STV is based on object detection DNN for forming a semantic model of the industrial-urban environment.

Mikhail Podvalkov
Email: mikhailp@colostate.edu

The RTCs (robot technical complex) and UAVs (unmanned aerial vehicles) become more self-automated and reduce the control of a human expert. Moreover, robots are increasingly used in specific areas with problems that do not allow standard methods and teachings, such as in the outdoor environment. The critical problem is navigation robots, especially indoor space. Most navigation robots could not use inside the industrial-urban environment, such as GPS, etc. Therefore, create a navigation map for tracing and positioning robots in the space. It is necessary to use the robot resource such as a sensor to create data for the map and upper control levels of the robot to process this data. Another problem of navigation in the indoor space is the algorithm running time, which should allow choosing the movement trajectory in real-time, approximately equal to 24 FPS (frame per second). The paper focused on the neural network object detection methods, which fined the static indoor item received from the RGB camera for subsequent map creation. The methods chosen in this article are SSD methods (single short detector) with backbone neural network models such as Mobilenet. The SSD mobile net trained on the COCO dataset and the subset of COCO with only indoor objects such as the chair, bed, couch, person, etc. The resulting model implemented on NVIDIA Jetson Nano simulated the technical capabilities of the robot. The result was upper of the required, allowing navigating robot in real-time. Moreover, it was faster than the other methods based on depth sensors.

Keywords—robots, object detection, SSD, Tensorflow, NVIDIA Jetson Nano

I. INTRODUCTION (*HEADING 1*)

Currently, existing ground-based RTCs and UAVs for various purposes are equipped mainly with remote control systems. The number of operations carried out in urban environments and buildings (urban and industrial zones, cinemas, railway stations and airports, power units of nuclear power plants, etc.) with the involvement of robotic means is quite large, and it is a steady growth trend according to expert [1]. Under these conditions, RTC and remote-controlled UAVs have several fundamental limitations due to the shortcomings of the communication channel (short-range or complete absence of a communication channel due to shielding by buildings). Therefore, RTC and UAV with semi-autonomous and completely autonomous control systems are promising for these conditions. As shown by theoretical and practical experience [2-4], the central and most difficult tasks in the creation of autonomous control systems are closely related tasks of navigation (determining the current linear and angular coordinates of the control object) and reconstruction (building a volumetric model) of the surrounding space.

The solution of the complete navigation problem with the required accuracy using traditional navigation aids in the conditions under consideration is impossible due to shielding of navigation fields (for example, GPS) and distortion of the Earth's magnetic field, as well as due to insufficient accuracy of inertial and odometric systems. Under these conditions, extreme navigation methods based on processing a sequence of images obtained by a scanning device in the movement process [5-12] are promising, mainly since the same data are used to reconstruct the surrounding space [8-10].

The majority of RTCs and UAVs use an RGB camera or/and depth sensor as a primary visual sensor. The information from this type of sensor is quite complex and could not be used in its pure form. Therefore, correct sensor data processing becomes an essential aspect of creating a map to solve a navigation map.

II. RELATED WORK

One method is to use the depth sensor to create the navigation map. The use of initial point clouds, directly generated by depth sensors, as a rule, of a large volume, is not always possible, especially for onboard control systems. Therefore, the problem of compressing the initial data without losing geometric and navigation information becomes urgent.

Direct use of the initial point clouds formed by the ranging STV (system technical vision) requires significant computational resources. For example, the HDL-32E 3D laser sensor generates 700,000 points, while the Asus Xtion Pro sensor generates 307,200 points 30 times per second. With such a large flow of information, even with high-performance computing technology, problems arise of its storage and processing in real-time. Therefore, the problem of compressing the initial information becomes urgent. One of the possible and promising ways of compressing the initial long-range images of artificial environments (various rooms, buildings, factories, and urban areas) is their representation in the form of a set of geometric primitives: planes, faces, lines, segments, edges, vertices, angles. Compared to the original image, such linear geometric objects are characterized by a much lower dimension and noise, as well as the possibility of finding various relationships between them and transitioning to a semantic description of the surrounding space

Compared to the original image, such linear geometric objects are characterized by a much lower dimension and noisiness and the ability to find various relationships between them and transition to a semantic description of the surrounding space. For example, instead of the coordinates of a set of points that fall on one plane, it is enough to remember the equation of this plane and its semantic description ("wall Si," "ceiling," "floor," "ramp Aj," "step Sk") and the equations of boundary lines. It significantly reduces the amount of stored information required to describe the external environment. This approach makes it possible to solve problems of the SLAM class more efficiently.

