Detection of Obstacle Features Using Neural Networks with Attention in the Task of Autonomous Navigation of Mobile Robots

Kirill Sviatov[1(✉)], Alexander Miheev[1], Sergey Sukhov[2], Yuriy Lapshov[1], and Stefan Rapp[3]

[1] Ulyanovsk State Technical University, Ulyanovsk, Russia
{k.svyatov,y.lapshov}@ulstu.ru, a.miheev@simcase.ru
[2] Ulyanovsk Branch of the Institute of Radio Engineering and Electronics, V. A. Kotelnikov of Russian Academy of Science, Moscow, Russia
s_sukhov@hotmail.com
[3] Darmstadt University of Applied Sciences, Darmstadt, Germany
stefan.rapp@h-da.de

Abstract. This article describes the design process of a software package for image recognition of a mobile robot camera using neural networks with attention, which allows to identify the probability of a robot colliding with obstacles standing in its way. A key feature of this software is using a dataset that is prepared without manual labeling of all obstacles and the probability of a collision.

Currently, an important task in mobile robotics is the need to use numerous heuristics and deterministic algorithms in control programs along with neural networks. The use of a single neural network that solves all the tasks of scene analysis (the so-called "end-to-end" solution) is impossible for several reasons: the high complexity of the training samples due to the large parameter space of the environment of the robot and the insufficient formalization of these parameters, as well as the computational complexity of machine learning algorithms, which is critical for mobile robots with strict energy requirements. Therefore, the development of a universal algorithm (end-to-end) is a laborious process.

The article describes a method that allows to use weakly formalized parameters of the robot environment for training convolutional neural networks with attention using the obstacle recognition task. At the same time, weak formalization reduces the time-consuming process of manual data labeling due to automatically generated datasets in the NVIDIA Isaac environment, and the attention mechanism allows increasing the interpretability of the analysis results.

Keywords: Artificial intelligence · Neural networks · Machine learning · Computer vision · Attention networks · Self-driving car

This work was supported by RFBR Grant 18-47-732004 p_мк and 18-07-00989 a

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12254, pp. 1013–1026, 2020.
https://doi.org/10.1007/978-3-030-58817-5_72

1 Introduction

One of the first steps in the process of autonomous navigation of a mobile robot is to analyze the external environment in order to build a model. During the analysis, the location of obstacles on the path of the robot is determined. This can be done with deterministic algorithms using specialized, expensive sensors (e.g. lidars). But in the mass use of mobile robotics, it is preferable to use ordinary cheap video cameras. Also, navigation methods based on video cameras allow to use more information about the environment and reduce the integral error when building a map and localizing the robot in space. But in this case, there is no simple way to detect formalized features: dynamic and static obstacles on the robot path, for example, other mobile robots, the road itself, or the walls.

In this paper, we consider the solution of three tasks for identifying obstacles on the way of a mobile robot:

1. Development of a software tool for generating a special dataset: a method for generating synthetic data for training a neural network for recognizing obstacles at robot camera image in an NVIDIA Isaac environment.
2. Recognition of weakly formalized features of a scene. A weakly formalized feature is understood as a binary characteristic indicating the danger of collision with an obstacle on the current way of the robot or the absence of such a danger. In this case, a fully formalized attribute would contain complete information about obstacles: their location in the frame, their size, shape, and class of the object. The same object may be an obstacle if it is located on the way of the robot, or it may not be. In addition to the fact of the presence of an object, it is necessary to analyze its position relative to the robot and other characteristics. Analysis criteria can contain dynamic properties of objects, which affects the time and cost of labeling the training dataset in the case of using neural networks or the development of heuristic algorithms. Only static objects are considered in this article, moving objects will be considered in following papers. This paper describes an approach in which a weakly formalized binary feature is used as the target feature for the training sample, indicating whether there is an obstacle for the robot or not: the neural network tries to automatically search for patterns in the complex features of scene objects, in order to determine the presence of an obstacle. This will significantly reduce the cost of labeling complex data.
3. Neural network unsupervised training for localization of obstacles. It is shown that a convolutional neural network with “attention” can train to localize obstacles in a camera frame more accurately than a regular convolutional network. It also allows to localize an object in a camera frame without additional cost for labeling the dataset compared to the classification task. Moreover, the resulting model can be used to interpret the results of the neural network, i.e. to clarify the reasons for its decisions, which is extremely important in the tasks of autonomous control of mobile robots, when decisions are made related to the safety of people.

