УДК 629.7.05

E. RULKO

TERRAIN RELATIVE NAVIGATION BASED ON DEEP FEATURE TEMPLATE MATCHING AND VISUAL ODOMETRY

Military Academy of the Republic of Belarus

The main hurdle for terrain relative navigation systems is the incongruity of visual features between a patch of a satellite reference map and a view from an onboard UAV camera. Images are taken during different time of year, under different weather, vegetation and lighting conditions, with different angles of observation. This work proposes the usage of deep feature template matching, where features are extracted during unsupervised training using a triplet loss. It provides semantic understanding, agnostic to terrain transformations. In order to overcome struggling to navigate over featureless terrains, the work proposes additional usage of visual odometry with the procedure of sticking to the map after encountering enough features, with the procedure of hypothesizing over possible locations. Passing a fragment of the reference map through the trained feature extractor, applying an entropy filter and then a pathfinding algorithm allows planning a flying path over areas rich of features relevant for navigation.

Keywords: *terrain relative navigation, template matching, triplet loss, deep learning, visual odometry, UAV.*

Introduction

Current proliferation and accessibility of open digital maps that include satellite images in confluence with a plethora of open source deep learning based solutions of performing feature extraction, visual odometry (VO) and simultaneous localization and mapping (SLAM) is the impetus of developing manifold terrain relative navigation systems (TRN).

The image of a terrain fragment on a map usually has been taken during different time of year, under different weather, vegetation and lighting conditions, and with different angles of observation in comparison with a current camera view. Some objects on maps like spots of light, reflected from iridescent objects, or cars on roads must be discarded from being used as landmarks and flying over a field, forest or some snowy surface may not present enough deep features to stick to a reference map. All this represent a palpable hurdle to existing TRNs. To address these issues this work proposes the usage of deep feature template matching, where the features are extracted during unsupervised training using a triplet loss over different terrain patches. This allows performing matching of terrain patches based on their semantic content with the imparted knowledge of a neural network about possible transformations of terrains over different time and conditions, including understanding of features that should be discarded. Combination of this solution with deep learning based VO allows performing navigation when a terrain doesn’t have enough deep features to stick to the reference map.

Additionally, having a model that is capable of extracting deep features relative for navigation over a satellite map allows applying an entropy filter to those extracted features and to consequently determine routes of a UAV with enough visual features to stick to.

Related Work

Among multifarious solutions of TRN systems, several works represent especial interest in terms of using deep learning for extracting weather, time of year, lightning conditions agnostic features. An approach based on autoencoder architecture is presented in [1]. During the first stage of training the network takes a tile of a satellite image (ortho tile) and reconstructs a corresponding tile of map's abstract layer. The second stage of training implies the usage of given latent space to reconstruct the original tile acquired from custom video – Figure 1.

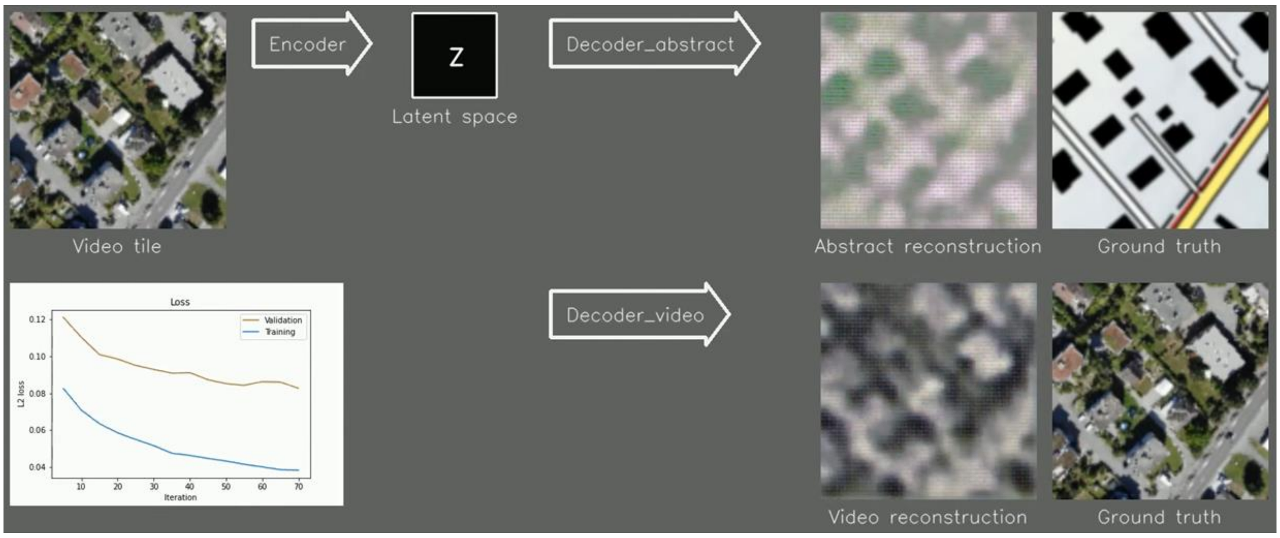

Figure 1: Mechanism of getting latent space [1]

The process of locating involves rotating, scaling images and cropping 5 tiles for a single iteration of template matching – Figure 2.

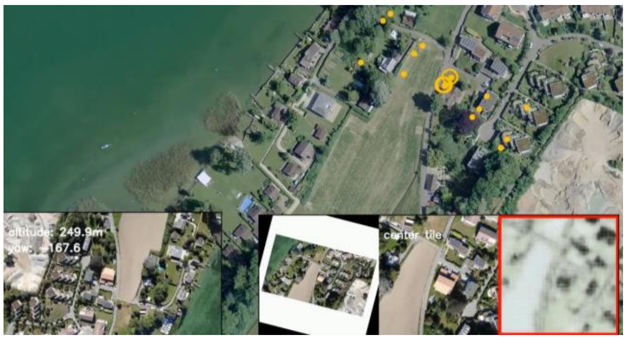

Figure 2: Navigation process [1]

As the demonstrated experiments indicate, the approach works well when flying over a terrain that has rich feature representation on a correspondent abstract map, like a city landscape, because the neural network learns how to produce an abstract map tile from a satellite one. It seems pretty difficult for this approach to move over some open area of fields which will be represented as just featureless white space.

A seasonally invariant deep transform for visual terrain relative navigation is presented in [2]. As the name suggest it involves targeted use of deep learning within an image transform architecture, which converts seasonal imagery to a stable, invariant domain that can be used by conventional algorithms (like homography via feature matching from *OpenCV*) without modification. During training, a *U-Net* like image transform model is exposed to matching cross-seasonal image pairs in twinned fashion – a single transform is identically shared between two parallel streams, with registration performance between the outputs used as a loss function to optimize the transform weights – Figure 3.

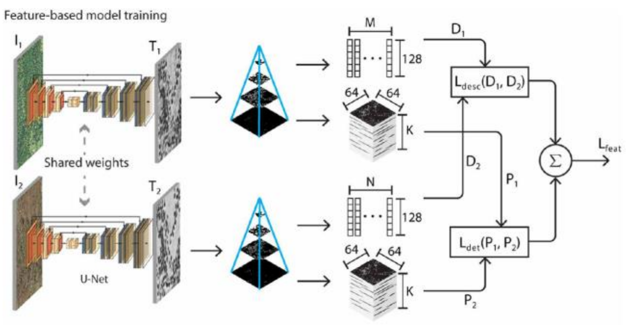

Figure 3: Season agnostic training process [2]

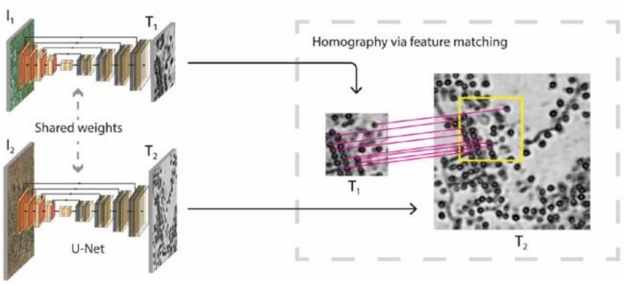

Figure 4: Inference preprocessing and matching [2]

During the inference stage the reference map image and a camera view image are subjected to the learned transformation and the resultant images allow using homography via feature matching in an ordinary way – Figure 4.

According to the work [2] this approach demonstrates superior performance under extreme seasonal changes while also being easy to train and highly generalizable. As the result of preprocessing, images of summer and spring mountain terrain will be transformed into pictures of ridges – Figure 5.

