Semantic Mapping and Object Detection for Indoor Mobile Robots

Semantic Mapping and Object Detection for Indoor Mobile Robots

S. Kowalewski, A.L. Maurin, J. C. Andersen
Department of Electrical Engineering, Danmarks Tekniske Universitet, Kgs. Lyngby, Denmark

Abstract. In this paper the authors present a full solution for object-level semantic perception of the environment by indoor mobile robots. The proposed solution not only provides means for semantic mapping but also division of the environment into clusters representing singular object instances. The robot is provided with information that not only allows it to avoid collisions with obstacles present in the environment, but also information about the localization, the class and the shape of each encountered object instance. This level of perception enhances the robot's ability to interact with the environment. The state-of-the-art deep learning solution, Mask-RCNN, is used for the image segmentation task. The image processing network is combined with an RTAB-Map SLAM algorithm to generate semantic pointclouds of the environment. The final part of the paper is focused on pointcloud processing: providing methods for instance extraction and instance processing. To verify the performance of the proposed methodology multiple experiments are conducted. Through the evaluation of the results it is possible to identify possible improvements.

1. Introduction

The environmental awareness is a crucial feature of an autonomously operating robot. The information about surroundings is acquired by available sensors and then processed internally to make autonomous operation in an unknown environment possible. The most basic approach is to use sensors to detect occupied areas and mark them as obstacles. In such case, the robot is aware of the placement of obstacles but lacks any information about the type of the detected objects. This approach is sufficient when the objective is to maneuver and avoid collisions but doesn't provide any additional information, that might be useful when interacting with encountered objects.

The methodology proposed in the paper expands this basic approach by adding an obstacle classification step and instance detection step. In the result, not only the occupancy information is acquired but also the semantic classification for each detected object is determined. Such approach allows to not only navigate around objects but also interact with them depending on the obstacle class; for instance a chair might be moved, whereas a table should be maneuvered around.

The main contributions of this paper are as follows:

- •Design of an end to-end solution that provides an object-level classification and outputs class, position and shape of each of the detected objects.
- •Verification of the performance through experiments.

The rest of the paper is divided in following manner: in Section 2 related work is reviewed. Section 3 provides the reader with a general overview of the proposed solution and the division of the problem into segments. In sections 4,5 and 6 the methodology used in each of the segments is described. Section 7 includes the description of conducted experiments. The results of experiments are presented in Section 8. Sections 9 and 10 include conclusion and future work, respectively.

2. State of the art

There's been multiple attempts to provide solutions capable of generating 3D semantic maps - maps in which each area is semantically classified. The most common methodology combines a semantic image segmentation solution with a SLAM algorithm.

An example of this approach is SemanticFusion [1] which is a combination of, at the time, the state-of-the-art Deconvolutional Semantic Segmentation network [2] with an ElasticFusion SLAM [3] algorithm and a Bayesian update scheme. This approach allows authors to generate accurate semantic 3D maps of the environment. Nakajima et al. [4] take a similar approach but they assign class probabilities to regions instead of single voxels of the 3D map. This provides a significantly faster solution that can perform at 30Hz frequency. Jeong et al. [5] present a methodology based on the same approach but use multiple sensors to allow outdoor, large scale operation.

A slightly different approach to the segmentation problem is presented by the authors of PointNet [6]. PointNet is a deep neural network that provides end-to-end pointcloud processing capabilities, outputting a variety of results including: object classification, part segmentation and semantic segmentation. Ability to directly process pointclouds is believed to provide superior results. The presented solution is used for single frame processing and authors do not explore the idea of generating global maps.

Nonetheless, these works are focused on the generation of the semantic map and do not attempt the object-level instance division as proposed in this paper.

3. Overview

The proposed system was divided into four essential submodules:

Image acquisition - this module is responsible for the acquisition of RGB and depth images from the Kinect camera.

Image segmentation - this module is responsible for processing acquired images. The performance of this module has a great impact on the whole system as this module adds the semantic meaning to the acquired images. The output images include annotations about detected objects, their classes and localization in the image. Mask-RCNN [7] is used to achieve the module's functionality. For the purpose of the neural network training and performing experiments a GeForce GTX 1080 graphical unit was used.

Robot localization and mapping module - this module is responsible for keeping track of the robot position and simultaneously creating a semantic pointcloud of the environment. RTAB-Map [8] is used to achieve the module's functionality.

