta_j)}},$$

where L is the Arcface loss. In comparison with the Soft-Max loss, it changes the logit and has a better geometric interpretation. After training is finished with cross-entropy loss we remove these layers because we are interested in the embeddings. For each extracted vector feature, we compare it with a database of predefined landmarks features, using cosine similarity as in the inner-product space, this will measure the similarity, and if the two vectors are in the same direction, then the value is 1 if opposite the cosine similarity is -1, and in case of orthogonal vectors it will be 0. The cosine similarity between two vectors A and B is given:

$$Cos_{sim} = \frac{A \cdot B}{||A|| \cdot ||B||},$$

where A is the feature vector extracted from the query image and B is the feature vector from the database, therefor to find the best match we need to find $Max(Cos_{sim}) \ for \ each \ B \ in \ D$ where D is a set of features extracted from images for landmarks (for our own dataset).



EfficientNet B3
Latent Vector
Features of famous landmarks
location of landmarks
Cosine similarity bigger than T
No
Terminate
Yes
Correct location and backward

Fig. 4. Diagram shows the process of adding the retrieval NN for auto-correction. T is a threshold that should be adapted according to the used device

IV. Results

In this section, we present the results of the developed method evaluation separately for each module.

A. Technical Details

We have resized the images to (max size 840 and kept the aspect ratio). Training and testing were done on RTX 3090 with 24 GB VRAM and CPU Intel core i9 12th generation with 128 RAM.Not all were used, our proposed system can work on Jetson Nano devices with acceptable latency.

B. Image Matching

Firstly we analyze how LoFTR works before adding the segmentation models. Fig. 5 shows the results,The blue points represent the potential pairs of matching between the two input images, green lines are the matched points after applying MAGSAC++ and filtering the outliers. The output is not bad but still not so good because of two reasons:

1) There are matching points between vehicles, and even though this could be helpful when the vehicles are parking (static objects), it will lead to worse results in the case of moving vehicles (dynamic objects). Therefore we added the segmentation model to remove these matches;
2) There are matching pairs inside the main vehicle because the camera is placed inside. To solve this issue, we should remove this part or dismiss these matches.

C. Image Matching with Segmentation Filtering

The results of this stage are the outputs of combining two components, the first part is to use segmentation to detect the vehicles and generate masks which will be passed to the LoFTR to filter out the undesirable matches (as shown in Fig. 6 and 7), passing the segmentation masks to LoFTR resulting on no blue points over dynamic objects (in this case cars), and therefore no matching lines between the images that are located on the dynamic objects.

We are thus able to exclude all undesirable matching (matches located on dynamic objects or from inside the

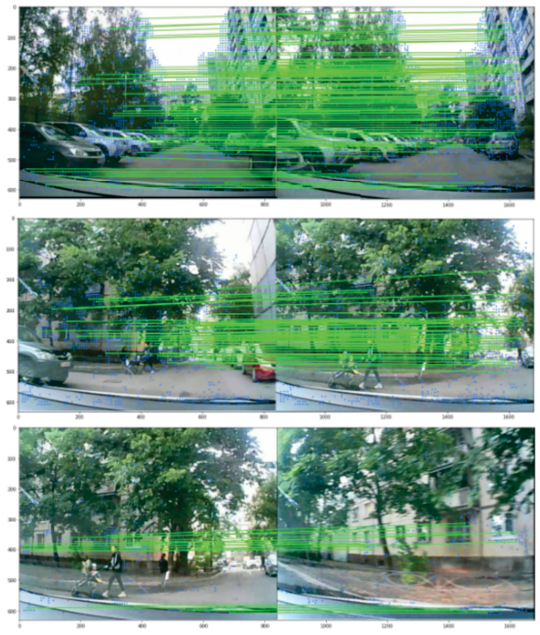


Fig. 5. Example of how LoFTR works on our data

vehicle). This provides us with a more stable set of pairs to calculate the rotation and transition between two images. Continuing with the example in Fig. 6, let us take the third pair of images and calculate the fundamental matrix using MAGSAC++ with an error smaller than 0.1 and $1e^5$ Iteration maximum. The result is:

$$F = \begin{bmatrix} 1.5e^{-07} & -5.8e^{-06} & 2.5e^{-03} \\ 1.2e^{-06} & 5.9e^{-07} & 5.1e^{-03} \\ -4.2e^{-04} & -4.6e^{-03} & -3.2e^{-01} \end{bmatrix}$$