2 Literature Review

Despite the success of using neural networks in image processing, they usually perform only certain processing steps: object detection, localization in the image, and class prediction. It is also possible to determine all available objects in the image for subsequent analysis, for example, a geometric assessment of the relative position of objects, their size, and shape. From this a decision is made on the parameters of the control actions to achieve the robot's goals.

Recently, a large number of models and approaches have been developed for detecting objects in images. They result in a segmentation with a multi-channel mask at the output [1, 2] or a detection bounding rectangles. A number of modifications of these methods for quality and speed [3–5] have been reported.

Navigation of the robot is performed dynamically because the state of the robot and the environment is constantly changing. Therefore, the same objects may be obstacles, or they may not be. It depends on the position of this object. Generally, it is better to detect obstacles earlier. This effect is similar to the rewards in Reinforcement Learning. Within this type of training, there are several methods for solving the problem of training a model with rewards or predicting the occurrence of an obstacle. In some approaches, neural networks perform the entire cycle of scene analysis and decision making [6–9]. But this approach is not so popular due to the low interpretability of the results of the network, which is expressed in the fact that a person does not understand the reasons for decisions. In addition, the use of Reinforcement Learning is a difficult task to implement with a number of disadvantages (instability, long training, etc.). Therefore, in this paper, the main focus is on simple "end-to-end" approaches that are able to solve the problem.

The attention mechanism in neural networks [10] is more popular now among researchers [11], because it allows us to interpret the results of the models and perform selection according to positions of objects in feature maps. Despite the fact that the technology appeared only several years ago, the range of application is very wide [12].

3 Offered Approach

The work [17] describes the problem of recognizing the relationship between objects in the image. The authors note that the variability of relations between objects is too large, therefore it is very difficult to create a one-hot sample to solve a similar problem. They offer an approach based on the search for patterns in textual descriptions of images. Such a network output can be considered weakly formalized, because human speech also has great variability, although textual descriptions of images are much easier to find. The authors use a recursive network that iteratively analyzes objects in the scene and words in the text. However, to solve such a complex problem, the authors use the preliminary extraction of features from the image, namely, they select objects and

spatial information between them. We solve a simpler problem and offer a weakly formalized end-to-end approach.

The paper [18], approaches to the analysis of scenes by mobile robots are considered in detail. These approaches imply a strong formalization of target labels. Objects are first recognized in the scene or the scene is classified (corridor, kitchen, bedroom, etc.). On the other hand, for the navigation task, it is necessary to distinguish one corridor from another for which visual signs are extracted from the images, for example, from hidden layers of the network, which allows to build a graph of the robot's movement. And although the vector of hidden signs is not interpretable and understandable for humans, the distance metric has been successfully used in the space of such vectors.

In the approach described in this paper, the detection of obstacles during movement of the robot can be performed discretely: at each iteration of the program control cycle, the robot detects the state around itself by an analysis of video camera frames and makes a small movement. In this case, the robot always moves co-directionally with the camera. Therefore, objects located in the center of the frame and close to the robot can be recognized as obstacles.

But these criteria for determining the obstacles are known only to the developers. The trained neural network model does not know these rules and assumptions and receives only binary labels "0" or "1" characterizing the absence or presence of an obstacle. The task of a neural network is to predict the presence of an obstacle taking into account its position, size, type of object, and all possible patterns that it can find in the dataset.

