Development of the Detecting System of the Landmark Tags to Increase the Navigation Accuracy of an Unmanned Vehicle in a Known Location

Pavel Belyaev, Anton Spivak, Evgenii Neverov
ITMO University
Saint Petersburg, Russia
monoe1337@gmail.com, anton.spivak@itmo.ru, datnever@ya.ru

***Abstract*—This paper proposes the landmark detection system for use in unmanned vehicle based on NVIDIA Jetson embedded device. The paper describes the synthetic dataset generation and training YOLOv4 neural network using data augmentation and transfer learning techniques to detect the landmark in incomplete data conditions. The real-time landmark detection module was implemented on the NVIDIA Jetson Xavier. The final mean average precision was resulted as 83.8%.**

I. Introduction

During autonomous movement of unmanned vehicles (UV), be it cars, robots, drones, etc. navigation plays a key role. Information about the exact location of the UV allows to significantly expand the possibility of using autonomous UV, which will raise the tasks of autonomous UV to a new level.

To solve the positioning problem, satellite navigation systems (SNS) such as GLONASS and GPS [1] are mainly used. The advantage of SNS is sufficient outdoor positioning accuracy and global coverage due to the large number of satellites. The disadvantages of SNS include the inability to determine the location in enclosed spaces, as well as a great dependence on weather conditions. Also, the disadvantages include insufficient accuracy in the tasks of positioning the UV.

These shortcomings can be corrected by using additional methods of UV navigation, namely the use of computer vision methods. The use of methods for detecting landmarks for positioning was proposed due to the use of the UV in winter conditions. This method serves to support UV navigation in case of failure of the main navigation systems.

This paper deals with the use of artificial intelligence tools, namely deep learning and the method of computer vision for detecting landmarks during the movement of the UV in a harsh climate. The need to use additional methods of orientation is due to the problems of the navigation systems described earlier. The most commonly used method is localization by means of a GPS module. However, the need to operate an unmanned vehicle (UV) in changing conditions (including premises) imposes restrictions on the use of global positioning systems. For example, in [2] navigation in an enclosed space based on various lidar configurations is considered. The approach in [3] describes the model of the UV functioning on public roads. The cost function is calculated based on the distance of the UV from the reference path, that is, the proximity to the edge of the carriageway. Nevertheless, the departure of the UV, in which the lidar is used as the main localization sensor, outside the premises, provided that it functions in an environment with almost complete absence of obstacles, as well as without a clearly distinguishable edge of the carriageway, leads to the impossibility of correctly obtaining information about its location. In [4] authors proposed the traffic sign recognition model and deployed it on a Raspberry Pi. The testing was performed in the TASS PreScan simulation environment which is an urban environment including cars, pedestrians, and road signs. The accuracy was 99.8%. Nevertheless, the lack of suitable datasets in the tasks other than urban environment leads to the necessity of developing own simulation, followed by the generation of dataset in conditions close to the expected conditions of unmanned vehicle use.

Based on the literature review, the following tasks were concluded: 1. Navigation should be carried out autonomously 2. Due to UV functioning in an outdoor environment with few features for retrieval, lidar-based systems are not suitable. 3. the use of a computer vision based system is possible, provided that orienteering marks are added to the terrain. Orientation marks must be recognized under different weather conditions (rain, snow) Therefore, it was decided to use a convolutional neural network approach to create a model for recognizing a landmark mark as an object and then reading the encoded landmark as it is coming up This work proposes the development of a system for detecting landmarks in a snowy area using YOLOv4 algorithms with subsequent deployment on NVIDIA Jetson Xavier.

II. Dataset and Object Detection Methods

A. Data collection

The presence of unique data for research is one of the main parts in computer vision. The data can be used for object detection, segmentation, classification. Despite the fact that there is a large amount of data in the public domain, there was a problem of the lack of the required dataset, since the tag model was developed for detection in winter conditions. To solve this problem, NVIDIA SDK tools were used to generate a synthetic dataset. NVIDIA has developed Isaac software for the development of robotic systems and work with artificial

intelligence. This technology allows robotic systems to be deployed with subsequent GPU computation, which increases the speed of computing vision models. Detection methods based on convolutional neural networks require a large number of labeled samples to train parameters. But in general, getting a lot of labeled real data can be tricky [5]. To make up for the lack of training images, NVIDIA proposed a simple method for creating images, which is presented below.