Instance detection and processing module - this module is responsible for splitting the created semantic pointcloud into separate object instances. The output of this module is the localization (in the global coordinate system), class and the shape (represented by a pointcloud) of each of detected objects. Moreover, this module is responsible for the instance processing task.

4. Image segmentation

To achieve the semantic image segmentation the Mask-RCNN deep neural network is used. The Mask-RCNN is a neural network architecture created in Facebook AI Research, it expands the capabilities of it's predecessor: Faster-RCNN [9], by adding an image segmentation output. The network processes the input image and provides localization of the classified objects together with segmentation masks.

The original RGB version of Mask-RCNN is modified to process RGB-D images. The reasoning behind the modification is that in many cases the objects seem to be hard to distinguish using only RGB images (for instance when colors of the object and background are similar). In these cases the depth image might provide the missing information necessary to detect object's boundaries and therefore enhance the performance of the network.

The neural network was trained using the SUN-RGBD dataset [10]. This dataset is a compilation of multiple different indoor datasets [11-13] and consists of 10.335 images. It contains 37 different object classes which represent objects that are expected to be found in an indoor environment. The dataset provides comprehensive labeling including all of annotations required for Mask-RCNN training.

The neural network was trained using a sub-set of available classes (chair, table, window, door, box, shelves, sofa, cabinet). The dataset was split into training and validation sets as provided in the dataset information file.

The pre-trained model used for transfer learning was a Mask-RCNN model trained on the the COCO dataset [14]. Multiple training sessions were conducted using RGB and RGB-D variations of Mask-RCNN. For the training sessions in which the Mask-RCNN enhanced with depth was used, the weights of the first convolutional layer of the network could not be transferred as the shape of the convolution was modified. In such cases, this layer was ignored during the transfer learning process.

The results and discussion can be found in Section 8.1.

5. Robot localization and mapping

To be able to map the environment, as the robot is exploring the environment, it is necessary to reliably localize the robot. Moreover, the semantic images generated by the image segmentation module have to be expanded into a pointcloud using the corresponding depth images. These actions are achieved by applying a simultaneous localization and mapping (SLAM) algorithm.

RTAB-Map algorithm was found to be suitable for the task. It is an RGB-D Graph-Based SLAM algorithm that ensures real-time operation. The robot localization and mapping module outputs a semantic 3D pointcloud of the environment together with a 2D costmap of inaccessible areas.

The original implementation of RTAB-Map algorithm had to be adjusted to be suitable for the solution proposed in this paper. A parallel graph was added that includes segmented images instead of the original RGB frames captured during the operation. This approach allows to keep the original operation of the RTAB-Map, which is based on the features extracted from camera frames, and at the same time generate a pointcloud based on semantic images.

6. Instance detection and processing

6.1. Instance detection

The generated semantic pointcloud provides substantial amount of information about the environment but lacks the division into single object instances. To achieve this task, the generated semantic pointcloud of the environment is processed by an euclidean cluster extraction algorithm. The parameters used for the cluster extraction can be found in Tab. 1.

To avoid clustering together points belonging to different classes the full semantic pointclout is divided into subsets corresponding to each of the classes. The euclidean cluster extraction is applied to each of the subsets.

Table 1. Euclidean cluster extraction parameters.

Parameter	Value
Search radius	10[cm]
Minimum size	50[pts]

6.2. Instance processing

The final part of the solution consists of a set of instance processing algorithms. Instance processing is a crucial part of the proposed solution, it illustrates the usefulness of the approach and possible capabilities of a robot with object-level perception of the environment. Two instance processing examples were implemented: chair orientation detection algorithm and approach pose estimation algorithm.

The chair orientation detection algorithm allows the robot to determine the orientation of each of the encountered chair objects. This is beneficial when interacting with chair objects as the robot is aware from which direction the object should be approached.

The approach pose estimation algorithm is used to, given the selected object instance, estimate the target position of the robot when navigating towards the object.

Chair orientation detection algorithm: The algorithm for the detection of the front of the chair is divided into two parts. Firstly, a plane parallel to the z axis is fitted into the instance pointcloud. To find the plane the RANSAC algorithm is used. The detected plane corresponds to the backseat of a chair. Obviously this part works only with the assumption that the chair has a distinguishable backseat. The plane provides two normal directions that represent the front and the back of the chair.