To solve the problem, it is necessary to select a random frame from the camera of the robot and a binary mask of the presence of an obstacle for the following cycles of the robot control program. Creating a suitable dataset is one of the main tasks in this research. For classification and segmentation, there are many open data sets, but there is, to the knowledge of the authors, no dataset for obstacle detection as described before. As part of the study, it was decided to create a synthetic dataset using the NVIDIA Issac robotic platform and the built-in simulation tools. Thus, the image from the robot's camera was replaced with a rendering of the virtual 3D world with photorealistic quality. Some papers show that the modern level of game graphics is a way to replace the real-world images for creating training samples [13].

But standard tools are not able to determine the likelihood of a risk of a movement for the robot, therefore, a software component was implemented for automatically labeling the sample based on segmentation masks for the scene, also obtained from the NVIDIA Isaac platform.

Two neural networks were trained in this research: a convolutional network with VGG architecture and a convolutional network with visual attention without a recurrent block in the style of [14].

Structure of the convolutional network (output of the model.summary() method):

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 240, 420, 3)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 238, 418, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 119, 209, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 117, 207, 32)      4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 58, 103, 32)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 56, 101, 32)       9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 50, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 26, 48, 64)        18496
_________________________________________________________________
flatten (Flatten)            (None, 79872)             0
_________________________________________________________________
dense (Dense)                (None, 64)                5111872
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 130
=================================================================
Total params: 5,144,834
Trainable params: 5,144,834
Non-trainable params: 0
```

Structure of the convolutional network with attention (output of the model.summary() method):

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 240, 420, 3)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 238, 418,32)       896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 236, 416, 32)      9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 118, 208, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 116, 206, 64)      18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 114, 204,64)       36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 57, 102, 64)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 55, 100, 64)       36928
_________________________________________________________________
reshape (Reshape)            (None, 5500, 64)          0
_________________________________________________________________
lambda (Lambda)              (None, 5500, 64)          0
_________________________________________________________________
tf_op_layer_Sum_1 (TensorFlo [(None, 5500)]            0
_________________________________________________________________
flatten (Flatten)            (None, 5500)              0
_________________________________________________________________
dense (Dense)                (None, 64)                352064
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 130
=================================================================
Total params: 454,690
Trainable params: 454,690
Non-trainable params: 0
```

4 Generation of the Dataset

To generate a training dataset, it was decided to use the NVIDIA Isaac robotics platform. The NVIDIA Isaac Developer Toolkit (SDK) contains software tools, libraries, GPU algorithms, and components designed to accelerate the development of software components for robots. NVIDIA Isaac can also be used to develop AI solutions optimized for deployment on the NVIDIA Jetson platform.

Other robotic frameworks such as V-Rep, ROS (with the Gazebo simulation environment) have similar functionality to NVIDIA Isaac, but the advantage of the Isaac SDK compared to them is the use of the Unity 3D framework for photorealistic simulation of the environment and robots (ISAAC SIM), as well as a rich set of libraries for solving problems of autonomous control, and the availability of pre-trained neural networks. Also, it contains software integrated with the Cuda SDK for

calculating motion parameters and scene analysis. Due to this, a significant acceleration of such calculations can be realized.

The task of generating the training dataset was formulated as follows: it is necessary to develop a software component for generating pairs of images in a simulation environment: a picture frame from a color camera and an image with color semantic segmentation, in which each pixel of the segmentation mask is mapping a number from 1 to 5 into the value of the RGB tuple (red, green, blue), corresponding to one of five classes:

- Free space for movement (floor)
- Laneline
- Obstacle
- Wall
- Beam

To accomplish this task, the Isaac Sim environment used the "Medium Warehouse" scene and a standard script that allows us to randomly change the position of the camera, perform "teleportation" (changing the position, roll, pitch, yaw). In addition, a segmentation neural network with the UNet architecture (standard Isaac SDK component) was used.

Before launching the "teleportation" commands, it is necessary to launch the Unity 3D environment from the simulator directory. The following parameters of teleportation and camera rotation where use: roll 180 degrees, pitch 68.5 degrees, yaw 90 degrees, the frequency of "teleportation" 30 Hz.