The most common method is RANSAC, Hough Transform, Local Normal Algorithm. RANSAC iterative methods search for subsets of points belonging to separate planes with a given tolerance. Algorithms like RANSAC are robust to input data noise but become sluggish as the number of points in the cloud and the number of planes in the frame increase. Hough Transform a 3D range-based image, represented as a set of points with coordinates <x, y, z> in a 3D $x \times y \times z$ space. The Hough transformation, in this case, is based on the use of the three-dimensional space of parameters $\rho \times \theta \times \varphi$, in which the search for the planes presented in the standard form. Algorithm of local normals of source point clouds by a sensor in an industrial-urban environment, subsets of points is found that fall on natural flat objects that exist in the real world. Explicit orientation and curvature information on these surfaces is lost during machining. The local normals extraction algorithm reconstructs this information by constructing a set of orthogonal vectors to the tangent plane passing through each point. The calculation of the normal vectors is a so-called local descriptor. The running time implementation of the RANSAC, Hough, and Local Normal algorithms is 29.6, 2, and 2.25 seconds which imposes restrictions on real-time control.

III. DATA

Another method of reducing complexity and simplifying is using RGB camera data and finding a static object whose position does not change during the robot movements. The position data of this object and the robot traveled path is enough to reconstruct the map, which could use in the navigation of the task. The most challenging part of this method is detecting an object from an RGB camera because this detection needs to be in real-time to have the ability to use in the robot control system. One of the ways to achieve this task is to use a DNN for object detection. The first and most critical part of problem-solving is finding the data that will suit the problem. The primary criteria for the data are the indoor images because it is the workspace of the robot and the presence of the fixed used in navigation.

One of the most used suitable for these tasks is the MS COCO dataset. MS COCO dataset stands for Common Objects in Context produced by Microsoft. The COCO dataset is used for many computer vision tasks such as object detection, segmentation, etc. The dataset contains high-quality images which could be used for any method in object detection. The number of images in the dataset equals 328 000, typically split 164,000 training, 82,000 validation, and 82,000 testing images. Also, the COCO dataset has 91 classes.

Also, the COCO dataset has more than 8000 images shown in Fig 1. Moreover, let compare the other datasets such as PASCAL VOC 2012, ImageNet, and the SUN [13]. We can conclude that MS COCO is better than others in object detection and segmentation in a natural context. Moreover, the COCO dataset has much more instances per category, which is better for the model's accuracy.

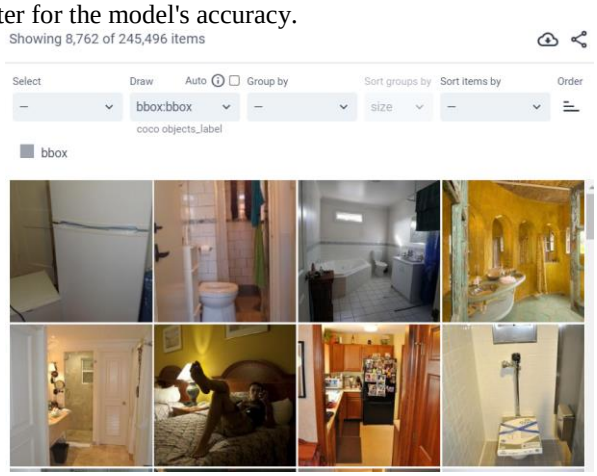

Fig. 1. The indoor subset of MS COCO

The COCO dataset has many classes, and most of them are for the object us never appear in the indoor environment, such as cars, airplanes, etc. Therefore, the data used in the project contain the indoor subset to increase accuracy; we mean detecting the wrong object. The subset contains a moving object such as a toothbrush; reduce it. The final subset of the COCO dataset contains 16 classes presented below

```
'person','chair', 'couch', 'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'clock', 'vase'
```

The class person is used, although it could be indoor, outdoor, and movable. Thus, detecting a person in the robots is an essential task of robots control because used in a lot of critical tasks such as emergency stops.

IV. METHODS

There are three standard methods to create a navigation map based on object detection. The methods used in this paper are based on RGB cameras. All three methods are pretty fast and do not require the high performance of the processing device.

The first method is based on changing the size of the fixed object. The shortening of the distance to the object is the change filling of the frame. The robot's closer to the object, the more space it takes on the frame shown in Fig 3. Using this changing of the size and the angle of the RGB camera, it could find the distance to the robot. The distance will proportionately become more significant than the closer robot to the object. Form this statement.