The NVIDIA SDK includes the Isaac Engine (Application Platform), Isaac GEMS (High Performance Robotics Algorithms Packages), Isaac Apps (Reference Applications), and NVIDIA Isaac Sim (Simulation Platform) [6]. Generating data Fig.1 in this way allows to collect a sufficient set of data in the absence of the possibility of collecting real data. Through NVIDIA Isaac Sim, it is possible to fine-tune data generation, namely, setting up patterns of object location, illumination, object overlap with other objects. All of these settings allow to generate unique data to train models. Finally, affine transformation methods were used for the data to increase its size by increasing, rotating, cropping the original data. In total, the final dataset contained 3126 labeled images example of which is presented on Fig.2.

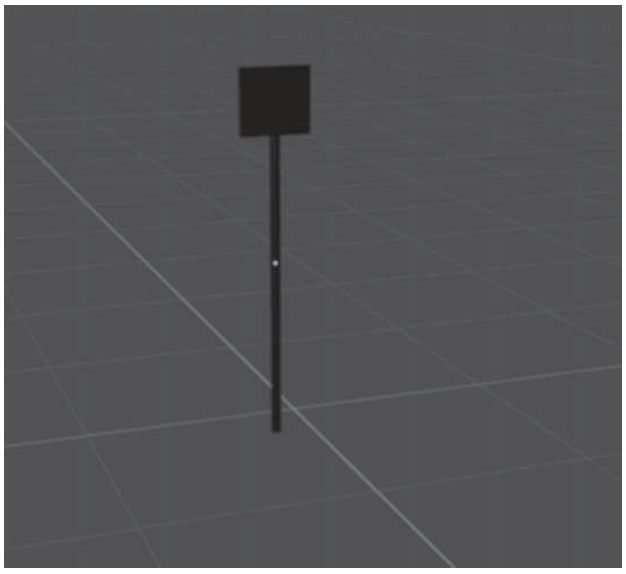

Fig. 1. Sample of the Blender tag model used for dataset generation

B. Transfer Learning

For the experiment, the transfer learning method was used for training on an synthetic dataset. For this, a pre-trained weights were applied, which had been trained using the MS COCO data in the original weight file, to pick patterns from the initial set of objects in the MS COCO set [7]. This is necessary to detect tags in a winter environment.

Transfer learning is a method for training neural networks, when the templates of a neural network that has been trained on one task are transferred to another task. This approach to solving the problem allows you to use the accumulated patterns from other datasets, which leads to optimization of

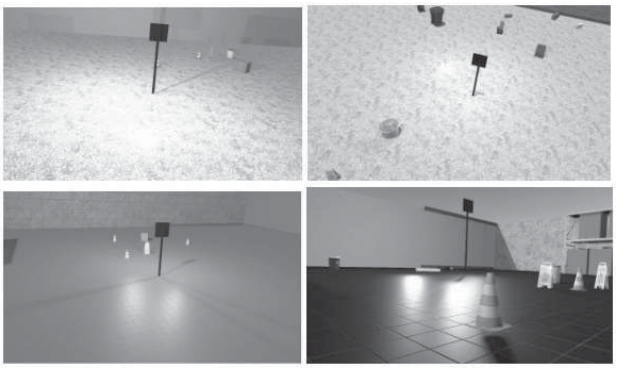

Fig. 2. An example of a generated dataset in Unity3D using the Isaac SDK

training on a smaller dataset. Considering the fact that the data can be perfect (artificial / generated data that may not be widespread or complex), this training method allows to use patterns from real data to improve the quality of training. One of the main advantages of this training method is the speed of training the model with increased accuracy and less error, which allows to create more complex models in less time, unlike training the model from scratch. Neural networks that are used for classification usually contain N output neurons in the last layer, where N is the number of classes. In the problem of landmarks recognition, the number of classes differs from the number of classes in the original dataset. In this case it was necessary to completely discard this last layer and put a new one with the required number of output neurons.