After executing the commands, a window opens with a visualization of the storage scene (Fig. 1).

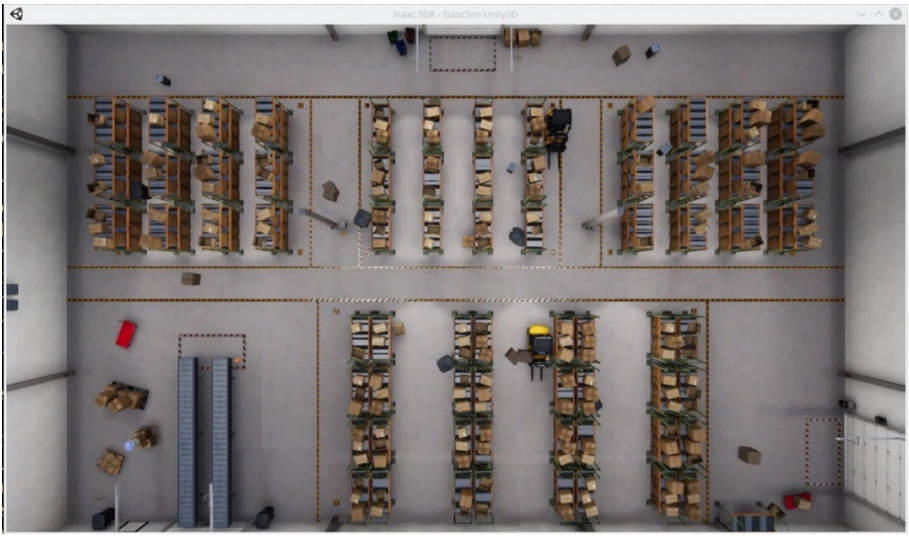


Fig. 1. Visualization of the scene (Medium Warehouse)

When the application (Codelet) is launched in the Isaac SDK environment, the WebSight server (standard visualization application) is launched on port 3000. It allows to display graphs, 2D and 3D scenes, the current state of the application (active nodes and connections), and also to update the configuration. When the "freespacednn" application is launched, images from the camera are displayed in the WebSight environment, as well as the result of the segmentation network with the UNet architecture (network model definition file - packages/freespace_dnn/apps/freespace_dnn_training_models.py).

Camera images and the results of the segmentation network are displayed in the WebSight environment through the channels "ColorCameraViewer/Color" and "Segmentation".

The target training sample should store files with the same names in the "image" and "seg" folders for the corresponding image and segmentation map. To save the camera image and the result of the segmentation network into files, it is necessary to develop an application in the Isaac SDK environment, using the capabilities of this platform.

The developed application consists of four files:

- __init__.py - service file required to run the software package, it may be empty
- BUILD - build file for the Bazel application, for compilation and launching of the software system.
- seg_dataset.app.json - application configuration file, designed in accordance with the requirements of the Isaac SDK.
- seg-dataset.py - application program code (Fig. 2).