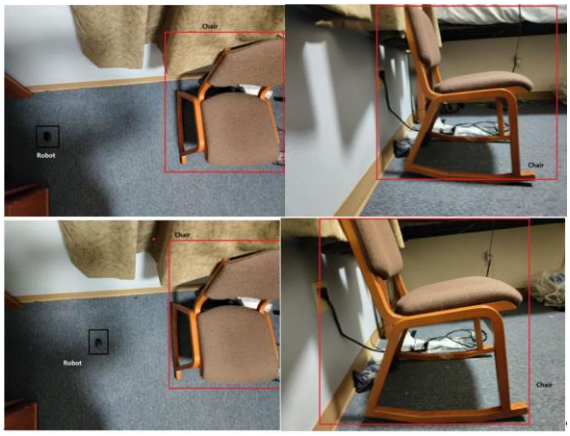

Fig. 2. The fixed object box changing

Form Fig 3, we can create a triangle that allows us to find the distance to the object. The triangles are similar, allowing the find the ratio between the sides of a triangle shown in Fig 4. The formula of ratio (1) is presented:

$$\frac{\Delta h + h}{d + r} = \frac{h}{r} \quad (1)$$

where -Δh is changing the width; - h is the width in the closet position; - d is the distance traveled; -r is the distance to the object. The final equation (3) derived from this proportion is presented below.

$$r = \frac{dh}{\Delta h} \quad (2)$$

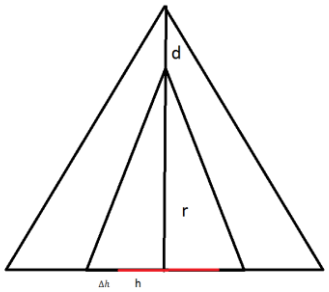

Fig. 3. The similar triangles

The second method for determining the distance from images obtained using stereo vision is one of the options for determining the distance to the required object [14]. The method involves using two identical cameras with a particular distance between them, called the base.

In the case of two identical cameras with parallel optical axes, the distance to point is defined as

$$r_i = \frac{fd}{x_1 - x_2} \quad (3)$$

where f is the focal length; d is the distance between cameras; x_1 and x_2 – coordinates projections on the left and right images. For a more convenient practical application of formula (3), we represent it in the form

$$r_i = \frac{dH}{tg\alpha(x_1 - x_2)} \quad (4)$$

where d is the base (distance between cameras); H- horizontal image resolution;

α - camera viewing angle; x_1 and x_2 coordinate of the first and second cameras, coordinates of the point to which the distance is respectively determined.

The difficulties of using this method lie in the complexity of the correct installation of two cameras: the axes of the cameras must be parallel to each other and perpendicular to the line connecting the centers of the cameras. Incorrect installation of cameras can result in a very significant measurement inaccuracy (the difference in one degree can be more than double the error). Using the methods in the algorithm rectifies images, but this leads to a severe complication. It is proposed to increase the base distance of the same order as the measured item. From the condition that the sensitivity of determining the distance was high (i.e., so that a change in the pixel difference per unit leads to a change in the determined distance by no more than 5%), it is possible to determine, starting from what difference of pixels formula (4) should be applied:

$$r_1 - r_2 = 0.05\, r_1 \quad (5)$$

$$\frac{0.95dH}{tg\alpha\Delta x_1} = \frac{dH}{tg\alpha\Delta x_2} \quad (6)$$

$$\Delta x_1 = 0.95\Delta x_2 \quad (7)$$

Taking into account that 1 $\Delta x_1 = \Delta x_2 + 1$, we obtain $\Delta x_1 = 19$. It is possible to determine the distance to the object using stereo vision. In order to be able to determine the distance up to 3m

at an angle view of one camera α = 160® and horizontal image resolution equal to 320 pixels, we calculate by the formula (3) the based

$$d = \frac{r_1 tg\alpha(x_1 - x_2)}{H} = \frac{3 * tg160° * 19}{320} = 0.06\,m \quad (8)$$

The last third method, based on sampling robot movement of all traveling time, is the simplest. The robot records the trajectory from the previous sample to the current sample. At the sample of the diapering, it records the object's place on the previous sample and the trajectory and uses it to reconstruct the map of surrounding space. Also, the robot tries to detect as many static objects as possible to detect the frame when it disappears from the frame.

All three methods are similar and do not require massive computations, but object detection is the most critical aspect of all methods.