C. YOLOv4

The choice of YOLOv4 architecture is based on the fact that YOLOv4 indicators are located on the Pareto optimality curve (Fig. 3) and outperform the fastest and most accurate detectors in both speed and accuracy, which improves performance in real time, based on [5]. Generally, a convolutional neural network is the most representative deep learning model. It is widely used for ultra-high resolution image reconstruction, image classification, face recognition, object detection, prediction and video analysis. In addition to YOLO [8], network architectures such as AlexNet, VGG [9], SSD [10], and R-CNN [11] are used to improve performance and accuracy. The choice of the YOLOv4 architecture is due to the fact that, unlike the YOLOv3 implementation, YOLOv4 is more appropriate for object detection in real-time. Some YOLO releases, such as tinyYOLO, allow models to be deployed to the embedded devices [12] with minimal loss of precision. Nevertheless, the advantage of using NVIDIA Jetson as an embedded device is the possibility to involve CUDA cores and thereby significantly speed up the inference of the model, i.e. to increase the frame rate. Unlike other CNNs that operate on a two-step basis, YOLO is able to predict bounding boxes and class probabilities in one step. Algorithms Fast R-CNN, Faster R-CNN are based on the region proposals approach, that is, region assumptions. They consist of two parts, the first part builds sets of regions that, with a certain probability, display

an object other than the background. The second part deals with the processing of these assumptions and, based on the processing results, classifies the object at each frame.

To balance speed and accuracy, the YOLOv4 networking architecture uses Darknet-53 inter-stage partial connectivity (CSP) framework. Based on [8], CSP Darknet-53 has the best performance on the COCO dataset. To increase the receptive field that affects the network unit, YOLOv4 integrates a spatial pyramid pooling unit (SPP) and a modified path aggregation network (PANet). As for the detection head, the YOLOv4 head is assembled in a new model designed to predict objects at multiple scales. There is only one class of objects in this study. Therefore, the number of filters = (class + 5) × 3 = 18.

Also in YOLOv4, new improvements were implemented, including Weighted-Residual-Connections (WRC), Cross mini-Batch Normalization (CmBN), Mish Activation, Complete Interaction over Union (CIoU) loss, Mosaic data augmentation, and DropBlock regularization.

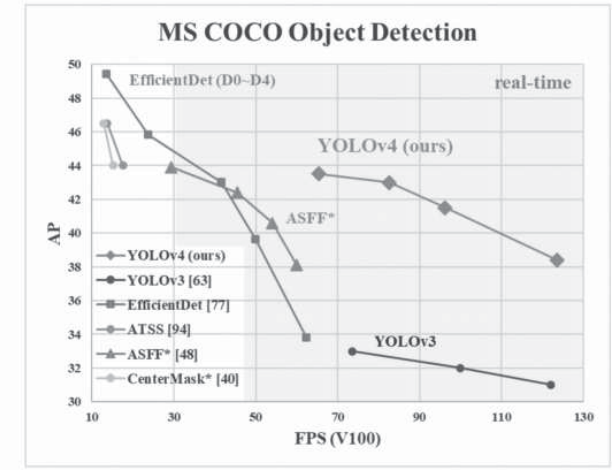

Fig. 3. Speed and performance of different models [8]

D. Research results

At this stage of the research, testing will take place in Unity3D using the Isaac SDK development tools. Isaac Sim allows to use Unity3D as the simulation environment for Isaac robotics. Isaac Sim Unity3D provides an expandable test environment to evaluate the performance of the Isaac navigation stack. It also provides an infinite stream of procedurally generated, fully annotated training data for machine learning. Features include emulation of sensor hardware and robot base models, scene randomization, and scenario management.

E. Equipment setup

Target platform is NVIDIA Jetson AGX Xavier. Xavier module makes AI-powered autonomous machines possible, running in as little as 10W and delivering up to 32 TOPs. As part of the world's leading AI computing platform, it benefits from NVIDIA's rich set of AI tools and workflows, which enable developers to train and deploy neural networks quickly. Jetson AGX Xavier is supported by the NVIDIA JetPack SDK, which can help save big by reducing development effort and expense. has an integrated Volta GPU with 512 CUDA cores that supports floating point and half floating point calculations. An Intel Realsense D415 camera was used to read the image. The D415 has a standard field of view and uses rolling shutter sensors. This field of view results in higher depth resolution for smaller objects or when more precise measurements are required. The rolling shutter sensors and smaller lenses allow for a lower cost, yet highly capable depth camera. The assembled robot model for further research stages is shown in Fig.4.