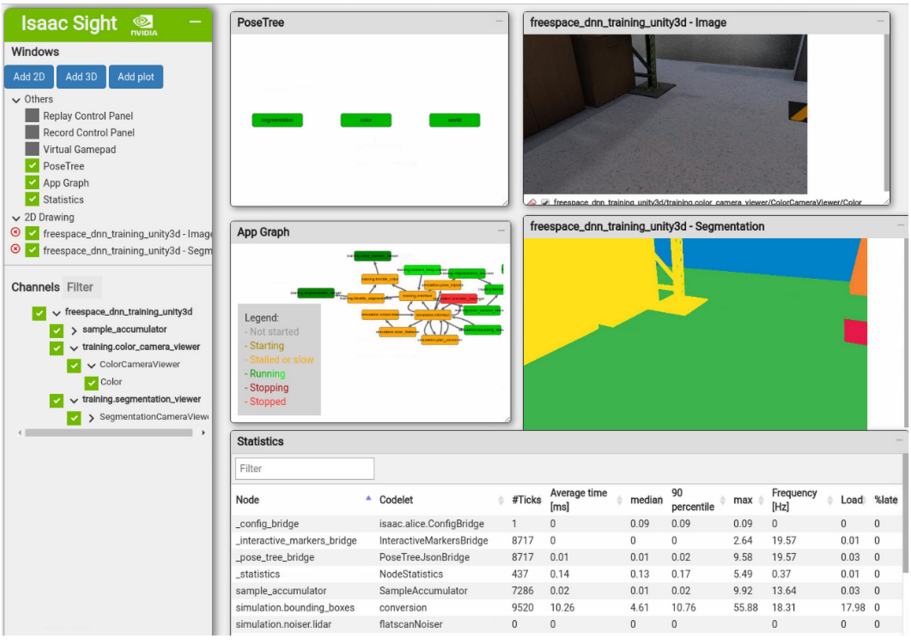


Fig. 2. Screenshot from the WebSight server page

The BUILD file contains parameters for the application launch and execution. The script seg_dataset.py implements the collection of messages from the robot camera, messages from the segmentation network, as well as the generation of a segmentation image, and the saving of all images in the target directory, taking into account the naming rules (the same file names for the corresponding images in the "seg" and "image" folders). To receive images from the camera and the segmentation network, it is necessary to create objects corresponding to the channels for the color image and segmentation data:

```
self.rgb_rx = self.isaac_proto_rx ("ColorCameraProto", "color")
self.seg_rx = self.isaac_proto_rx ("SegmentationCameraProto",
"segmentation")
```

The entry point for the program is the "tick" method, which is automatically called by the Application type object when the developed application is registered in the Isaac SDK environment. Checking for the presence of an image in the data channel is carried out using the self.rgb_rx.available () method.

If the image is available for reading, then the reading is performed by the rgb_image_proto = self.rgb_rx.get_proto () code.

Saving is done using standard python language methods:

```
rgb_image = np.frombuffer (rgb_image_buffer, dtype = rgb_data_type)
rgb_image = rgb_image.reshape ((rgb_rows, rgb_cols, rgb_channels))
im = Image.fromarray (rgb_image)
im = im.resize ((420, 240), Image.ANTIALIAS)
im.save ("/images /color /robot_image _ {}. png".format (num))
```

A segmentation mask is generated similarly with the only difference that segmentation data is a matrix with scalar values corresponding to class numbers. It is necessary to transform them into a color segmentation image (3 RGB channels for each pixel) with some simple code:

```
width, height, channels = image.shape
out = np.empty ((width, height, 3), dtype = np.uint8)
image = np.stack (image, axis = -1)
image = np.stack (image, axis = -1)
# 1 = red 255 0 0
# 2 = blue 0 0 255
# 3 = green 0 255 0
# 4 = cyan 0 255 255
# 5 = yellow 255 255 0
red = lambda x: 255 if (x == 1) or (x == 5) else 0
green = lambda x: 255 if (x == 3) or (x == 4) or (x == 5) else 0
blue = lambda x: 255 if (x == 2) or (x == 4) else 0
out [:,:, 0] = np.vectorize (red) (image)
out [:,:, 1] = np.vectorize (green) (image)
out [:,:, 2] = np.vectorize (blue) (image)
```

As a result, a dataset of synthetic data is obtained, where x (features) is the rendering of the scene, that simulates the video camera data located on the mobile robot, and y (target) is the corresponding mask of scene objects. Each class of scene objects is represented in the mask by its color. In essence, this is an automatic segmentation of a scene (Fig. 3).