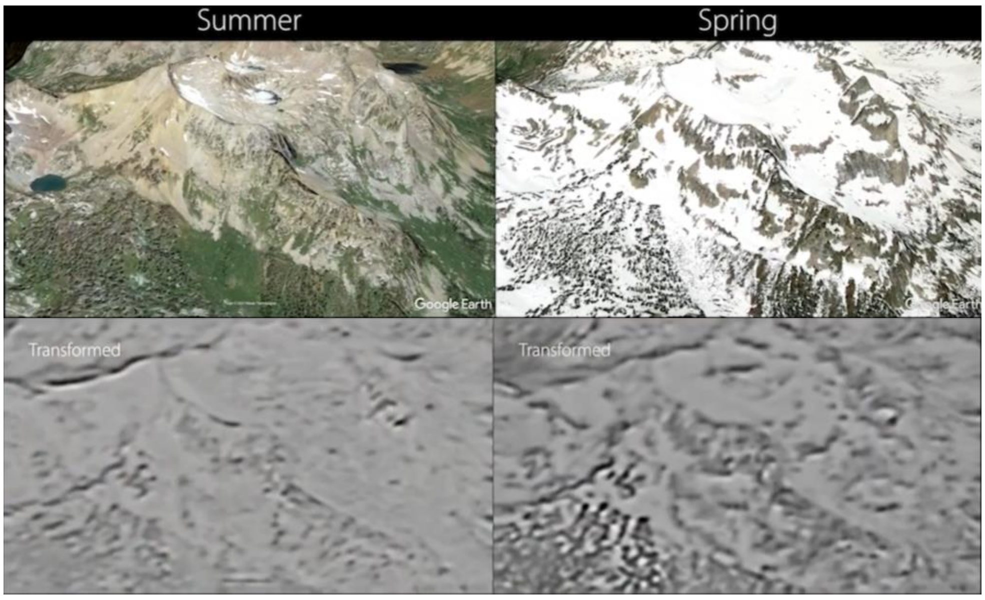

Figure 5: Season agnostic transformation [3]

However, the open example of the approach usage demonstrates that it requires relatively high altitude to get enough landmarks like buildings or mountain terrain features and due to the incongruity in angles of observation on satellite maps some objects, especially high buildings, appear different to a degree that their outlines won’t match while using conventional feature matching, without taking into account semantic context – Figure 6.

It’s also necessary to mention that the major providers of open maps such as Google, Bing or Yandex usually present a terrain over the Republic of Belarus during seasons without extensive snow coverage, which represent some hurdle for the aforementioned approach [2] and the approach that will be presented in the current work.

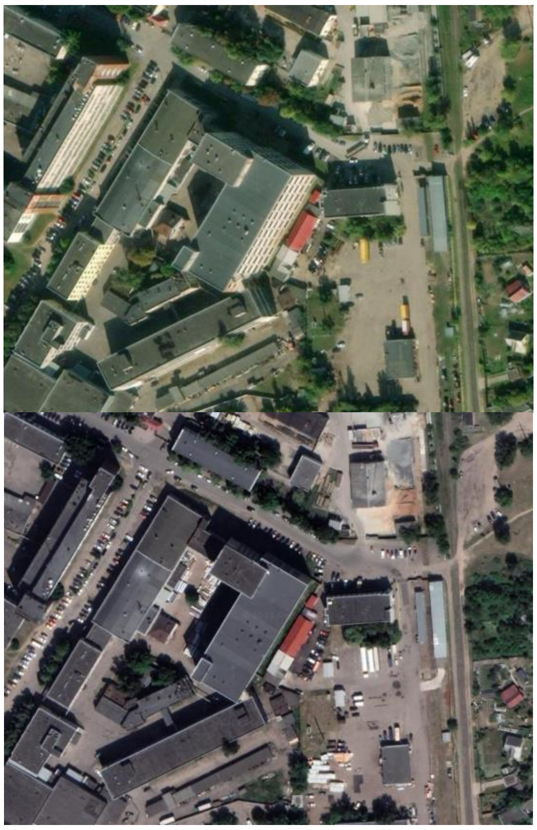
Figure 6: The same place in Minsk city from Bing (above) and Google (below) providers

But it can be surmountable by using for training existing terrains from other places of the world or by different map providers, which contain examples of extensive pieces of land with snow coverage that can also be found on satellite maps without it.

Modern TRN systems also extensively use existing open source keypoint matchers. The advantage of deep learning based matchers such as *LoFTR* (Local Feature Matching with Transformers) [4] over ordinary keypoint detectors such as *OpenCV ORB* [5] was presented in [6] – Figure 7. Other algorithms from *OpenCV* such as *SIFT*, *SURF* or *BRIEF* don’t seem to work much better, whereas there is a plethora of opens source deep learning based matchers, differing in year of inception, accuracy and performance, such as: *KP2D* (Neural Outlier Rejection for Self-Supervised Keypoint Learning) [7], *SuperGlue* (Graph Neural Network combined with an Optimal Matching layer) [8], *Efficient-LoFTR* [9], *LightGlue* [10] and other. Some of solutions are just modified and refined versions of previous ones. During the research conducted within the framework of this work, the majority of solutions were tested in practice, relatively to the TRN task.

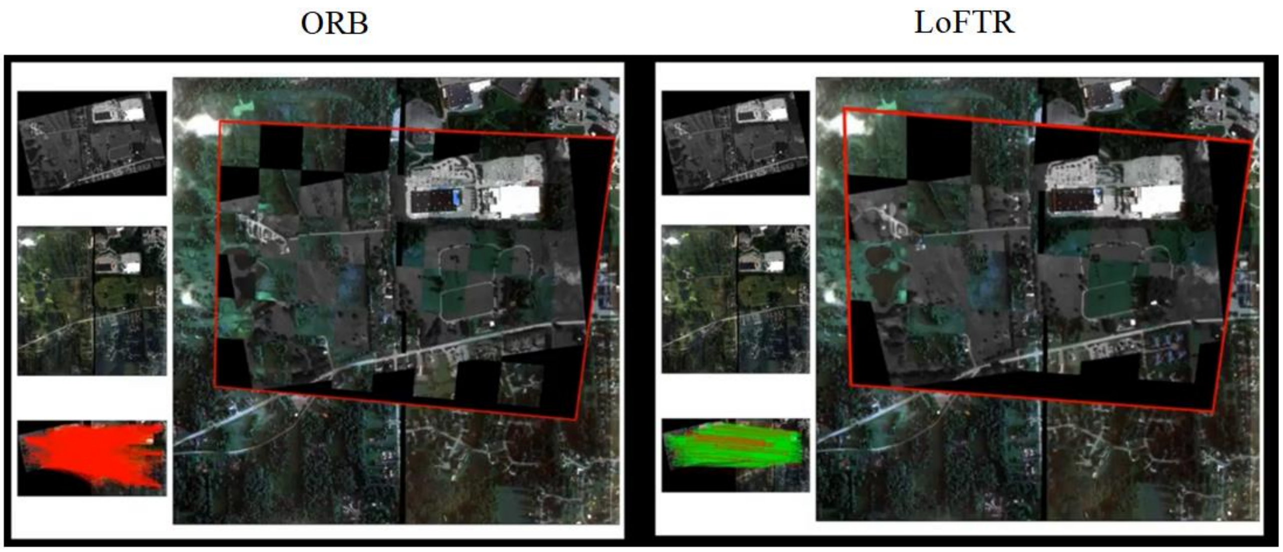

Figure 7: Comparison in precision with flying over a real terrain [6]

Especially apt for that in terms of performance and accuracy was *GlueStick* [11], which provides robust image matching by sticking points and lines together. Lines are especially important while using manmade landmarks – Figure 8*a*. At the same time it also extracts point based features – Figure 8*b*. This allows feature matching (Figure 10*c*) and subsequent homography alignment (Figure 10*d*).

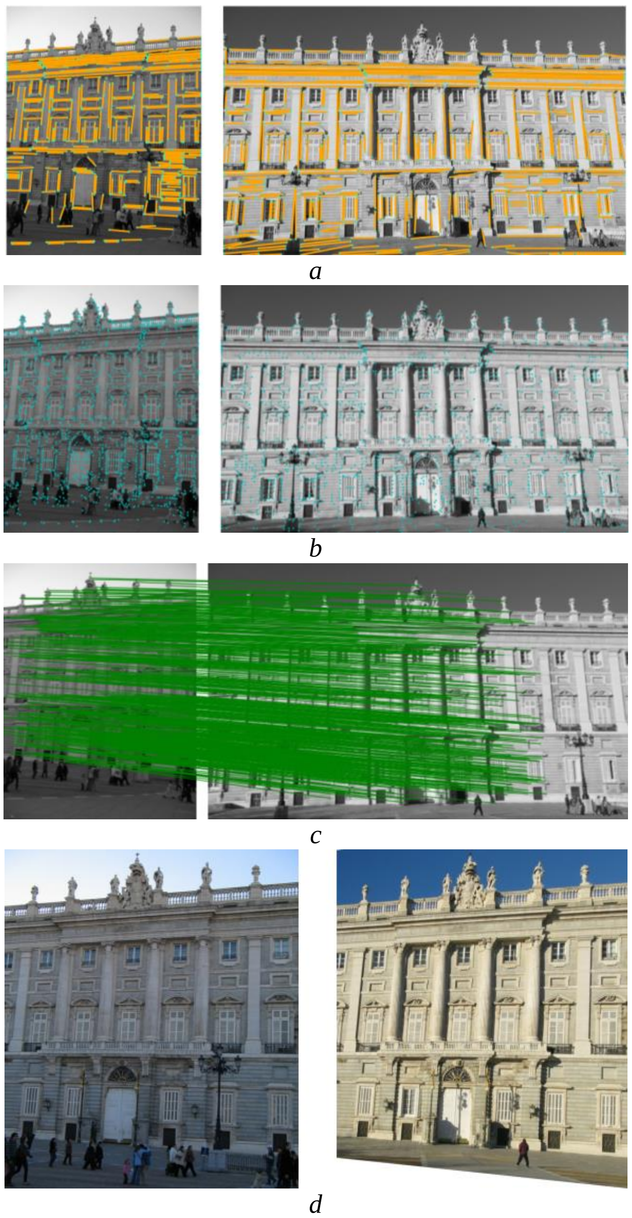

Figure 8: Matching using *GlueStick* [11]

However, the usage of *GlueStick* per se for matching two patches of terrain from different sources wasn't so robust (Figure 9*a*), whereas it has demonstrated relatively high precision for the purpose of VO, when it's necessary to keep track of the same features viewed from different angles over time due to the process of flying (Figure 9*b*).