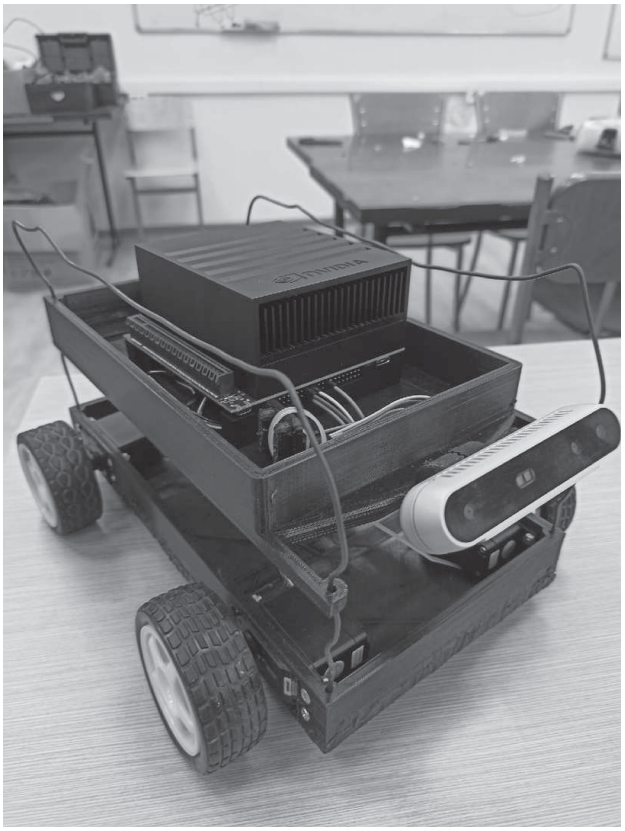

Fig. 4. Developed testing robot model for subsequent experiments

III. Performance and Accuracy Ratings

To assess the performance and accuracy of algorithms, there are a number of estimates, which can be calculated for a specific set of data, you can compare the performance of several algorithms. In our case, the problem arises of choosing the necessary measure due to the fact that it is necessary to evaluate both the quality of detection of objects from the list of classes we have defined, and the correct classification of this object.

For the task of detecting objects, such a measure as the average accuracy (AP) is usually used, obtained from two indicators: precision and recall [13].

Possible events when an object of one specific class is detected: True positives - an object of the required class was

correctly enclosed in a framing window - found. False positives - something else was enclosed in the framing window, but not an object of the desired class.

A. Precision and recall

Precision denotes the percentage of true positives among all results for this class.

$$Precision = \frac{True\ positives}{True\ positives + False\ positives} \quad (1)$$

Recall refers to the percentage of framing windows found for a given class, among those represented in the ground truth of this image.

$$Recall = \frac{True\ positives}{Number\ of\ ground\ truth\ boxes} \quad (2)$$

B. IoU

In order to be able to assess the correctness of enclosing an object in a framing window, the IoU metric is used. A typical threshold value indicating correct detection is IoU more then 50% Fig.5.

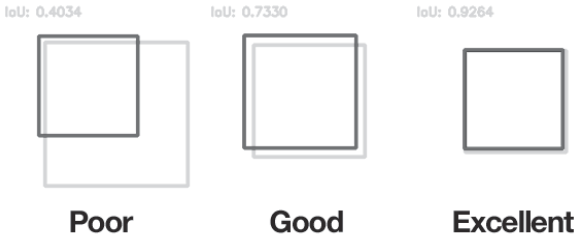

Fig. 5. Examples of Intersection over Union (IoU) [14]

Intersection over Union is a metric used to assess the accuracy of algorithms that detect objects. The IoU value is equal to the quotient of the area of intersection of the predicted framing window and the response window from the ground truth by the area of the union of these windows Fig.6.

C. Mean Average Precision

Since we are evaluating the accuracy for the detection and classification algorithm, we cannot limit ourselves to the average accuracy, so in the future we will operate with the concept of Mean Average Precision. Measure Mean Average Precision (mAP) - average AP.

$$Mean\ Average\ Precision = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \quad (3)$$

where Q - number of requests.