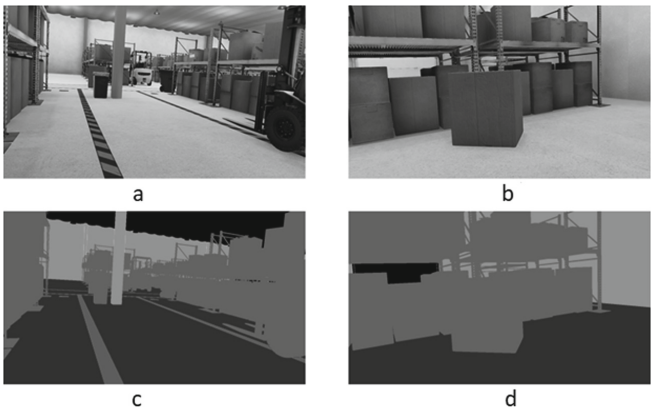


Fig. 3. In the upper part of the figure (a, b) examples of the virtual world renderings are presented. In the lower part of the figure (c, d) the mask of objects corresponding to these rendering are presented. It can be seen that each type (class) of objects in the scene has its own color.

5 Automatic Sample Labeling for the Classification Task

After obtaining the basic sample, it is necessary to change the target variable "y" so that it was is a scene mask, but a binary scalar value (0 or 1) indicating the presence of an obstacle that could be dangerous for the robot during the next program control cycle.

Before solving the dynamic problem, assumptions are made that the current robot path coincides with the center of the image from the robot camera. Such an assumption is typical for control tasks in mobile robotics.

For the binarization of the target parameter, a simple hypothesis is proposed: if objects are located in the central zone of the frame and occupy a sufficiently large area, then they are an obstacle, if they are in another place or occupy a small area, then they are not an obstacle. With a segmentation mask, this hypothesis can be implemented using the OpenCV library.

The algorithm can be represented by the following pseudo-code:

```
function calc_risk(segmentation_mask, triangle_points, area_thresh-
old):
        triangle_mask = draw_triangle_mask(triangle_points)
        roi = image_bitwise_and(segmentation_mask, triangle_mask)
        _, blocks_clr_mask = filter_non_blocks_colors(roi)
```

```
risk_area = count_nonzero(blocks_clr_mask)
risk_status = 0
if risk_area > area_threshold:
      risk_status = 1
return risk_status
```

As an intermediate stage in the execution of this algorithm, the following segmentation masks where obtained (Fig. 4). A simple threshold function based on the object position determines whether an object is an obstacle or not.

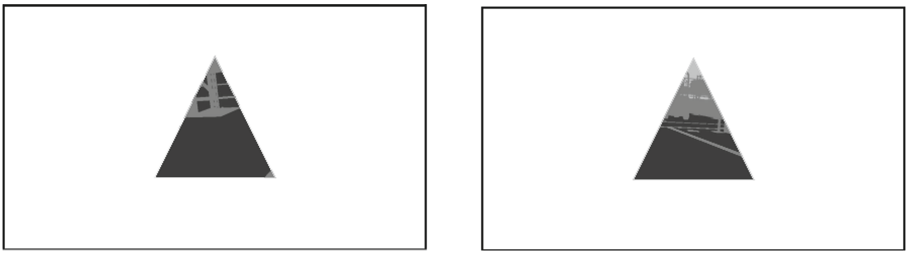


Fig. 4. The appearance of the zone of interest (variable "roi" in pseudo-code) to assess the risk of movement. In this area, the area of potential barriers is estimated on the frame and a binary mark is calculated.

6 Experiments

Only a visual representation of the scene is provided as input for the neural networks without additional information about the objects. A flag characterizing the risk of further movement of the robot is expected as output.

Each model is run 10 times for statistical confidence in the results. The training is run with 15 epochs, after which the models are saturated, the batch size is 32, the total sample consists of 3511 images, 33% of which were validated. After each epoch, the training sample was shuffled (shuffle = True), and between each run (out of 15 iterations), the samples were randomly divided into training and validation subsets.

Table 1 shows that a convolutional neural network successfully finds a correlation between objects, their position in the frame, and risk. A convolutional network with attention works more accurately, also it allows to localize these objects without additional labeling of the training sample.