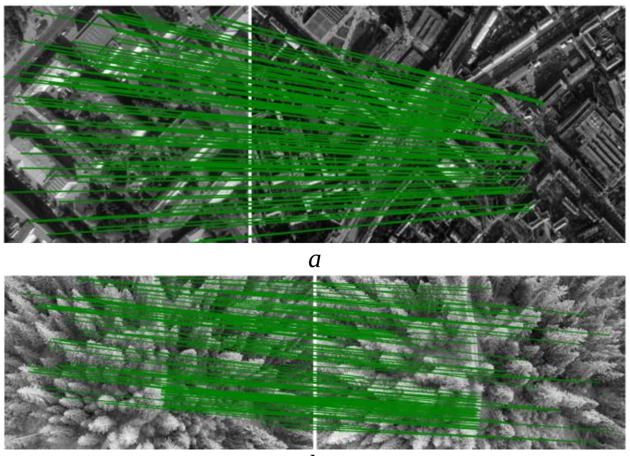

Figure 9: Usage of GlueStick for TRN per se (*a*) and over the same terrain with different angles (*b*)

Proposed approach

The suggested approach of this work is based on deep feature template matching. There are existing solutions of that, like robust template matching using scale-adaptive deep convolutional features [12] or *QATM* (quality-aware template matching for deep learning) [13]. In [12] scale-adaptive deep convolutional feature vectors are extracted from the template and the input image via the pre-trained VGG-Net – Figure 10. Each layer represents a different level of deep features of the actual image contents. Normalized cross-correlation (NCC) is used to measure the distance between features of the template and the image to detect the target image patch. Nonetheless, existing pretrained networks are not necessarily apt for extracting features that are relevant for comparing two images of the same terrain under different conditions.

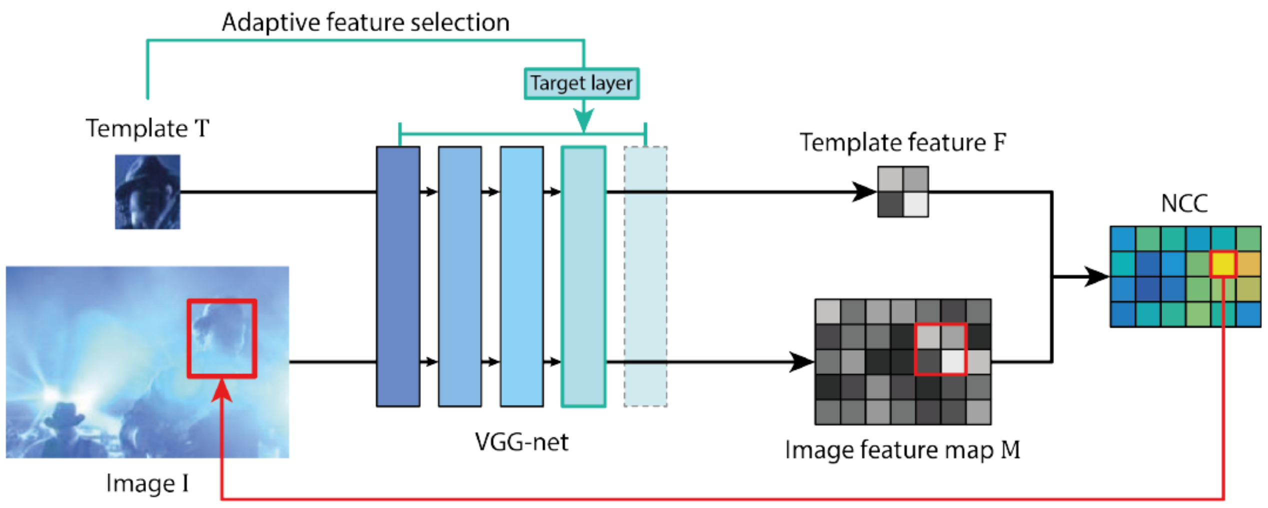

Figure 10: Scale-adaptive deep convolutional feature extraction based template matching [12]

Usually neural networks that are used as backbones for computer vision tasks are pre-trained on datasets like *ImageNet* [14]. It means that it can be used for finding a cat on a picture, using textures and high level features such as eyes, nose, ears and whiskers, but will struggle with finding a specific patch of terrain, because of the necessity to understand domain specific transformations caused by snow or vegetation coverage and discrepancy between seasons and lightning conditions. It's similar to the face recognition task, when we need to build deep feature space in which the same face has the same features [15]. Additionally it may allow moving along that space to perform a smooth transition from one face to another. For extracting such deep features this paper suggests using a triplet loss [15] – Figure 11.

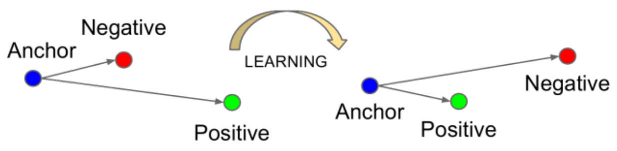

Figure 11: Triplet loss [15]

The main idea behind the triplet loss is to minimize the distance between an anchor (patch of particular area from source *A*) and a positive (patch of the same area from source *B*), both of which have the same identity (but may differ in terms season, angles of observation and so on) and at the same time to maximize the distance between the anchor and a negative of a different identity (random patch of a different area either from source *A* or *B*) – Figure 12.

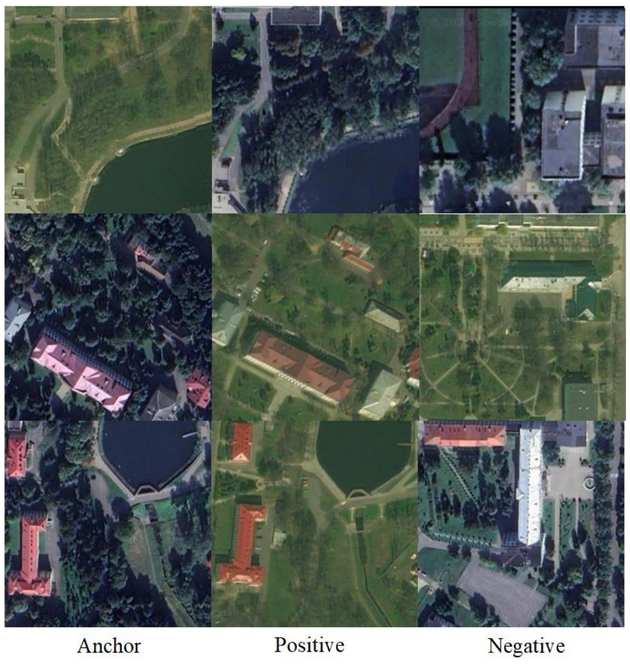

Figure 12: Examples of triplets for rotation insensitive training

Performing matching, the orientation of terrain fragments must be the same. Giving the possible inaccuracy of an onboard compass, the network is also train to be agnostic to discrepancy in orientation within the range of 20 degrees. It's achieved through augmentation in form of random rotation of positive image within that range during training (Figure 12 positive). For making the network agnostic to height uncertainty, images can also be randomly scaled. In order to get a rich feature space, images cropped from Google maps zoom level 13 to 20. The network's architecture is a modified version of ResNet50 – Figure 13.

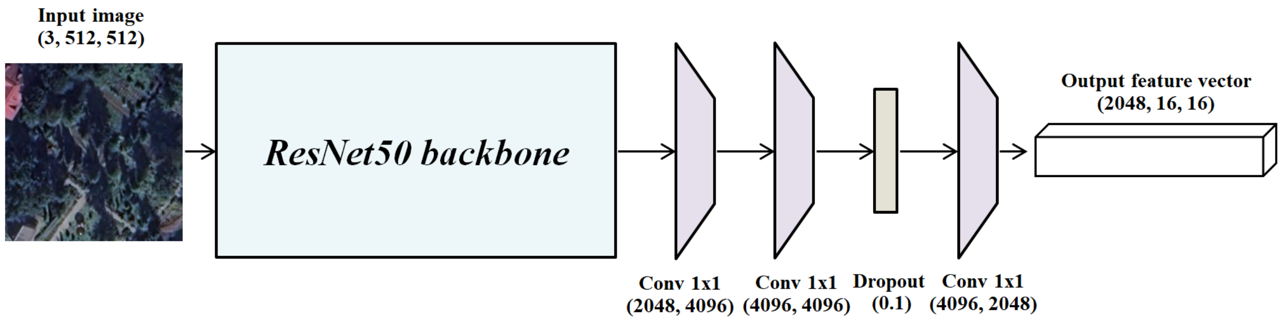

Figure 13: Feature extractor architecture

Triplet loss for the training process is described like this [15]:

$$L = \sum_i^N \left[\left\| f(x_i^a) - f\left(x_i^p\right) \right\|_2^2 \| f(x_i^a) - f(x_i^n) \|_2^2 + \alpha \right]_+ , \tag{1}$$

where *f(x)* – function representing feature extraction; x^a, x^p, x^n – anchor, positive and negative samples; *N* – batch size; *α* – bias.