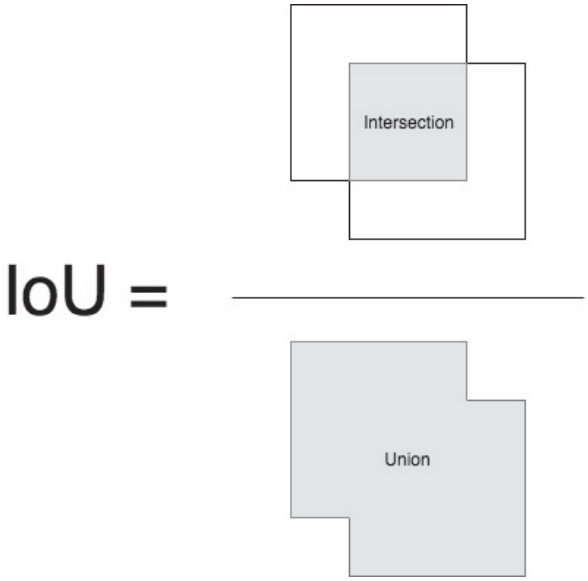

Fig. 6. Finding the Intersection over Union (IoU) [14]

IV. Results

After the process of labeling the data, the stage of training was performed. Training data is presented as the form of the ratio of the image with the txt file of the label. The label file consists of the class number and point coordinates in the *x, y, x + h, y + h* format, where *h* is the width and height of the stroke. In this experiment, training took place in one class, and therefore the cfg parameters were selected. As mentioned earlier, the transfer of training the model took place on the pre-trained MC COCO scales, since it contained the basic patterns for the label. The original data was generated synthetically and was close to ideal, which could lead to poor results, so the transfer learning was performed. The model was trained on a working machine using an NVIDIA GTX 1060 6GB graphics card. The training was performed on the operating system Ubuntu 18.04, with the version of the NVIDIA driver 450.102.

TABLE I. Comparison of mAP measure and average IoU for a dataset with affine transformations and without affine transformations in simulation

	Model without affines, %	Model without affines, %
mAP	25.8	83.8
average IoU	46.31	78.73

Initially, the training was organized on 1563 images with a size of 640x480, duration of 2000 epochs, however, the output model had a low accuracy rate for objects located further than 10 meters. To solve this problem, affine transformations were applied. In turn, affine transformations create new images by changing the color, tilt, reflection, zooming, scaling, noise of the original image. In [15] it is said that the use of affine transformations is not enough for a greater increase in accuracy, but as a result of additional training on the changed

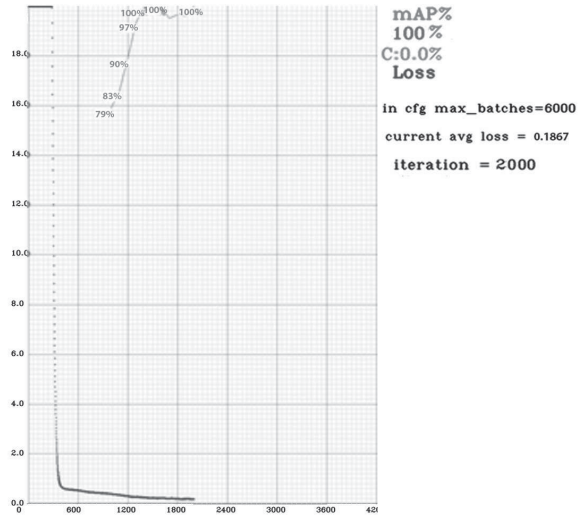

Fig. 7. Transfer learning results for training on dataset without affine transformations in 2000 epochs

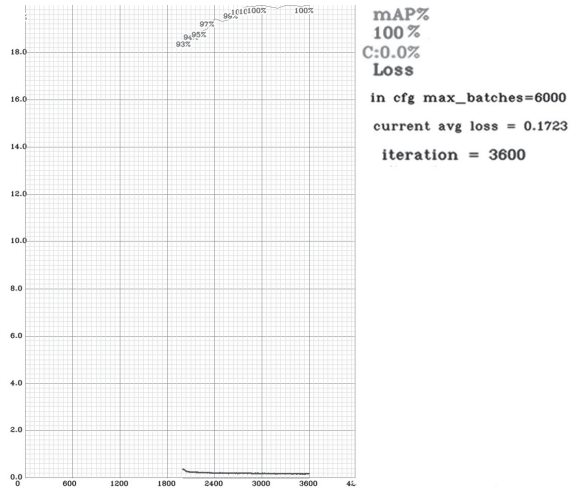

Fig. 8. Transfer learning results for training on dataset with affine transformations in 1600 epochs