Table 1. Accuracy of motion risk recognition

Metric	CNN	ATT	P Value by Fisher's criterion
Train Accuracy, %	$\mu = 0.949$, $\sigma = 0.142$	$\mu = 0.981$, $\sigma = 0.005$	$0.22 > 0.05$
Valid Accuracy, %	$\mu = 0.939$, $\sigma = 0.145$	$\mu = 0.947$, $\sigma = 0.015$	$0.24 > 0.05$

A network with attention allows to not only get a class label but also an attention mask. After several epochs of training, visualization of attention masks shows high values on the desired objects. Figure 5 shows that if an object is located on the way of the robot, then it is classified as an obstacle (a, c), and when the object is located far from the center of the frame, then it is ignored (e, g). At the same time, attention covers potential obstacles regardless of their location, but the binary score is determined correctly. In several frames, potential obstacles are practically not visible. This is exactly the behavior that was expected from the system.

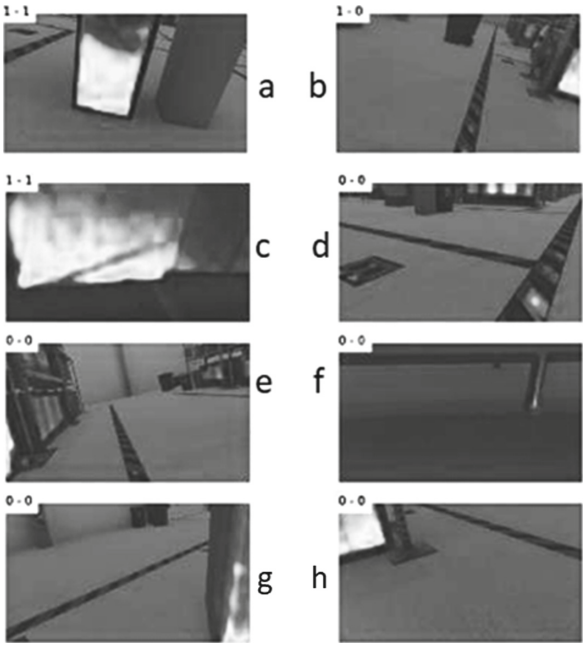


Fig. 5. Heat maps of attention. White signals potential obstacles in the frame.

Next, this mask was binarized using a threshold function and the contours of the proposed objects (obstacles) were selected. With sufficient accuracy of the model, one can localize objects in the frame and evaluate their parameters (area, specific object class, etc.). The whole process is implemented without manual labeling of segmentation masks for training of the network (Fig. 6).

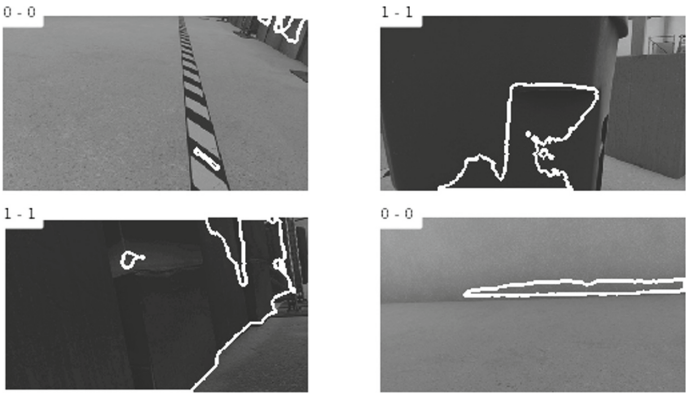


Fig. 6. Selecting objects using attention. The object is considered a connected area with a value of attention above 10% of the maximum level of the mask of attention. In the frame at the bottom left, one can see that the cabinets stand out quite accurately. The box in the upper right frame is fragmented.

In this paper, we do not measure the quality of the selection of the contours of objects (obstacles). This is the task for the next study. The approach itself turns out to be viable, but the accuracy of the result can be improved.