A. Object Detection

The days when object detection was solved exclusively by classical machine learning (cascades, SVM ...) are over - now approaches based on the Deep Learning model. The first R-CNN model approach was proposed in 2014 that significantly influenced research and development in object detection. Its improvements (in the form of Fast R-CNN and Faster R-CNN) and becoming more accurate is why it is used today.

In addition to R-CNN, many more approaches implement object search: the Yolo family, SSD. Some offer an alternative approach, while others develop the current one towards increasing the performance indicator; first, they need to define the terminology of the object detection task.

- Bounding box - coordinates that bound a specific area of the image - most often in the form of a rectangle. 4 coordinates can represent it in two formats: centered (c_x, c_y, w, h) and normal ($x_{min}, y_{min}, x_{max}, y_{max}$.
- Hypothesis (Proposal), P - The object is supposed to be located in a specific region of the image called proposal (specified using a bounding box).
- End-to-end training is when raw images are received at the network's input and ready-made answers at the output.
- IoU (Intersection-over-Union) is a metric of the degree of intersection between two bounding boxes.

One of the first approaches used to determine the location of an object in a picture is R-CNN (Region Convolution Neural Network). Its architecture consists of several sequential steps:

- Determination of a set of hypotheses.
- Extracting features from putative regions using a convolutional neural network and encoding them into a vector.
- They classify the object within the hypothesis based on the vector from step 2.
- Coordinates hypothesis is an improvement (correction) on this step.
- Everything is repeated from step 2 until all hypotheses from step 1 have been processed.

These were the so-called Two-Stage (or Region-based) approaches. In parallel with them, analogs in DL-detection were developed - One-Stage approaches. These include such neural networks as Single-Shot Detector (SSD), You Only Look Once (YOLO)

The basic concept of YOLO is to build a CNN network for tensor prediction. It uses CNN to reduce spatial dimension with output channels at each location. YOLO performs linear regression using two fully connected layers to make bounding box predictions.

The above methods have their pros and cons. The RNN accuracy is enough for navigation, but the speed is far from the required value for a control robot in real-time. However, YOLO and SSD could not reach that accuracy, but it is much faster [15]. Unlike YOLO, SSD could find a smaller object because features were taken from the last convolutional layer of the network. The detection of the small object is more important for navigation because then, before NNs find the critical object, then more precisely becomes a navigation map more precisely.

B. SSD

The article used an SSD for the reasons stated above. Single Shot MultiBox Detector is a deep learning model used to detect objects in an image or video source. A single shot detector is a simple approach to solving the problem, but it has been very effective so far. An SSD consists of the backbone model and the SSD head. The Backbone Model is a pre-trained image classification network as a feature extractor. Typically, the fully linked classification layer is removed from the model. SSD Head is another set of convolutional layers added to this backbone. The output is interpreted as bounding boxes and feature classes in the spatial location of the activations of the last layer.

Instead of using the traditional sliding window algorithm, the SSD divides the image into grids, and each grid cell is responsible for detecting objects in that area of the image. If the object is not found, we output it as "nothing," or, to be more precise, we put "0" to indicate that the object was not found.

The problem of when are many objects of the same instance in one image could solve by using Anchor Box. Anchor blocks are assigned multiple anchors/anchor blocks that are predefined and have a fixed size and shape in a grid cell. Based on this, we can detect multiple objects in the image.

What is offered? Suppose we have some m × n feature map obtained from one of the convolutional layers of the neural network. Let us go through it with a convolution with 3 × 3 kernels, which would output 4 + Cl channels at the output. Cl - the number of classes that we want to detect plus "background" and 4 is, as in YOLO, some description of the object's rectangle. SSD divides the image with a grid, like in YOLO, because the

feature at the convolutional layer output absorbs information about the pixels of a square in the original image and, therefore, can detect an object located in this square. The earlier the layer we use to pick up the feature map, the larger its size (i.e., m and n) and the smaller we can detect. The converse is also true since we detect objects on a feature map with a kernel of a fixed size 3 × 3, then in order to find large objects in the image, we need to take feature maps from the last layers of the neural network.

To make it easier for the detector, instead of one 3 × 3 × (4 + Cl) convolution. Each convolution will target objects with a specific aspect ratio (aspect) and scale. Except for an object whose bounding rectangle coincides with a set of pixels contributing to this feature (receptive field), we will add more objects a little less, a little more, stretched horizontally and stretched vertically. If recalling RCNN, there were also anchors, each of which was responsible for the object's position and aspect ratio. However, in RCNN, the features were taken from one, the last layer of the network, and here we use intermediate layers, which should improve separability detected objects.