Usage of *Weights & Biases* developer platform [16] proved to be effective for search of optimal hyperparameters such as: learning rate, batch size, dropout values, configurations of add-ons to the backbone and even pretrained backbone itself.

In case of not using an onboard compass for getting the rotation angle (may be connected with natural or artificially induced magnetic anomalies) the orientation must be determined only by comparing an image of observed terrain and the reference satellite map. For that a different approach to composing images for a triplet loss has been proposed. An image of the same area from a different source but with the same orientation is used as a positive sample, whereas an image of the same area, from the same source but rotated within 15 to 45 degrees is used as a negative one – Figure 14.

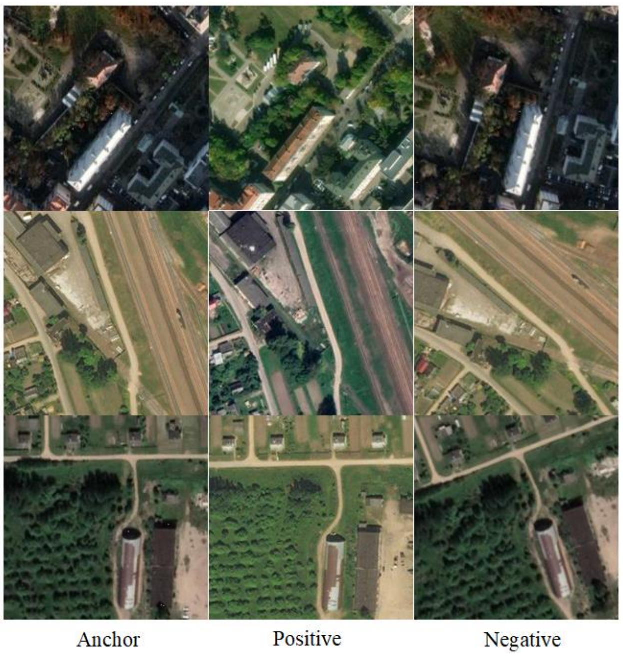

Figure 14: Examples of triplets for rotation sensitive training

A single multi-head network may be used for both purposes: navigation and rotation correction – Figure 15.

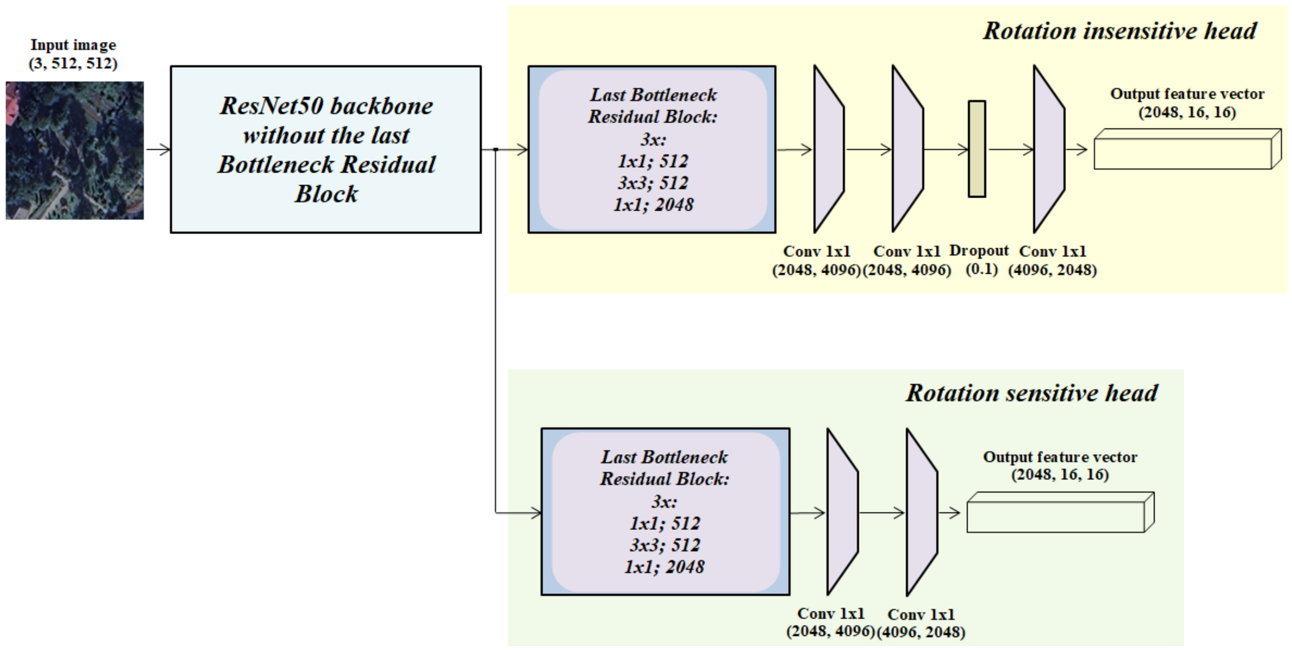

Figure 15: Two output head feature extractor

The training process in this case involves two stages: training of the rotation insensitive head together with the ResNet50 backbone and then freezing the backbone and training of the rotation sensitive head. Usage of a single network is expedient due to computational limitations of onboard hardware. Otherwise two separate networks (Figure 13) each for rotation sensitive and rotation insensitive feature extraction for navigation and angle evaluation respectively have proven to be better.

The actual process of navigation implies finding the location of a smaller image form camera on a rotated bigger image of the reference terrain view and shifting that reference view according to the flight in order not to move outside an area of search – Figure 16.

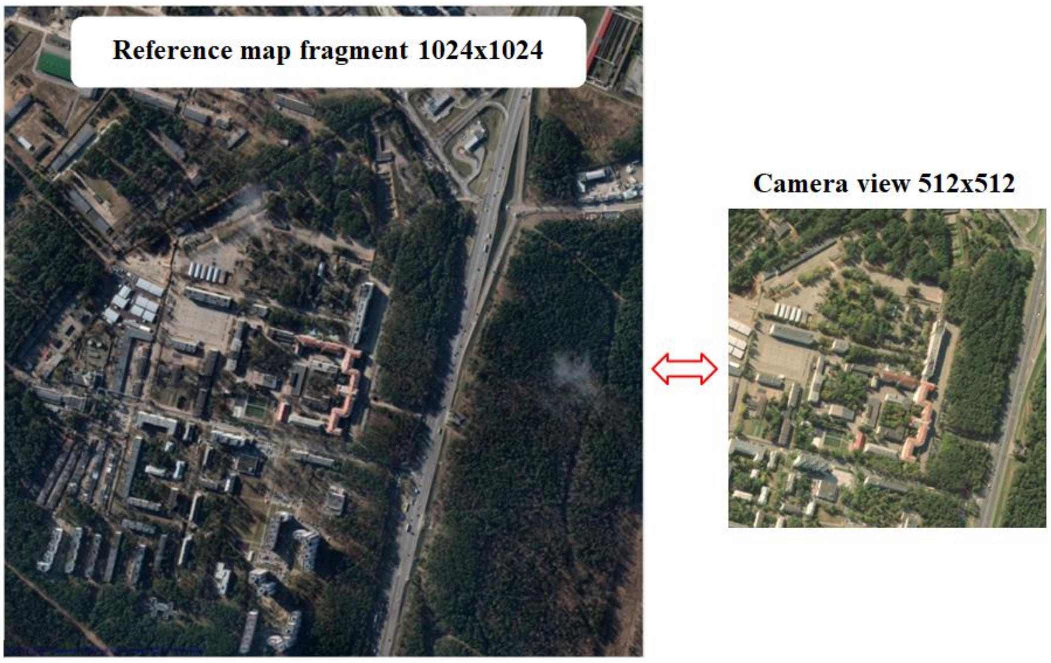

Figure 16: Task of locating camera image

The reference image is acquired using preliminarily loaded onboard base of tiles, with the scale of map provider and ancillary scaling that correspond to the current height of flying in order to ensure scale congruity between images.

Extracted deep features allow performing quality aware matching based on image semantics. Usage of a convolutional network allows working with images of any existing standard resolutions and respective reference images. For two images on Figure 16 we'll get two vectors of shape (2048, 32, 32) and (2048, 16, 16), so the task is boiled down to locate a patch of 16x16 on a patch of 32x32 – Figure 17.

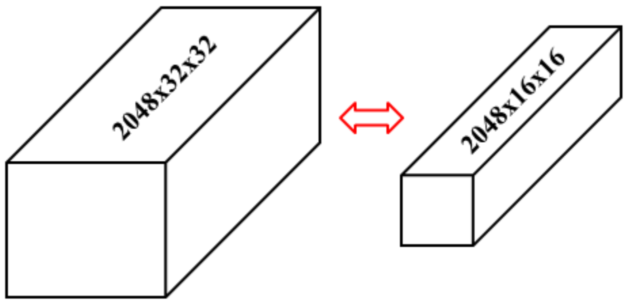

Figure 17: Task of locating feature vector

Matching is performed by sliding an image view feature vector over a bigger reference image feature vector within the plane of view (a window of 16x16 over one of 32x32) and calculating Euclidean distance between an image feature vector and a corresponding crop from the reference feature vector. A position with minimal distance will give the best matching.

The drawback of this approach is that we are not determining exact coordinates of our location. The coordinates of the reference image patch are known and we are locating relatively to it with the discretization of 32 pixels (image of 1024x1024 is convolved to 32x32). But that uncertainty is not accumulated over the flight. It's important to just properly shift the reference image according to UAV's movement.