7 Conclusion

The article describes an approach for generating synthetic datasets with automatic marking using the NVIDIA Isaac robotic platform for scene analysis tasks for mobile robots. In the proposed approach, the obstacle detection problem is solved as a binary classification problem using the "end-to-end" convolutional neural network model with an accuracy of 93.9% without complex data labeling or development of additional heuristics.

Using the attention mechanism, the developed approach can be used to obtain additional information about obstacles, such as the location of objects in the frame and their size. The use of attention increases the accuracy of recognition of obstacles in a weakly formalized feature.

Further work can be aimed at improving the accuracy of localization and determining the spatial properties of obstacles (size, contours, distance to the object, etc.), as well as predicting situations of collisions with dynamic objects.

Weak formalizations of the target variable can be applied in a wide range of scene recognition and analysis tasks in addition to obstacle recognition for mobile robots. Attention mechanism allows to select the required objects with weakly formalized data sample.

References

1. Liu, X., Deng, Z., Yang, Y.: Recent progress in semantic image segmentation. https://arxiv.org/ftp/arxiv/papers/1809/1809.10198.pdf
2. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation
3. Tech report (v5). https://arxiv.org/pdf/1311.2524.pdf
4. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. https://arxiv.org/pdf/1506.02640.pdf
5. Girshick, R.: Fast R-CNN. https://arxiv.org/pdf/1504.08083.pdf
6. Zhao, Z.-Q., Zheng, P., Xu, S., Wu, X.: Object detection with deep learning: a review. https://arxiv.org/pdf/1807.05511.pdf
7. Gu, S., Holly, E., Lillicrap, T., Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. https://arxiv.org/pdf/1610.00633.pdf
8. Singh, A., Yang, L., Hartikainen, K., Finn, C., Levine, S.: End-to-end robotic reinforcement learning without reward engineering. https://arxiv.org/pdf/1904.07854.pdf
9. Vezhnevets, A.S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., Kavukcuoglu, K.: https://arxiv.org/pdf/1703.01161.pdf
10. Srinivas, A., Laskin, M., Abbeel, P.: CURL: contrastive unsupervised representations for reinforcement learning. https://arxiv.org/pdf/2004.04136.pdf
11. Xu, K., et al.: Show, attend and tell: neural image caption generation with visual attention. https://arxiv.org/pdf/1502.03044.pdf
12. Wang, W., et al.: Learning Unsupervised Video Object Segmentation through Visual Attention. http://openaccess.thecvf.com/content_CVPR_2019/papers/Wang_Learning_Unsupervised_Video_Object_Segmentation_Through_Visual_Attention_CVPR_2019_paper.pdf
13. Saleh, F.S., Sadegh, M., Salzmann, M., Petersson, L., Alvarez, J.M.: Effective use of synthetic data for urban scene semantic segmentation. https://arxiv.org/pdf/1807.06132.pdf
14. Sun, J., Darbehani, F., Zaidi, M., Wang, B.L.: SAUNet: shape attentive U-Net for interpretable medical image segmentation. https://arxiv.org/pdf/2001.07645v3.pdf
15. Sviatov, K., Miheev, A., Kanin, D., Sukhov, S., Tronin, V.: Scenes segmentation in self-driving car navigation system using neural network models with attention. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11623, pp. 278–289. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24308-1_23
16. Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., Jagersand, M., Zhang, H.: A Comparative Study of Real-time Semantic Segmentation for Autonomous Driving. http://openaccess.thecvf.com/content_cvpr_2018_workshops/papers/w12/Siam_A_Comparative_Study_CVPR_2018_paper.pdf
17. Liao, W., Shuai, L., Rosenhahn, B., Yang, M.Y.: Natural language guided visual relationship detection. https://arxiv.org/pdf/1711.06032.pdf
18. Ortiz, J.C.R.: Scene Understanding for mobile robots exploiting Seep Learning Techniques. https://ridda2.utp.ac.pa/bitstream/handle/123456789/6473/tesis_jose_carlos_rangel_ortiz.pdf?sequence=1&isAllowed=y