Now we can calculate the essential matrix:

$$E = K'^{T} F K = \begin{bmatrix} 2.7e^{-07} & -4.6e^{-06} & 1.8e^{-03} \\ 3.6e^{-06} & 3.1e^{-06} & 1.2e^{-05} \\ -1.4e^{-03} & -1.2e^{-03} & -2.8e^{-06} \end{bmatrix}$$

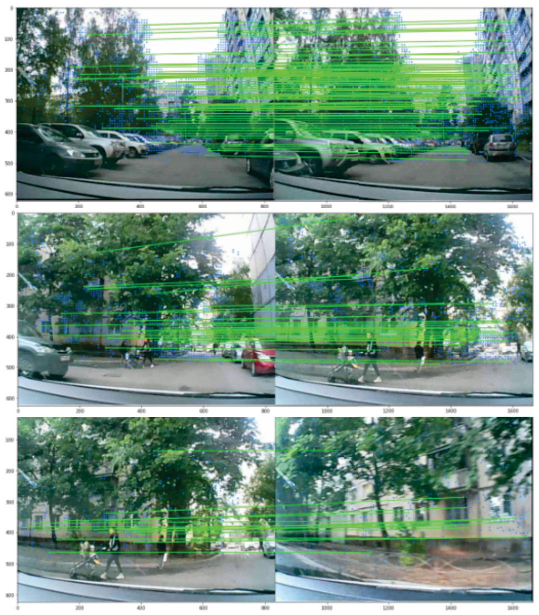


Fig. 6. Results of filtering the undesirable matching using oneformer

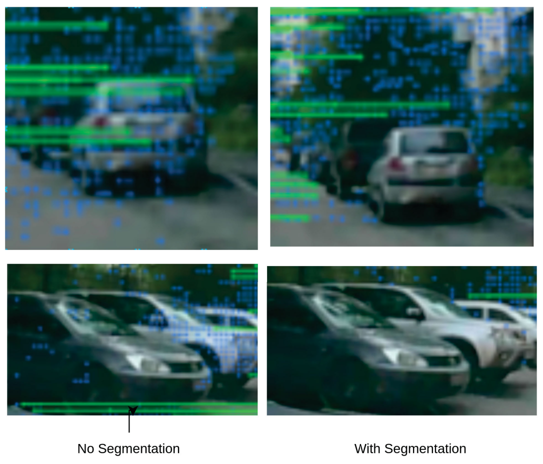


Fig. 7. Filtering the undesirable matching using segmentation on specific regions

From the essential matrix we can calculate the transition and rotation we obtain Euler angles in degrees: 0.03 degrees around the x-axis, 0.09 degrees around the y-axis, and -40.0 degrees around the z-axis, which means that the vehicle rotated by 40 degrees anticlockwise, and if you took a look at the two images, it seems to be correct. The transition was estimated at one meter. Now if we took the pair in the middle in Fig. 6, the rotation is estimated at almost zero degrees around the x, and y-axis and -4 degrees around the z-axis, with transition 0.77 meters.

D. Image Retrieval

As we mentioned earlier our model was trained using the Google Landmarks Retrieval Dataset. For testing we have collected images of famous landmarks in Saint-Petersburg (see Fig. 8).