Once the backseat plane is detected it is necessary to determine which direction represents the front and which represents the back of the chair. This part of the algorithm works on the assumption that distance-wise, the front of the chair is longer than the back. To determine the orientation of the chair a simple algorithm was proposed: points with maximum distance from the plane are found on the both sides on the plane; the one with a larger distance is determined to be on the front side of the plane.

The detected front direction is then represented with an unit vector placed in the mean point of the instance pointcloud. To allow operations on the 2D obstacle map, the found vector is casted onto the ground plane.

Approach pose estimation algorithm: The approach pose estimation algorithm was introduced to add a capability of navigating the robot towards a selected object instance. To achieve this, the space around the object is sampled by checking in the general obstacle map if the location is accessible. The space is sampled either in ray-like or arc-like fashion. The operation of the algorithm is illustrated in Fig. 1.

7. Benchmarking methods and experiments

7.1. Guidebot

To be able to perform experiments in the real-life scenario and verify the performance of the system it was necessary to use a mobile robot. For this purpose a DTU robot - Guidebot was used. It is a differential drive wheeled mobile robot equipped with a range of sensors including: a Kinect camera (XBOX 360), a laser scanner (SICK LMS200-30106) and wheel encoders.

7.2. Image segmentation

Multiple training sessions were run to provide more insight into the optimal training procedure. To measure the performance of the image segmentation task the mean average precision (mAP@0.5) [15] is used. The results are presented in Section 8.1

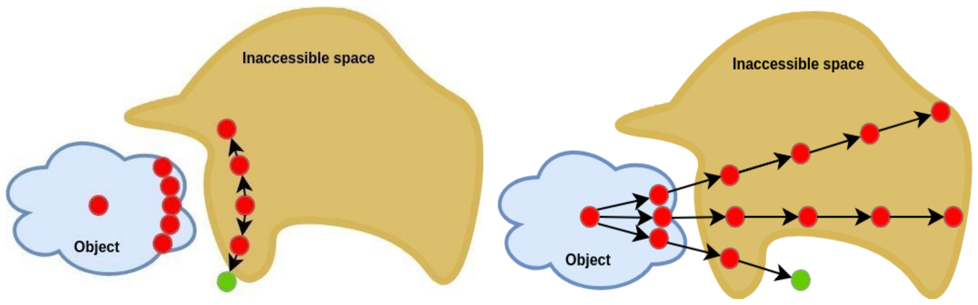

Figure 1. Approach pose estimation algorithm: on the left ray sampling; on the right arc sampling. Red dots represent unsuccessful sampling steps whereas green dots represent the final successful attempt.

7.3. Full solution

A final experiment was proposed to verify the performance of the whole solution in a real life scenario. The experiment consisted of two stages that are described in detail in this section. A single room was setup by placing different recognizable objects at known positions. On Figure 3a it can be seen that to make the task more challenging, a variation of objects was selected for each class (for example round and rectangular tables or different types of chairs). Tab. 2 contains information about objects used in the experiment and their placement. Additionally the last column provides information if the object was detected by the algorithm during the experiment.

During the experiment it was noticed that the robot detects background objects. The localization measurement was not conducted for these objects but for the sake of the completeness of the experiment they are present in the Tab. 2 with ”unknown” coordinates.

During the first stage of the experiment the robot was manually controlled. The goal was to map the robot’s environment and create a global semantic pointcloud. Then, the instance detection algorithm provided information about the position and the number of detected objects. These results allow to quantify the performance of the robot by verifying how many of the objects were detected and how precise the solution is.

For the detected objects, a mean localization error was calculated. The error was derived using the following formula, where **d** is the number of the detected objects, x_i^{gt} and y_i^{gt} are the coordinates of the i-th ground truth object, x_i^{pred} and y_i^{pred} are the coordinates of the corresponding predicted object.

$$e = \frac{\sum_{i=1}^{d}[abs(x_i^{gt} - x_i^{pred}) + abs(y_i^{gt} - y_i^{pred})]}{d} \tag{1}$$

Thus, before the start of the second stage of the experiment, the robot has already mapped the environment and is aware of detected objects. During the second stage commands were sent to the robot to approach multiple of the detected objects to see if the robot was able to maneuver in the environment and reach the selected objects.

Table 2. Placement of the objects during the final experiment.