Auxiliary correction of the orientation angle by means of image comparing is performed after the iteration of location determining. In this case a current camera view is compared with the set of rotated crops from the reference source turned within the range of -10 to +10 degrees – Figure 18.

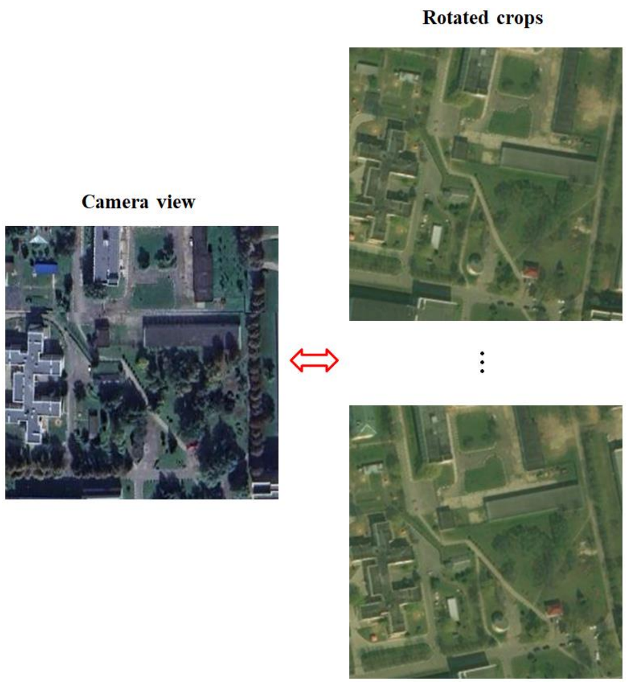

Figure 18: Task of correcting orientation

In a similar way, for that we extract deep features by a rotation sensitive head or a separate network and get the angle which corresponds to a crop image that provides a minimum distance in terms of feature space.

A complex of programs, connected via network, was created for experimental study of different facets of the suggested approach and proving their feasibility. A Unity 3d environment simulates flying over a terrain with different weather and lightning – Figure 19.

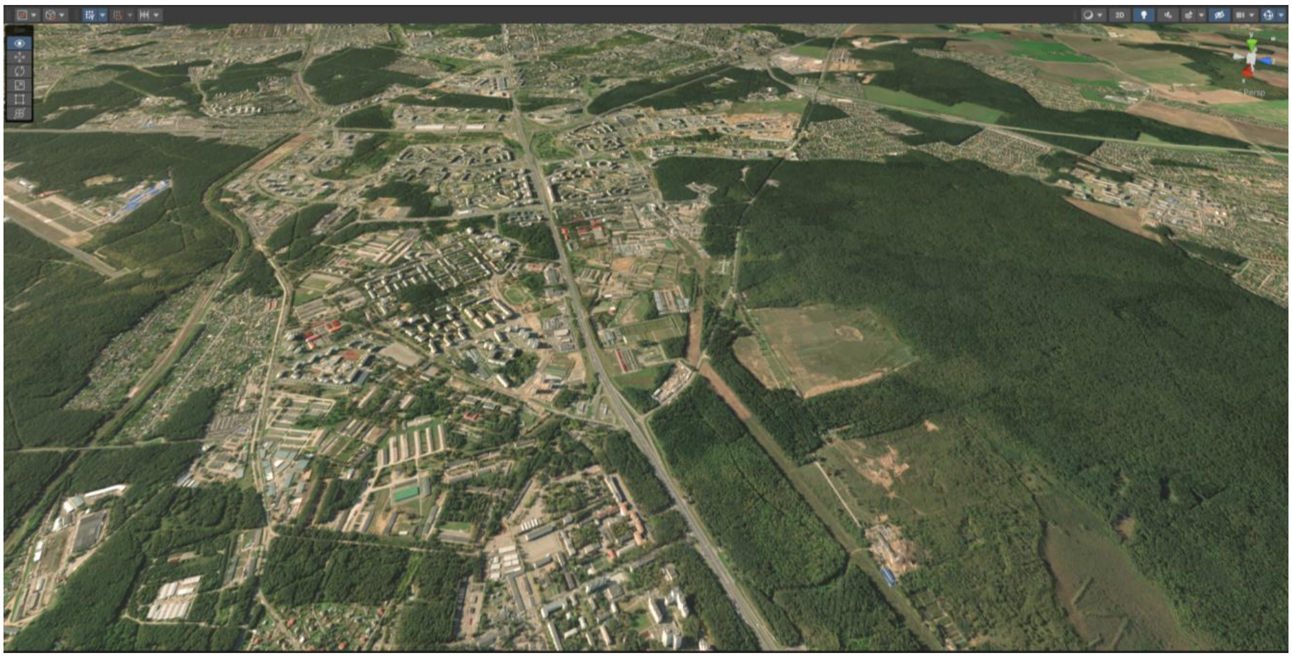
Figure 19: Simulation environment

Defining a desirable route to fly is performed by setting a set of waypoints onto a map. During the flight time a model UAV determines its location by video feed and analyzes its necessity to turn when it's in the vicinity of the next waypoint according to the route. That procedure is performed with a discretization that depends on UAV speed, height, view angle of a camera and reference image size in order not to fly outside the search area – Figure 20.

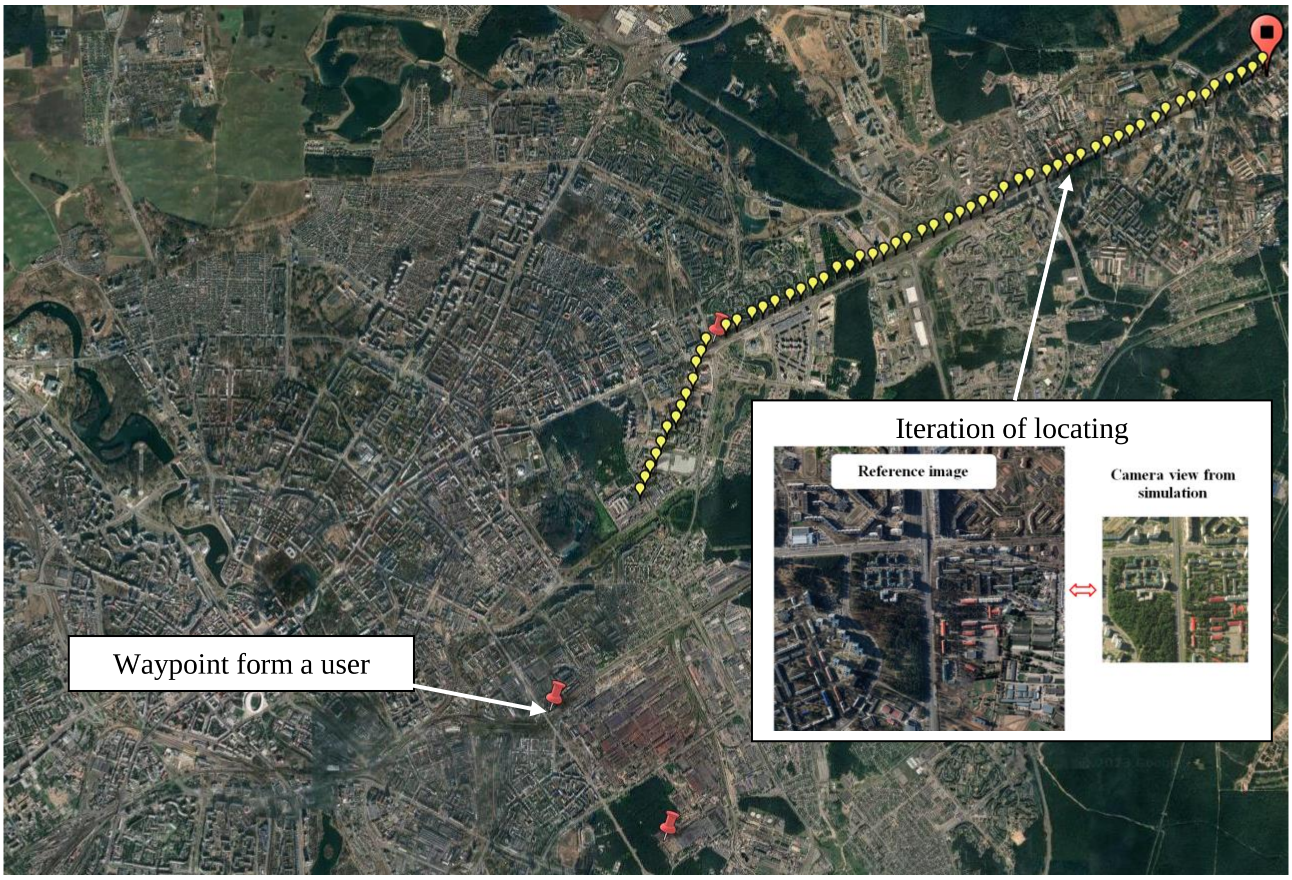

Figure 20: Process of flying

However, usage of template matching is struggling with flying over the deep forest, leading the determined location astray from a UAVs real position, moving forward towards a consecutive waypoint – Figure 21.