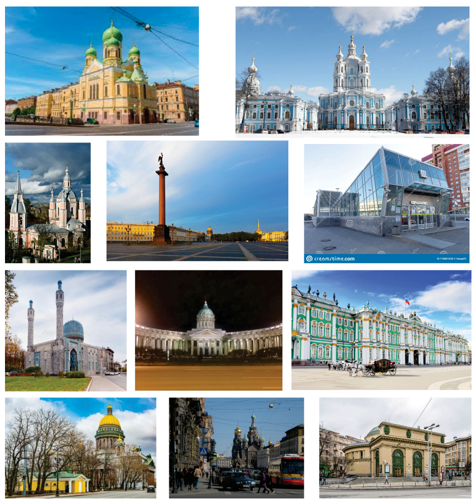


Fig. 8. Landmarks Example in our dataset For St. Petersburg city

Our model was able to achieve 35% mAP@100 metric on the Google Landmarks Retrieval Dataset. Unfortunately, because we still don't have a large dataset for landmarks, the training was done completely on Google landmarks data, then we took a small sample of our data (famous landmarks in Saint Petersburg) and tested the model on it to test the generalization, robustness, and stability. To show how the neural network works on our collected landmarks, let us take a landmark from the street view as a query image and calculate the cosine similarities with the extracted embedding. In Fig. 9, we can see the matches found using our system. The first two green arrows show a correct match, and for the third query image, we can see that the system failed to find the correct match.

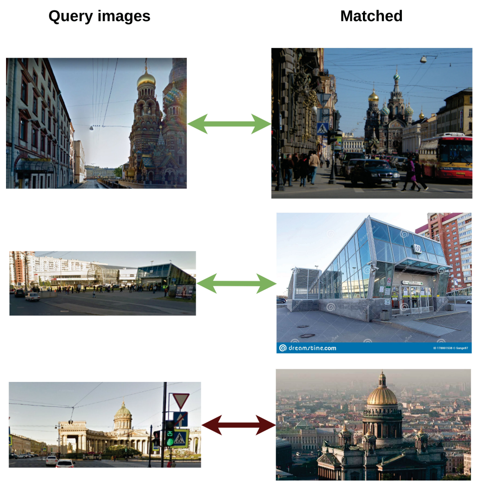


Fig. 9. Results example of image retrieval neural network for three different queries

V. Conclusion

In this paper, we proposed a method for estimating the vehicle location in case of GPS connection is lost using a monocular RGB camera by taking sequential frames from the video with a time difference of one second. The proposed system consists of three neural networks:

1) the first neural network we used for image segmentation to filter out the dynamic object in the scene (pedestrians, other vehicles, bicycles);
2) we used LoFTR neural network to find the matching pairs between images, and then use these matches to calculate the rotation and transition;
3) the retrieval part is used to auto-correct the location in case the driver passed by a known landmark.

There is still room for improving this work, by expanding our retrieval dataset and fine-tuning the model on it. We will have a huge impact on the auto-correction part. Merging the system with our monocular depth estimation system and 3D mapping will optimize the matching process and help further to get better estimation results for the rotation and transition. Moreover we are planning to optimize LoFTR architecture to run faster on edge devices.

Acknowledgement

This work was supported by the Russian Science Foundation (project 18-71-10065). Experiments have been implemented in scope of Russian Foundation for Basic Research project 19-29-06099.

References

[1] M. Shan, F. Wang, F. Lin, Z. Gao, Y. Z. Tang and B. M. Chen, ”Google map aided visual navigation for UAVs in GPS-denied environment,” 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, China, 2015, pp. 114-119, doi: 10.1109/ROBIO.2015.7418753.

[2] Zhao, Zirui Neil et al. “Visual-Based Semantic SLAM with Landmarks for Large-Scale Outdoor Environment.” 2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI) (2019): 149-154.

[3] Z. Hong, Y. Petillot and S. Wang, ”RadarSLAM: Radar based Large-Scale SLAM in All Weathers,” 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 5164-5170, doi: 10.1109/IROS45743.2020.9341287.

[4] D. Esparza and G. Flores, ”The STDyn-SLAM: A Stereo Vision and Semantic Segmentation Approach for VSLAM in Dynamic Outdoor Environments,” in IEEE Access, vol. 10, pp. 18201-18209, 2022, doi: 10.1109/ACCESS.2022.3149885.