Object	x[m]	y[m]	Detected?
Chair #1	1.25	1	Detected
Chair #2	2.10	1	Detected
Chair #3	3.10	1	Detected
Table #1	2.55	1	Detected
Table #2	0.4	1.2	Detected
Box #1	0.3	1.2	Detected
Box #2	4.45	-0.55	Detected
Box #3	unknown	unknown	Detected
Box #4	unknown	unknown	Detected
Box #5	unknown	unknown	Detected
Shelves #1	3	-0,85	Not detected
Shelves #2	unknown	unknown	Detected
Shelves #3	unknown	unknown	Detected

Table 3. mAP of the best model for each of the classes. It can be observed that classes with more training instances have performed better than classes underrespresented in the dataset.

Class	mAP	# training instances
Chair	0.631	11514
Table	0.53	6913
Window	0.464	2549
Door	0.415	2000
Cabinet	0.399	1464
Sofa	0.334	701
Box	0.229	1982
Shelves	0.203	482

Table 4. Detected objects and their placement. Green: true positives; Red: false positives; Blue: detection that were present in the environment but not considered during the setup of the experiment

Object	x[m]	y[m]	Notes
Chair #1	1.2	-0.95	Matched
Chair #2	3.18	1.13	Matched
Chair #3	1.94	1.21	Matched
Chair #4	2.57	-0.98	Missclassification
Table #1	2.36	0.94	Matched
Table #2	0.57	1.18	Matched
Table #3	3.17	-1.04	Missclassification
Table #4	5.13	1.05	Missclassification
Table #5	4.98	0.81	Missclassification
Window #1	5	1	Missclassification
Window #2	1.29	-1.46	Missclassification
Box #1	0.33	1.31	Matched
Box #2	4.87	0.4	Detected background object
Box #3	4.27	-0.3	Matched
Box #4	5.22	0.63	Detected background object
Box #5	4.95	1.81	Detected background object
Shelves #1	1.62	-1.47	Detected background object
Shelves #2	4.96	0.35	Detected background object

8. Results

8.1. Segmentation results
To find a model with an acceptable performance, the training sessions were conducted using both: the RGB architecture and the RGB-D architecture. The compilation of performed training sessions can be seen in Fig. 2.

The best performing model achieved the mAP of 0.414 (perfect mAP score is 1). The visual inspection of the results has proven a good performance of the network. The model was trained for 90 epochs (450000 iterations). SGD optimization was used with a learning rate of 0.001. The architecture of the best performing model was designed for RGB images as input. A more detailed analysis has been performed for the best performing model: Tab. 3 shows the performance of the model and number of training examples for each of the classes.

Unexpectedly, the Mask-RCNN with RGB-D inputs has achieved inferior results (maximum mAP 0.386). The reason for this might be the fact that the first convolutional layer was ignored during the transfer learning procedure. Moreover, for each hyperparameters setting only a single training session was conducted. Due to the stochastic nature of neural networks a single training result does not provide a definite answer about performance differences.

8.2. Full solution results
Tab. 4 provides information about objects detected during the experiment.

Firstly, it was observed that the final solution achieved a high recall value - detecting nearly all of the present objects. The recall value for the class, table and box classes reaches 1.0 what means that all of the ground truth objects were detected. For the shelf class the recall value drops to 0.67 due to the fact that one of the shelves was not detected.

The robot performs noticeably worse when the precision value is taken into account. This means that the final solution detects multiple objects that were not present in the environment. The robot detected 1 chair, 3 tables and 2 windows that were not present in the prepared room. It is worth noticing pointclouds for the majority of the miss-classified instances consist of small number of points. This means that the solution might be improved by changing the pointcloud acceptance threshold in the instance detector.

The calculated mean localization error value (see Formula 1) is 0.25[m]. It is expected that even better results could be achieved if the mapping process was extended to allow the robot to see the object from increased number of viewpoints.

For the second stage of the experiment, three detected object instances from the following classes were chosen: a chair, a table, a box. For each of the tested object the solution provided positions and orientations that would result in the robot facing the selected obstacle. The determined pose was visualized on the screen and was correct. Using a Robot Operating System (ROS) navigation package it was possible for the robot to avoid collisions and approach the selected object, given a desired end pose.

9. Conclusion

The main objective of the paper was to provide an semantic mapping and instance detection solution for indoor mobile robots.

The methodology has been designed, implemented and verified. The final experiment has shown that the robot equipped with the proposed system was able to map it's environment and assign semantic meaning to sections of the map. Moreover, the robot was able to divide the environment into single instances of objects belonging to various classes - keeping track of their localization and shape. Additionally, it has been proven that the acquired information about the shape of objects can be reliably used to determine orientation of objects. Finally, it was possible to use the acquired information to autonomously navigate the environment and approach objects.