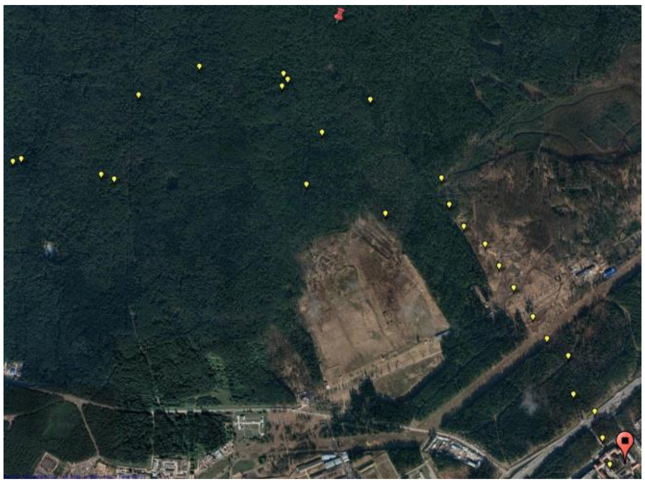
Figure 21: Flying over deep forest

Initially, flying over roads, patch of forest crossed with roads, border of forest and shrubs, it could determine its location. Moving deeper in the forest it had failed. Like for a human, hovering on a balloon over featureless forest or desert, navigation relying only on a sight from above would be almost intractable.

One way to overcome that is to pass a big fragment of the reference map through the network, extract deep features and apply an entropy filter [17] to that. Then set start and end points on a map and use one of the path finding algorithms (like A^* [18]) using only areas with high entropy level in terms of deep features, relevant for navigation – Figure 22.

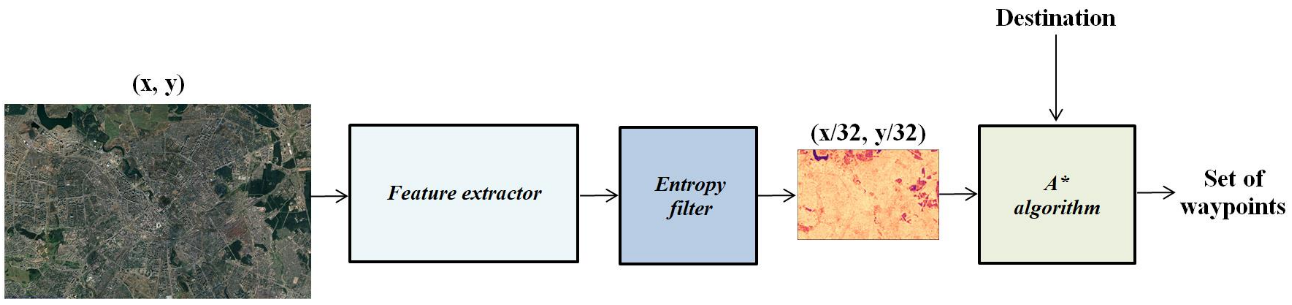

Figure 22: Path finding with respect to richness of navigation relevant deep features

Another way to address the problem of flight over terrain with poor features or with low altitude is to use VO as ancillary mechanism. There are several cases that illustrate the concrescence of the approaches. In case of struggling to perform template matching it doesn't provide enough confidence, expressed in terms of Euclidean distance for one particular template location over an average value. It's especially evident in cases of flying along the roads with low amount of distinctive features on the sidelines, because all the crops taking along the road will look relatively the same – Figure 23.

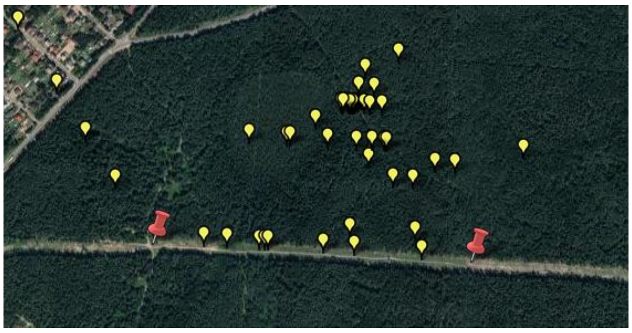
Figure 23: Flying over a featureless road

If such a situation with low maximum value over the average one exists, the reference map is shifted simply by a vector given by using VO between two consecutive images, like on Figure 9*b*. If we encounter an image that through template matching provides high enough confidence during N iterations – we align the reference map according to that. Number N is chosen in order not to move outside the search are due to possible VO errors. If we continue flying using VO (and shifting the reference map according to it) more than N iterations – we start analyzing Euclidean distance between consecutive terrain images. If distance exceeds a threshold – we perform a “looking around” procedure by soaring if necessary for getting a better vantage point and increasing the amount of features for performing template search over bigger space of the reference map in comparison with the case of regular flight. And here we use a gimmick of hypothesizing about possible location, taking K of most likely patches on a reference map (with respect to L past stored views from the camera), and comparing the next set of M consecutive images from camera with crops form the reference map – Figure 24.

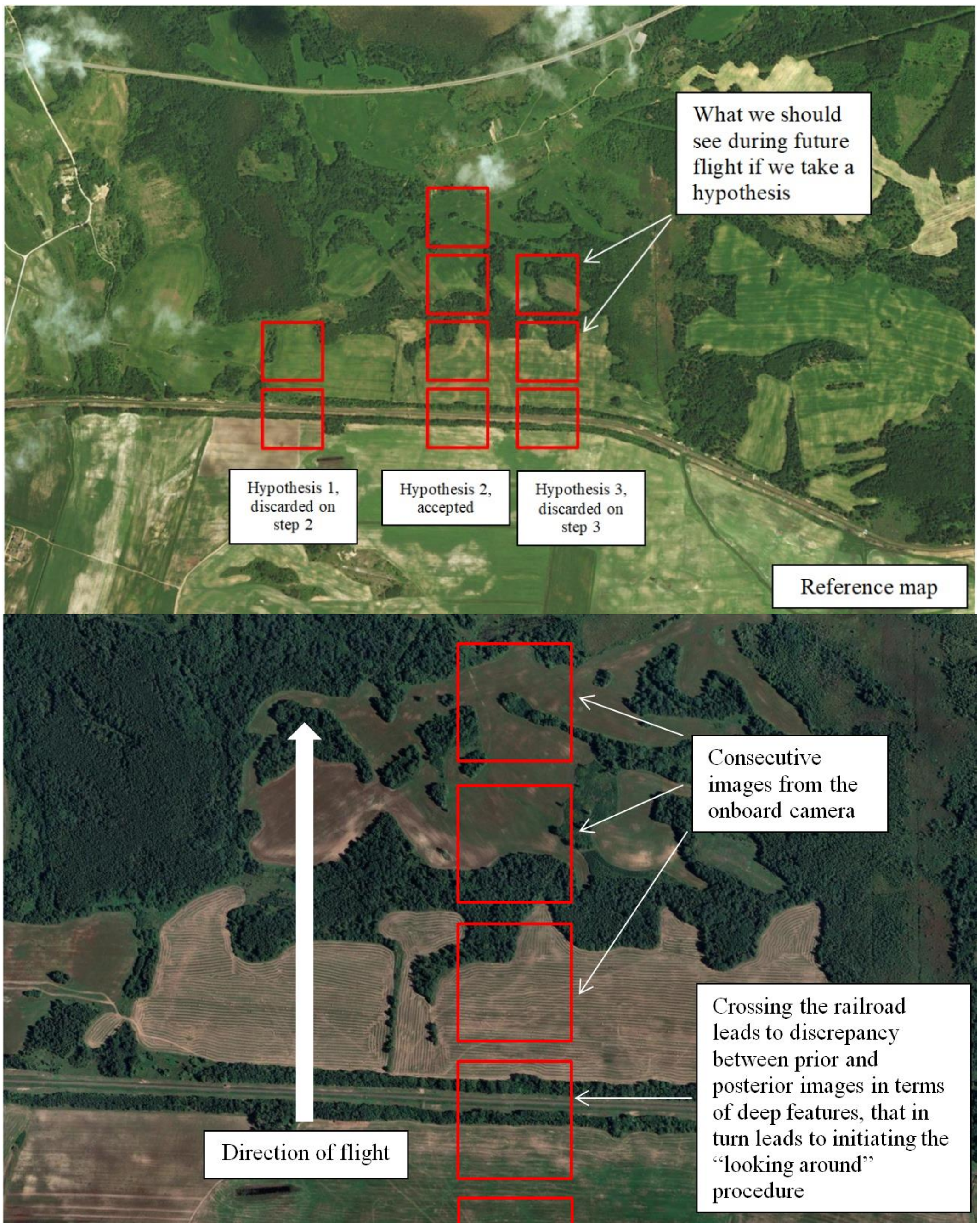
What we should see during future flight if we take a hypothesis
Hypothesis 1, discarded on step 2
Hypothesis 2, accepted
Hypothesis 3, discarded on step 3
Reference map
Consecutive images from the onboard camera
Direction of flight
Crossing the railroad leads to discrepancy between prior and posterior images in terms of deep features, that in turn leads to initiating the “looking around” procedure

Figure 24: Hypothesizing over potential locations

If an image from the camera doesn't correspond to a hypothesized future image, the hypnosis is discarded. In this case navigation process relies on accumulation, unlike single shot template matching used in a normal mode. The number of tracked hypotheses depends on computational capacity of an onboard system. During hypothesis checking, if a new terrain image provides a better matching score than currently tracked locations, the least confident hypothesis is discarded and a new one is started to be tracked.