data, there was a significant increase in accuracy (Table I). The increase in accuracy is due to the fact that the initial training was performed on similar, equidistant generated data (Fig.7). When testing the detection of landmark objects of the model was carried out on remote data, the neural network could not recognize small objects, however, scaling of the data improved maP and IoU. In turn, additional training took place over 1000 epochs. Training results on dataset without affine transformations (Fig.7) were achieved in the interval of 2000 iterations with the mAP metric 100%, when additional training of the model with affine transformations (Fig.8) required 1600 iterations with mAP 100%.

The next experiment was aimed at measuring mAP with and without affine transformations. The measurement performed on the data created in the Unity3D environment. The data was created in a way to simulate snow conditions, the distance

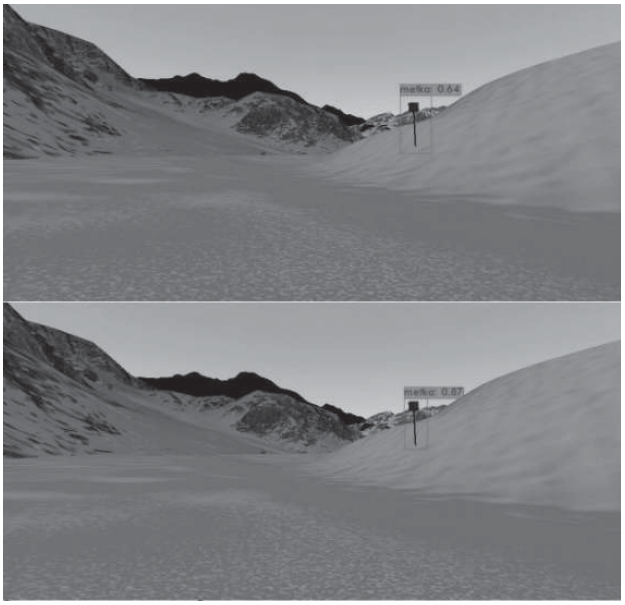

Fig. 9. Test results for models without affine transformations AP=0.64% (top) and with affine transformations AP=87% (bottom)

from the camera to the object is comparable to the real proportions. As a result of the experiment, 100 images were measured at a distance of 10 meters, 20 meters, 30 meters from the camera to the object. The results of the experiment are presented in the table II and in the figure Fig.10. Testing on images generated in an artificial testing environment, where the location of the mark varied from 5 meters to 20 meters, the affine transformations model (Fig.11) produced a result with a mAP metric of 100% and an average IoU metric of 78.73%, when on the same data the model without affine transformations (Fig.12) gave a result with a mAP metric of 46.31% and an average IoU metric of 42.75%.

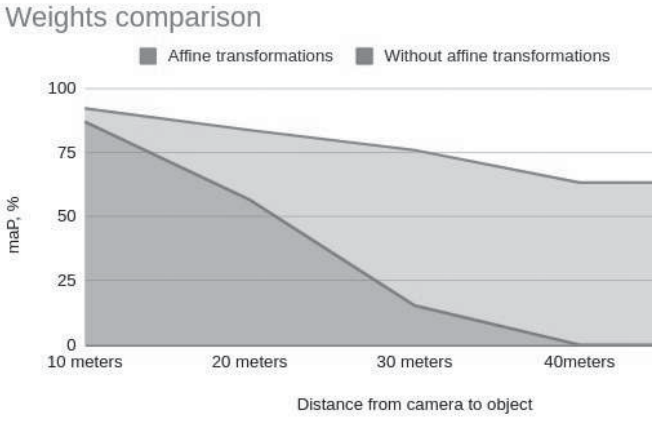

Fig. 10. Graph comparing the read tag-landmark for the data set using the affine transformation and a set of data without the use of affine transformations

```
 calculation mAP (mean average precision)...
 Detection layer: 30 - type = 28
 Detection layer: 37 - type = 28
52
 detections_count = 90, unique_truth_count = 48
class_id = 0, name = metka, ap = 100.00%         (TP = 46, FP = 0)

 for conf_thresh = 0.25, precision = 1.00, recall = 0.96, F1-score = 0.98
 for conf_thresh = 0.25, TP = 46, FP = 0, FN = 2, average IoU = 78.73 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 1.000000, or 100.00 %
Total Detection Time: 1 Seconds
```