[5] P. Newman, D. Cole and K. Ho, ”Outdoor SLAM using visual appearance and laser ranging,” Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., Orlando, FL, USA, 2006, pp. 1180-1187, doi: 10.1109/ROBOT.2006.1641869.

[6] Ren, Ruike, Hao Fu, and Meiping Wu. 2019. ”Large-Scale Outdoor SLAM Based on 2D Lidar” Electronics 8, no. 6: 613. https://doi.org/10.3390/electronics8060613

[7] Lu, Z., Hu, Z., Uchimura, K. (2009). SLAM Estimation in Dynamic Outdoor Environments: A Review. In: Xie, M., Xiong, Y., Xiong, C., Liu, H., Hu, Z. (eds) Intelligent Robotics and Applications. ICIRA 2009. Lecture Notes in Computer Science(), vol 5928. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-10817-4_25

[8] R. Mur-Artal and J. D. Tardós, ”ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras,” in IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255-1262, Oct. 2017, doi: 10.1109/TRO.2017.2705103.

[9] J. Wang, M. Rünz and L. Agapito, ”DSP-SLAM: Object Oriented SLAM with Deep Shape Priors,” 2021 International Conference on 3D Vision (3DV), London, United Kingdom, 2021, pp. 1362-1371, doi: 10.1109/3DV53792.2021.00143.

[10] Kashevnik A, Ali A. 3D Vehicle Detection and Segmentation Based on EfficientNetB3 and CenterNet Residual Blocks. Sensors. 2022; 22(20):7990. https://doi.org/10.3390/s22207990

[11] J. Sun, Z. Shen, Y. Wang, H. Bao and X. Zhou, ”LoFTR: Detector-Free Local Feature Matching with Transformers,” 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 2021, pp. 8918-8927, doi: 10.1109/CVPR46437.2021.00881.

[12] T. Shitao, Z. Jiahui, Z. Siyu, T. Ping, ”QuadTree Attention for Vision Transformers,” 2022 ICLR

[13] W. Qing, Z., Jiaming, Y. Kailun, P. Kunyu, S. Rainer, ”MatchFormer: Interleaving Attention in Transformers for Feature Matching”, 2022 Asian Conference on Computer Vision.

[14] P. -E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, ”SuperGlue: Learning Feature Matching With Graph Neural Networks,” 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 4937-4946, doi: 10.1109/CVPR42600.2020.00499.

[15] Tyszkiewicz M, Fua P, Trulls E. DISK: Learning local features with policy gradient. Advances in Neural Information Processing Systems. 2020;33:14254-65.

[16] D. Baráth, J. Noskova, M. Ivashechkin and J. Matas, ”MAGSAC++, a Fast, Reliable and Accurate Robust Estimator,” 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 1301-1309, doi: 10.1109/CVPR42600.2020.00138.

[17] Kashevnik A., Lashkov I., Gurtov A. Methodology and Mobile Application for Driver Behavior Analysis and Accident Prevention. IEEE Transactions on Intelligent Transportation Systems, IEEE. 2019. Vol. 21(6). P. 2427–2436

[18] Kashevnik A., Lashkov I., Ponomarev A., Teslya N., Gurtov A. Cloud-Based Driver Monitoring System Using a Smartphone. IEEE Sensors, IEEE. 2020. Vol. 20(12). P. 6701–6715.

[19] J. Jain , J. Li , M. Chiu , A. Hassani , N. Orlov , H. Shi, ”OneFormer: One Transformer to Rule Universal Image Segmentation”, 2022, arXiv:2211.06220v2, https://doi.org/10.48550/arXiv.2211.06220

[20] https://www.cityscapes-dataset.com/

[21] http://www.cs.cornell.edu/projects/megadepth/

[22] https://www.kaggle.com/c/landmark-retrieval-2021

[23] Martin A. Fischler and Robert C. Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24, 6 (June 1981), 381–395. https://doi.org/10.1145/358669.358692

[24] https://www.kaggle.com/competitions/image-matching-challenge-2022/data