In case when finding relevant features for navigation is impossible (the aforementioned process of hypothesizing didn't provide results with enough confidence during *J* iterations), like when we moved from the forest, without elements of a distinctive border line, to the field, we have to address the entropy reference map (Figure 22). If the next waypoint is closer than the distance *D*, and it has enough features for navigation in its vicinity we continue moving. Else we must find on a reference map a closest big enough area with rich features and add an auxiliary waypoint in the center of it just for the sake of navigation. Being there, we determine the location and continue moving towards the desirable destination. In the real life it corresponds to the case when moving from the forest to the field we know that there is a nearby settlement to the west and we move there.

The advantage of template matching approach is the ability to learn different type of terrain transformation, utilizing existing open datasets, like pre and post disaster imagery [19] which is especially relevant for military UAV's – Figure 25.

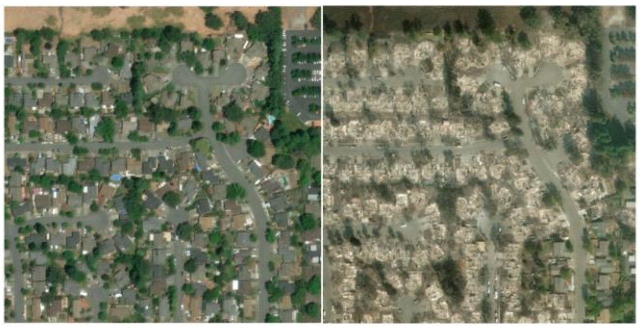
Figure 25: Pre and post disaster dataset [19]

The important point for onboard systems is computational requirements of any suggested approach. Template matching doesn't require frequent iterations of locating. It's only necessary that a UAV won't fly outside the area of search on a patch of the reference map. In a simulated environment, using NVIDIA GeForce RTX 2080 SUPER (compute capability 7.5), flying altitude 500 meters, onboard camera resolution of 512x512 and a patch of the reference map for searching of 1024x1024, a single iteration of navigation takes about 0.5 sec and provides speed of a UAV up to 75 meters per second, because otherwise a UAV leaves the search area of a reference map. It means that onboard CUDA-enabled products such as Jetson Nano (compute capability 5.3), Jetson TX2 (compute capability 6.2), Jetson Orin Nano (compute capability 8.7) or even Raspberry Pi 6 can perform such a task with sufficient for UAVs speed. In terms of RAM consumption 4 GB is enough only for navigation, but for when a UAV also performs object detection during flight, using a costumed trained YOLOv5, it requires 8 GB.

The entire territory of the Republic of Belarus and its adjacent area with a scale sufficient for navigation from high altitude (about 1 km) requires approximately 500 GB of storage which is apt for a modern M.2 SSD situated onboard. More detailed map of Minsk city with its outskirts requires 20 GB.

Conclusions, alternative solutions and future work

This work relies on the concrescence of strong points from two approaches. Deep feature template matching provides semantic understanding, agnostic to domain specific transformation caused by discrepancy in angles of observation, time of year, weather, vegetation and lighting conditions. Visual odometry allows keeping track of observed features on land and performing shifting of the reference map in order to stick to that again when the observed terrain has enough features.

Within the framework of conducted research it's necessary to mention solutions that were tested but haven't worked well.

In simulation navigation can also be performed by usage of the same point (line) based matcher (like *GlueStick*), that has been used for VO. Location and the shifting vector for the reference map are determined based on trigonometry using camera's obliquity – Figure 26.

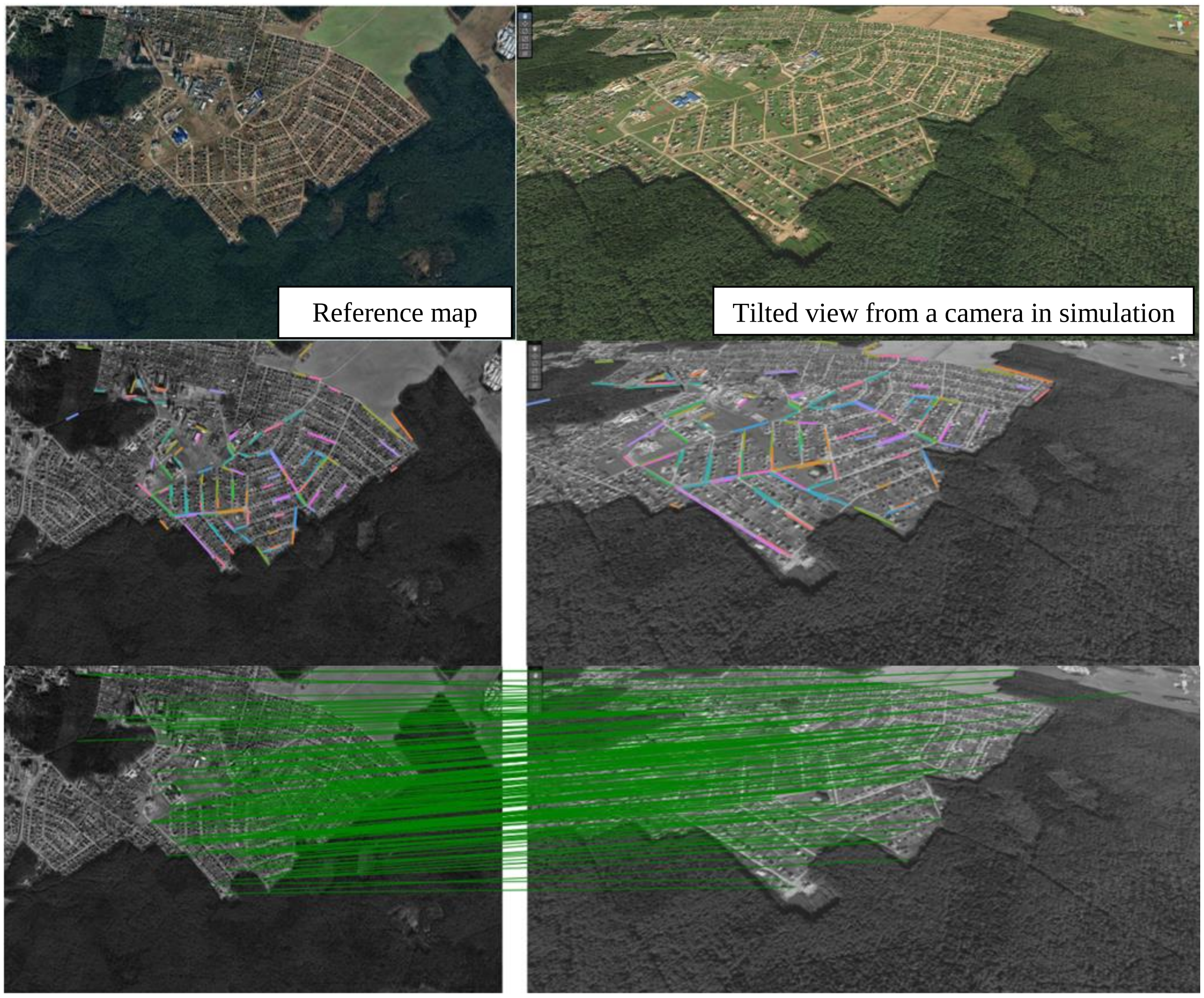

Figure 26: Usage of *GlueStick* in simulation

But, as that's been mentioned, it demonstrated poor results with real camera feed due to inability to handle the situation with discrepancy in angles of observation.

An approach with predicting a bounding box that frames a camera view onto the reference map in a YOLO-like manner was also tried but was proved to be not efficient in terms of precision.

Several solutions represent relevance to the current work in terms of possible future usage. Unsupervised learning of visual features by contrasting cluster assignments [20] may be used for pre-training a backbone on a set of terrain images of different scales.

On a low altitude, flying through a city, a point cloud can be built from a depth map that can be predicted even from a monocular camera by open solutions [21]. It then allows performing counter matching with segmented street view. An example of visual based SLAM in city is described in [22] – Figure 27.

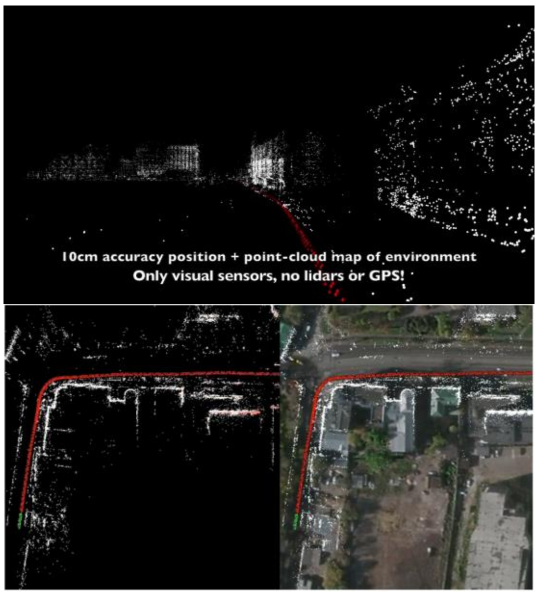
10cm accuracy position + point-cloud map of environment
Only visual sensors, no lidars or GPS!

Figure 27: Visual-based SLAM in the city [22]

A good solution in terms of not going astray during the navigation process is to use existing open source projects for lane (road) detection [23] to stick to a particular road and recognize road junctions during the flight, matching them with the reference map – Figure 28.