Fig. 11. Transfer learning results for generated landmark data with affine transformations in 2000 epochs

```
 calculation mAP (mean average precision)...
 Detection layer: 30 - type = 28
 Detection layer: 37 - type = 28
52
 detections_count = 175, unique_truth_count = 48
class_id = 0, name = metka, ap = 46.31%          (TP = 22, FP = 15)

 for conf_thresh = 0.25, precision = 0.59, recall = 0.46, F1-score = 0.52
 for conf_thresh = 0.25, TP = 22, FP = 15, FN = 26, average IoU = 42.75 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.463110, or 46.31 %
Total Detection Time: 0 Seconds
```

Fig. 12. Transfer learning results for generated landmark data without affine transformations in 1600 epochs

TABLE II. COMPARISON OF MAP MEASURE FOR A DATASET WITH AFFINE TRANSFORMATIONS AND WITHOUT AFFINE TRANSFORMATIONS

	mAP with affines, %	mAP without affines, %
10 meters	92.18	86.93
20 meters	83.71	56.49
30 meters	75.91	15.33
40 meters	63.28	0

V. CONCLUSION

Outdoor landmarks detection system using YOLOv4 algorithms, followed by the deployment of NVIDIA Jetson Xavier was proposed in this paper. In order to deal with the lack of a real-world data YOLOv4 neural network was trained using synthetic dataset. YOLOv4 network structure was tweaked to create more robust landmark detection functionality. This training method is suitable for creating a model with no or a small amount of data.

Further step of the research will be the development of a detection system based on NVIDIA Jetson AGX Xavier. For this, a system consisting of three levels was developed. The first level of the system is reading the landmark by using the YOLOv4 neural network. This system was presented in this article. The second level of the system is the tag object proximity module based on the apriltag module in the NVIDIA SDK. The third level is a module for reading a landmark and placing coordinates on a known map of the navigation area. This system was presented in this article.

REFERENCES

[1] D. Wells, N. Beck, A. Kleusberg, E. J. Krakiwsky, G. Lachapelle, R. B. Langley, K.-p. Schwarz, J. M. Tranquilla, P. Vanicek, and D. Delikaraoglou, "Guide to gps positioning," in *Canadian GPS Assoc*. Citeseer, 1987.

[2] W. Zhang, N. Liu, and Y. Zhang, "Learn to navigate maplessly with varied lidar configurations: A support point based approach," *arXiv e-prints*, pp. arXiv–2010, 2020.

[3] J. Fickenscher, S. Schmidt, F. Hannig, M. E. Bouzouraa, and J. Teich, "Path planning for highly automated driving on embedded gpus," *Journal of Low Power Electronics and Applications*, vol. 8, no. 4, p. 35, 2018.

[4] P. S. Zaki, M. M. William, B. K. Soliman, K. G. Alexsan, K. Khalil, and M. El-Moursy, "Traffic signs detection and recognition system using deep learning," *arXiv preprint arXiv:2003.03256*, 2020.

[5] C. Wu, S. Xu, G. Song, and S. Zhang, "How many labeled license plates are needed?" in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 2018, pp. 334–346.

[6] *Nvidia Isaac Documentation*. [Online]. Available: https://docs.nvidia.com/isaac/isaac/doc/overview.html

[7] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[9] W. Yu, K. Yang, Y. Bai, T. Xiao, H. Yao, and Y. Rui, "Visualizing and comparing alexnet and vgg using deconvolutional layers," in *Proceedings of the 33 rd International Conference on Machine Learning*, 2016.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[12] S. Zhang, Y. Wu, C. Men, and X. Li, "Tiny yolo optimization oriented bus passenger object detection," *Chinese Journal of Electronics*, vol. 29, no. 1, pp. 132–138, 2020.

[13] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," 2013.

[14] *Quality metrics and loss functions 3D ML problem*. [Online]. Available: https://medium.com/phygitalism/3d-ml-metrics-loss-functions-9708ff0476e2

[15] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW)*. IEEE, 2018, pp. 117–122.