C. Backbone SSD

The SSD architecture needs to have a backbone of the network in this project. The backbone model usually is a pre-trained image classification network as a feature extractor. Moreover, it is better to use a pre-trained network on NVIDIA core[16]. Backbone Network is typically a trained network without a final connected classification layer. The deep neural network is used to find semantic features of the input image while preserving the spatial structure of the image, albeit at a lower resolution. This network is critical to all real-time performance because the speed and methods used by the network directly determine the FPS result of all Object detection of the SSD method. Figure 1 shows the benchmark of the most popular network model [17]

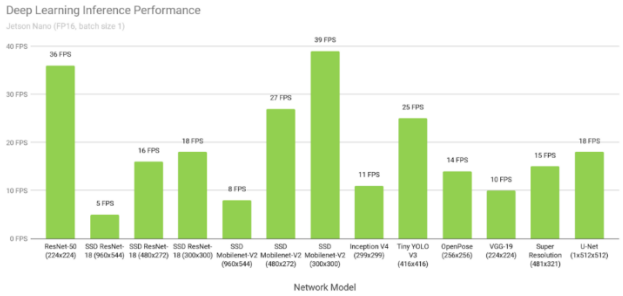

Fig. 4. The benchmark result of the most popular network

Based on this data in this Fig 4, choose MobileNet as the faster network. Besides reducing the memory consumption and increasing the network's accuracy, the SSD MobileNet trained not on all COOC 2017 datasets but on several indoor classes that have increased a network speed.

V. Experiments

The object detection methods solve the half-automatic robot's navigation problem. To emulate a robot system in this project, uses an NVIDIA Jetson Nano and the IMX219-160 camera like an RGB sensor. One of the last major remaining things is to choose the rating system of received methods. The critical parameter for this problem is the robot's time when transferring on object detection connected and depends on FPS (Frame Per Second).

A. Evaluation Parameters

Selection of the minimum required FPS depend on a massive number of factor. However, the most critical is the RGB camera, dimensions of the robot, robot speed characteristic, and input size. The RGB camera is responsible for the maximum range the image remains clear. Also, consider the image input size witch 300 x 300 pixels, which gives the maximum height of the frame occupancy. These two parameters allow the calculation of the work distance of the camera. To find this quantity, we use the Thin lens formula (9)

$$d = \frac{fh}{H} \qquad (9)$$

Where H-is maximum height equal 300 pixels or 0.0005 m; f- is a focal length equal 3,15 mm or 0.00315 m;h- is the height of the object is equal to 1.8 m and d- is work distance. The work distance from formula (9) is equal to 11.34 meters.

The other essential parameters are size, and the speed characteristic for this task takes the maximum robot length of 0.3 m and the max speed of 0.15 m/s. The FPS equals 22.68 round-up 23 FPS for this parameter and works distance.

B. Technical Implementation

To create the SSD MobileNet model necessary to use TensorFlow Object detection API [18]. We installed API Object Detection and pycocotools to create a pre-trained model and re-train it on the needed dataset.

After downloading the COCO data and raw annotation, we modify it to the subcategory discussed in Section III. The penultimate step is to change the configuration file and train the model. TensorFlow uses the record file to work with the pre-train model. The validation and the train annotation could transform two record files. The test dataset annotation was transformed into CSV format and used the updated "generate_tfrecord" [19] to create the test record file.

The NVIDIA Jetson does not work with the TensorFlow model. Therefore, we translated it to the TensorRT model and implemented code [20].

C. Result

The SSD MobileNet is working on the NVIDIA Jetson Nano; some results are shown in Fig. 5.

Fig. 5. Result of the streaming the model

The transformation to the TesnorRT model can be done with different quantization [21], such as float 32, float 16, int 8. The result of the maximum achieved FPS is presented in Table 1.

TABLE I. PERFORMANCE WITH DIFFERENT QUANTIZATION

Parameters	Quantization		
	Float 32	***Float 16***	***Int 8***
FPS	21	23	27

The fastest model is with quantization integer 8. Maintaining similar accuracy with the INT8 model for the same model architecture provides significant inference performance boost on the same hardware compared to the other quantization.

CONCLUSION

The project's final result is creating an object detection modelthath could work on embedded systems like robots. Moreover, the simultaneously with the model could be run other heavy applications such as recording the video, etc. which prof that methods do not consume the all the resources of the system. The low resource use allows us to conclude that collaboration performance of the Object Detection and the reconstruction algorithm based on the data form detector is possible. Furthermore, the best performance is low time-consuming, which allows the use of the SSD MobileNet in real-time.