To improve the results of the system, a detailed analysis of the performance of the image segmentation module has been conducted. A modification to the RGB Mask-RCNN implementation has been introduced by adding a depth channel to the input. No improvement of the performance was observed for the network with RGB-D inputs.

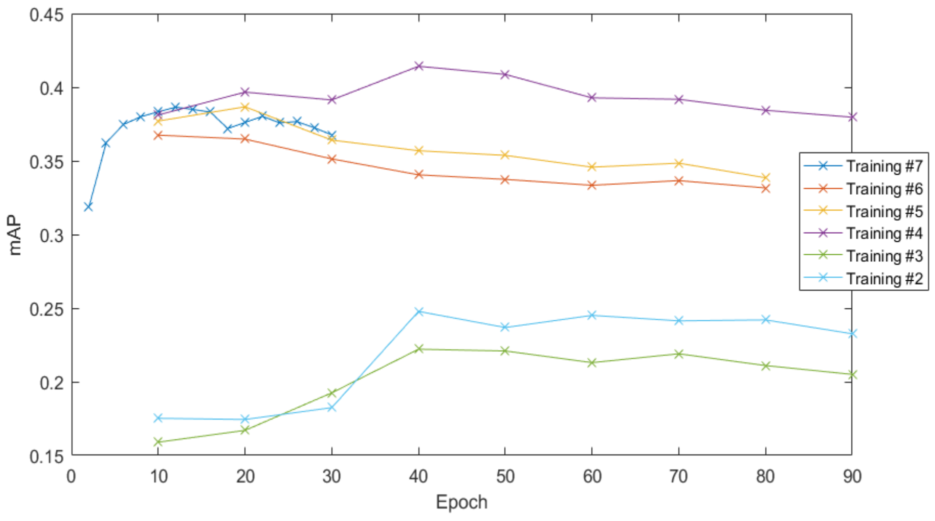

Figure 2. Performed training sessions. During the training sessions #2 and #3 the initial convolutional layers were frozen and therefore the performance of these models is inferior. After noticing that the models tend to overfit during long training procedures, session #7 was ran shorter with a finer performance sampling.

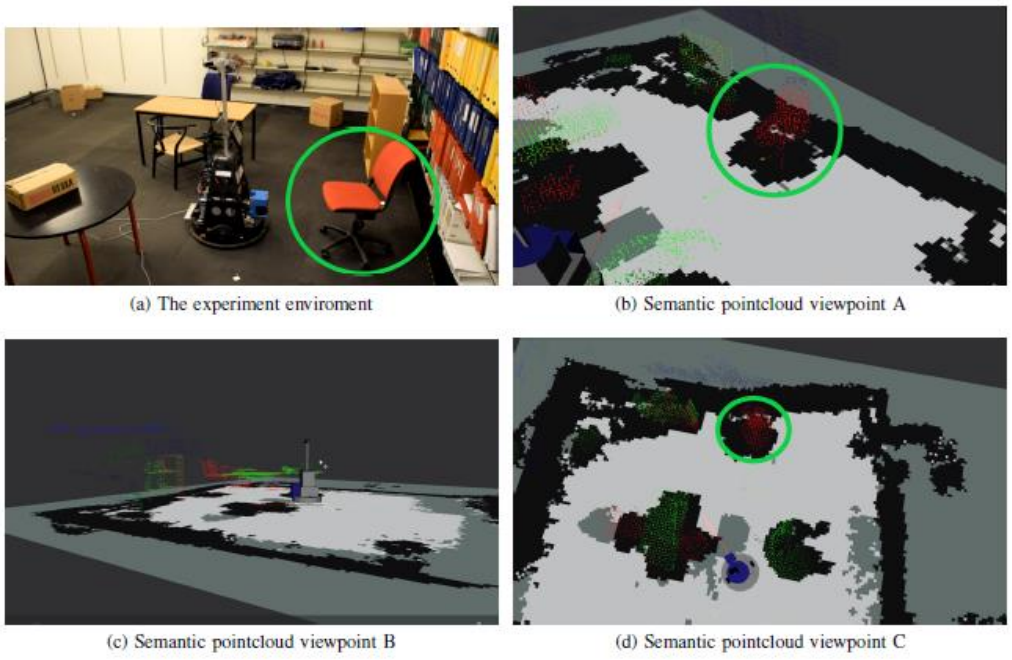
(a) The experiment enviroment
(b) Semantic pointcloud viewpoint A
(c) Semantic pointcloud viewpoint B
(d) Semantic pointcloud viewpoint C