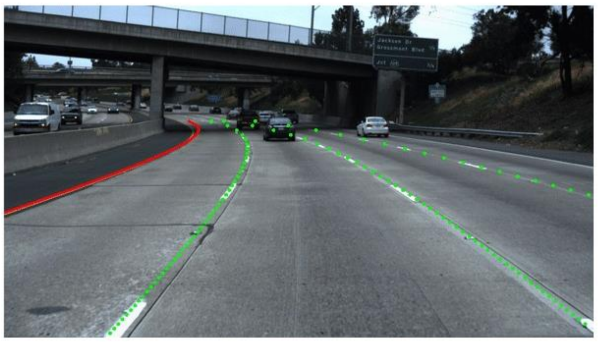

Figure 28: Line detection process [23]

In addition to it, the suggested in this paper approach can also be further modified. Calculation of a shifting vector based on VO, in case when we don't have enough features to perform template matching, can be performed with respect to some terrain features inferred by a neural network, in order to improve precision. Points, tracked from tree tops over the forest due to their closeness to a UAV will give a different shifting magnitude in comparison with points from a filed when a UAV flies the same distance. In this case must we collect real data with known positioning and train a neural network based on the value of odometry distance and actual distance that will correct the magnitude of a shifting vector based on pictures of terrain it observes. For improving VO a UAV can also have some onboard inertial navigation system and a network will perform sensor fusion in order to determine a resultant shifting vector.

For the “looking around” procedure the search for patches of the reference map that match a camera view (for hypothesizing over potential locations) is performed just by grid search over a potential area of the reference map. A more efficient way is to extract deep features in advance and organize a hash table for performing a quick lookup based on similarity, using solutions like fuzzy hashing [24].

A more intuitive way of navigating for a human being would be to look around with oblique view (like on Figure 26) and consider all the observable features, not only the view right from above perpendicular to the surface. It requires for a neural network to understand the transformation between a 2d satellite map and a 3d view of a terrain given under certain angle and form certain height. For such a task it can be trained using existing 3d terrains in solutions like Cesium [25] – Figure 29.

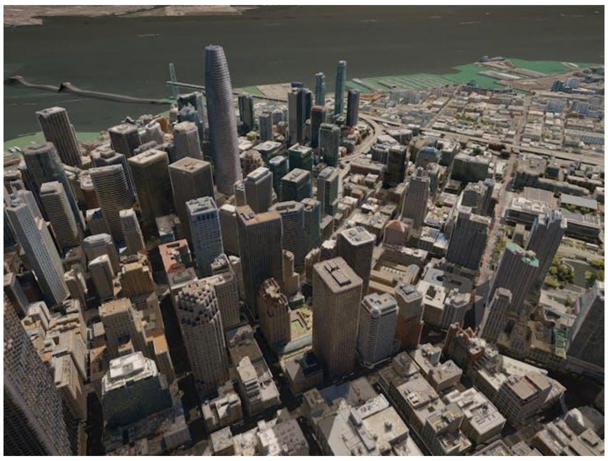
Figure 29: Cesium platform for Unity 3d [25]

Combination of looking around with different obliquity of a camera, looking perpendicular to the surface, accumulating features and performing maneuvers in order to alleviate uncertainty in understanding the position is the area of further research.

REFERENCES

1. **Image Transformation Learning for Drone Navigation without GPS**: [Electronic resource] // HuCE – cpvrLab. URL: https://www.youtube.com/watch?v=5JEFe2_L4So. (Date of access: 23/10/2024).

2. **Anthony T. Fragoso et all.** A seasonally invariant deep transform for visual terrain-relative navigation. Science robotics. 2021. Vol. 6, № 55.

3. **Autonomous Navigation with Improved Visual Terrain Recognition**: [Electronic resource] // Caltech. URL: https://www.youtube.com/watch?v=U5Kr0YI3sec. (Date of access: 23/10/2024).

4. **Jiaming Sun et al.** LoFTR: Detector-Free Local Feature Matching with Transformers. 2021. arXiv: 2104.00680.

5. **ORB (Oriented FAST and Rotated BRIEF)**: [Electronic resource] // OpenCV. URL: https://docs.opencv.org/4.x/d1/d89/tutorial_py_orb.html. (Date of access: 23/10/2024).

6. **ORB vs ML-aided Visual TRN**: [Electronic resource] // KEF Robotics. URL: https://www.youtube.com/@kefrobotics6924. (Date of access: 12/01/2024).

7. **Jiexiong Tang et al.** Neural Outlier Rejection for Self-Supervised Keypoint Learning. 2019. arXiv: 1912.10615.

8. **Paul-Edouard Sarlin et al.** SuperGlue: Learning Feature Matching with Graph Neural Networks. 2020. arXiv: 1911.11763.

9. **Yifan Wang et al.** Efficient LoFTR: Semi-Dense Local Feature Matching with Sparse-Like Speed. 2024. arXiv: 2403.04765.

10. **Philipp Lindenberger et al.** LightGlue: Local Feature Matching at Light Speed. 2023. arXiv: 2306.13643.

11. **Rémi Pautrat et al.** GlueStick: Robust Image Matching by Sticking Points and Lines Together. 2023. arXiv: 2304.02008.

12. **Jonghee Kim et all.** Robust template matching using scale-adaptive deep convolutional features. APSIPA Annual Summit and Conference. 2017.

13. **Jiaxin Cheng et al.** QATM: Quality-Aware Template Matching For Deep Learning. 2019. arXiv: 1903.07254.

14. **ImageNet**: [Electronic resource] // URL: https://www.image-net.org. (Date of access: 23/10/2024).

15. **Schroff et al.** FaceNet: A unified embedding for face recognition and clustering. 2015. arXiv: 1503.03832.

16. **Weights & Biases**: [Electronic resource] // URL: https://wandb.ai/site. (Date of access: 23/10/2024).

17. **Examples. Filtering and restoration. Entropy**: [Electronic resource] // Scikit-image. https://scikit-image.org/docs/stable/auto_examples/filters/plot_entropy.html. (Date of access: 23/10/2024).

18. **A* Search Algorithm**: [Electronic resource] // https://www.geeksforgeeks.org/a-search-algorithm. (Date of access: 23/10/2024).

19. **Annotated high-resolution satellite imagery for building damage assessment**: [Electronic resource] // xBD Dataset. URL: https://xview2.org/dataset. (Date of access: 23/10/2024).

20. **Mathilde Caron et al.** Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. 2021. arXiv: 2006.09882.

21. **Lihe Yang et al.** Depth Anything V2. 2024. arXiv: 2406.09414.

22. **Dioram Visual Navigation(based on Dioram SLAM One) point-cloud mapping in a city scale**: [Electronic resource] // Dioram: Computer Vision, Machine Learning, SLAM. URL: https://www.youtube.com/watch?v=DwAT46MdyXk. (Date of access: 23/10/2024).

23. **Awesome-lane-detection**: [Electronic resource] // Github.com. URL: https://github.com/amusi/awesome-lane-detection?tab=readme-ov-file#2023. (Date of access: 23/10/2024).

24. **J. Oliver, J. Hagen.** Designing the Elements of a Fuzzy Hashing Scheme. Science robotics. 2021 IEEE 19th International Conference on Embedded and Ubiquitous Computing (EUC), Shenyang, China, 2021, pp. 1-6.

25. **Real-World 3D Geospatial Capability for Unity**: [Electronic resource] // Cesium.com. URL: https://cesium.com/platform/cesium-for-unity. (Date of access: 23/10/2024).

РУЛЬКО Е.В.

НАВИГАЦИЯ ПО СНИМКАМ МЕСТНОСТИ НА ОСНОВЕ СОПОСТАВЛЕНИЯ ГЛУБОКИХ ПРИЗНАКОВ И ВИЗУАЛЬНОЙ ОДОМЕТРИИ

Военная академия Республики Беларусь

Основной проблемой для систем навигации по снимкам местности является несоответствие визуальных признаков между фрагментом опорной картой и изображением с борта БПЛА. Снимки могут быть сделаны в различное время года, в различную погоду, с различными растительным покровом, условиями освещения и под различными углами обзора относительно плоскости земной поверхности. Данная работа предлагает использование сопоставления глубоких признаков, извлеченных в рамках неконтролируемого обучения с использованием триплет-ошибки. Это обеспечивает понимание семантики изображений, не зависящей от трансформаций местности. В рамах полёта над местностью с недостаточным количеством визуальных признаков для навигации (лес, поле), в работе предложено дополнительное использование визуальной одометрии с процедурой привязывания к опорной карте после получения достаточного количества признаков, с построением гипотез относительно местоположения. Извлечение глубоких признаков натренированной сетью из опорной карты и применение к ним фильтра энтропии позволяет планировать маршруты полёта над местностью, обладающей достаточным разнообразием признаков, необходимых для навигации.

Ключевые слова: *навигация по снимкам местности, глубокие признаки, машинное обучение, визуальная одометрия, БПЛА.*

Рулько Евгений Викторович, кандидат технических наук, доцент. Начальник научно-исследовательской лаборатории моделирования военных действий учреждения образования «Военная академия Республики Беларусь». Сфера научных интересов: глубокое обучение, машинное зрение, обучение с подкреплением, нейронауки, активный вывод, принцип свободной энергии, рефлексивное управление.

Eugene Rulko, PhD, associate professor in computer science. The head of the research laboratory of military operation simulation of the educational institution «Military academy of the Republic of Belarus». Research interests: deep learning, computer vision, reinforcement learning, neuroscience, active inference, free energy principle, reflexive control.

E-mail: eugeni1533@gmail.com