The Object Detection API is a convenient tool for creating a custom model that is not inferior in any other way. Also, the SSD method is robust and has resource intensity to fit the majority of object detection tasks.

The future extensions of the project are to add the reconstruction algorithm to create a navigation map and implement it on the real robot. Also, comparing the methods based on object detection with classic navigation tasks such as path planning is necessary for future research. Moreover, the new application is to rewrite the methods to fit task associated with UAVs and implement it in the actual vehicle.

REFERENCES

[1] . Lapshov V.S., Noskov V.P., Rubtsov I.V., Rudianov N.A., Ryabov A.V., Khrushchev V.S. Fight in the city. Combat and support robots in an urbanized area // Izvestia SFedU. Technical science. - 2011. - No. 3 (116). - S. 142-146. (in Russian)

[2] Kalyaev A.V., Noskov V.P., Chernukhin Yu.V., Kalyaev I.A. Homogeneous control structures of adaptive robots. - M .: Nauka, 1990 .- - 147 p. (in Russian)

[3] Noskov V.P., Rubtsov I.V. Experience in solving the problem of autonomous control of the movement of mobile robots // Mechatronics, automation, control. - 2005. - No. 12. - C. 21-24.

[4] Lakota N.A., Noskov V.P., Rubtsov I.V., Lundgren J.-O. Moore F. Experience of using elements of artificial intelligence in the control system of a workshop transport robot // Mechatronics, automation, control. - 2000. - No. 4. - S. 44-47. (in Russian)

[5] Noskov V.P., Noskov A.V. System of extreme navigation of a workshop transport robot // Sb. Scientific. tr. "Artificial intelligence in technical systems." - M .: State. ISSP, 1998. - S. 136-144. (in Russian)

[6] Noskov A.V., Noskov V.P. Recognition of landmarks in long-range images // Collection "Mobile robots and mechatronic systems." - M .: Publishing house of Moscow State University, 2001. - P. 179-192. (in Russian)

[7] Noskov V.P., Noskov A.V. Navigation of mobile robots by long-range images // Mechatronics, automation, control. - 2005. - No. 12. - C. 16-21. (in Russian)

[8] Noskov A.V., Rubtsov I.V., Romanov A.Yu. Formation of a unified model of the external environment based on information from a video camera and a rangefinder // Mechatronics, automation, control. - 2007. - No. 8. - S. 2-5. (in Russian)

[9] Kazmin V.N., Noskov V.P. Volumetric vision in the navigation support system of an unmanned aerial vehicle. Vestnik MGTU im. N.E. Bauman. Ser. Mechanical engineering. - 2012. - Issue. Special robotics. - S. 113-121. (in Russian)

[10] Zagoruiko S.N., Kazmin V.N., Noskov V.P. UAV navigation and 3D-reconstruction of the external environment according to the data of the onboard STZ // Mechatronics, automation, control. - 2014. - No. 8. - P. 62-68. (in Russian)

[11] Segal A., Haehnel D., Thrun S. Generalized-ICP // Proc. of Robotics: Science and Systems, RSS, 2009.(in Russian)

[12] Mitra N., Gelfand N., Pottmann H., Guibas L. J. Registration of Point Cloud Data from a Geometric Optimization Perspective // Proceedings of the 2004 Eurographics / ACM SIGGRAPH symposium on Geometry processing. - 2004. - P. 22-31

[13] Lin, T.-Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C. L.; and Dollar,' P. 2015. Microsoft COCO: Common Objects in Context. arXiv:1405.0312.

[14] E.V. Legchekova and O.V. Titov "Determination of the distance to the object in the areas of vehicle movement, using the analysis of video data" (in Russian)

[15] Dr. Sudeep Pasricha Lecture 10.1: Advanced Machine Learning Models Part 1: Vision and Audio page. 67

[16] The official NVIDIA site: https://docs.nvidia.com/tao/tao-toolkit/text/overview.html#pre-trained-models

[17] The official NVIDIA site: https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks

[18] The official TensorFlow site: https://github.com/tensorflow/models/tree/master/research/object_detection

[19] The official GitHub site: https://github.com/datitran/raccoon_dataset/blob/master/generate_tfrecord.py

[20] The official GitHub site: https://github.com/dusty-nv/jetson-inference

[21] The official NVIDIA site: https://docs.nvidia.com/deeplearning/frameworks/tf-trt-user-guide/index.html