Figure 3: The experiment environment and the generated semantic pointcloud seen from different viewpoints. Light green - chairclass; Dark green - box class; Red points - table class; Blue - shelves class. To make scene interpretation easier for the reader,one of the chairs is marked with a green circle

10. Future work

Although the results of experiments are satisfying, there are possibilities to further improve the performance of the solution. A more insightful Mask-RCNN training procedure should be conducted to provide a definite answer about the performance of the RGB and RGB-D Mask-RCNNs. Additional training sessions should be run for original and modified versions of the network to remove the impact of performance variance. Moreover, the performance of a neural network depends strongly on the number of available training examples. Throughout experiments it was proven that many of the classes available in the used dataset were underrepresented and hence the performance for these classes was inferior. A different dataset might be used or the existing dataset might be expanded to improve the quality of the image segmentation process.

The used SLAM algorithm (RTAB-Map) adds a significant limitation to the solution - the environment has to be static. In the future, it would be reasonable to replace the current SLAM algorithm with one that is suitable for operation in a dynamic environment. This would greatly expand capabilities of the solution.

Only a few scenarios for instance processing were presented. It is believed, that the paper provides a foundation to implement various instance processing algorithms - allowing the robot to approach and interact with different classes of objects. In the future, a set of behaviours should be implemented for each of the detectable object classes.

11. References

[1] Janoch, A., Karayev, S., Jia, Y., Barron, J. T., Fritz, M., Saenko, K. and Darrell, T. 2011. "*A category-level 3-D object dataset: Putting the Kinect to work*" in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*.

[2] Qi, C. R, Su, H., Mo, K.and Guibas, L. J. 2016. "*PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*" *CoRR,* vol. abs/1612.00593.

[3] Noh, H., Hong, S. and Han, B. 2015. "*Learning Deconvolution Network for Semantic Segmentation*" *CoRR,* vol. abs/1505.04366.

[4] Jeong, J., Yoon, T. and Park, J. B. 2018. "*Multimodal Sensor-Based Semantic 3D Mapping for a Large-Scale Environment*" *CoRR,* vol. abs/1802.10271.

[5] McCormac, J., Handa, A., Davison, A. J. and Leutenegger, S. 2016. "*SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural*" *CoRR,* vol. abs/1609.05130.

[6] Xiao, J., Owens, A. and Torralba, A. 2013. "*SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels*" in *2013 IEEE International Conference on Computer Vision*

[7] He, K., Gkioxari, G., Dollár, P. and Girshick, R. B. 2017. "*Mask R-CNN*" *CoRR,* vol. abs/1703.06870.

[8] Everingham, M., Gool, L., Williams, C. K., Winn, J. and Zisserman, A. 2010. "*The Pascal Visual Object Classes (VOC) Challenge*" *Int. J. Comput. Vision,* vol. 88, no. 2, pp. 303--338.

[9] Labbé, M. "*Real-Time Appearance-Based Mapping*" 2018. [Online]. Available: http://introlab.github.io/rtabmap/. [Accessed 21 06 2018].

[10] Matterport, "*Mask-RCNN model pre-trained on COCO dataset*" [Online]. Available: https://github.com/matterport/Mask_RCNN/releases/download/v2.0/mask_rcnn_coco.h5. [Accessed 21 06 2018].

[11] Silberman, N., Hoiem, D., Kohli, P. and Fergus, R 2012. "*Indoor Segmentation and Support Inference from RGBD Images*" in *ECCV*.

[12] Ren, S., He, K., Girshick, R. B. and Sun, J., 2015. "*Faster R-CNN Towards Real-Time Object Detection with Region Proposal Networks*" *CoRR,* vol. abs/1506.01497.

[13] Song, S., Lichtenberg, S. P. and Xiao, J. 2015. "*SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite*" in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[14] Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J. and Leutenegger, S. 2016. "*ElasticFusion: Real-time dense SLAM and light source estimation*" *The International Journal of Robotics Research,* vol. 35, no. 14, pp. 1697-1716.

[15] Nakajima, Y., Tateno, K., Tombari, F. and Saito, H. 2018. "*Fast and Accurate Semantic Mapping through Geometric-based Incremental Segmentation*" *CoRR,* vol. abs/1803.02784.