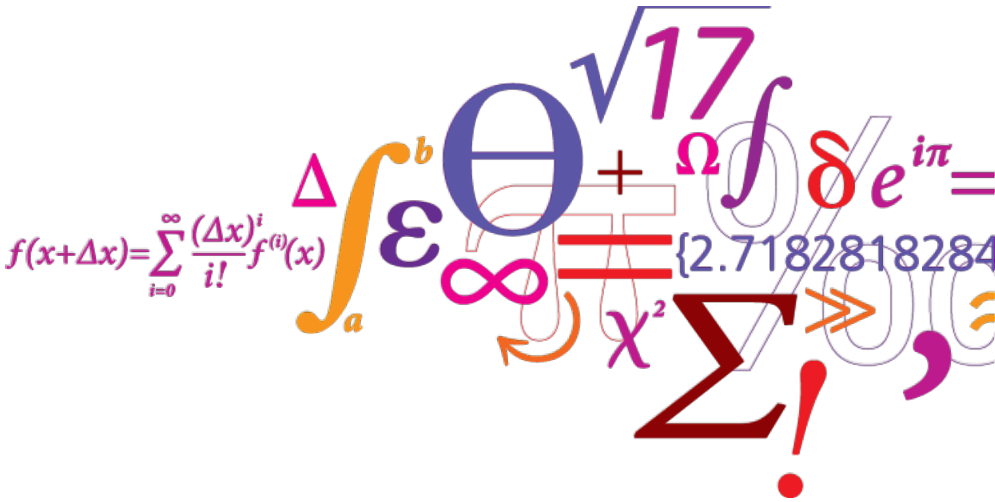
Obstacle detection and avoidance for indoor mobile robots

Author: Szymon Kowalewski
Supervisors: Adrian Llopart Maurin, Jens Christian Andersen



DTU Electrical Engineering
Department of Electrical Engineering

Introduction

Obstacle detection - basic approach:

- Robot knows which areas are occupied by obstacles
- Limited understanding of the environment

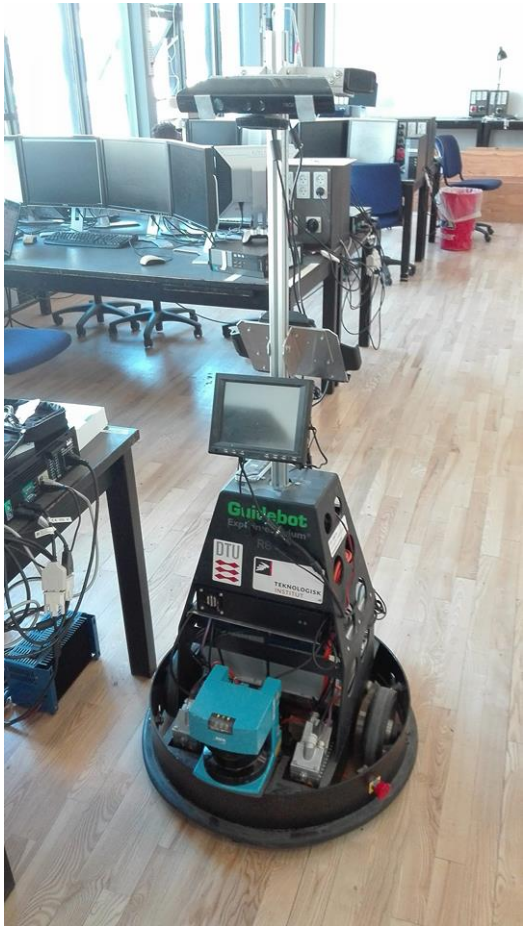
Obstacle detection enhanced with **obstacle classification** and **instance extraction**

- Robot knows which areas are occupied by obstacles
- Robot knows the location and the type of obstacles
- Robot knows the shape of obstacles

Introduction: Robot

Relevant sensors:

- Kinect camera
- Wheel encoders



Proposed solution

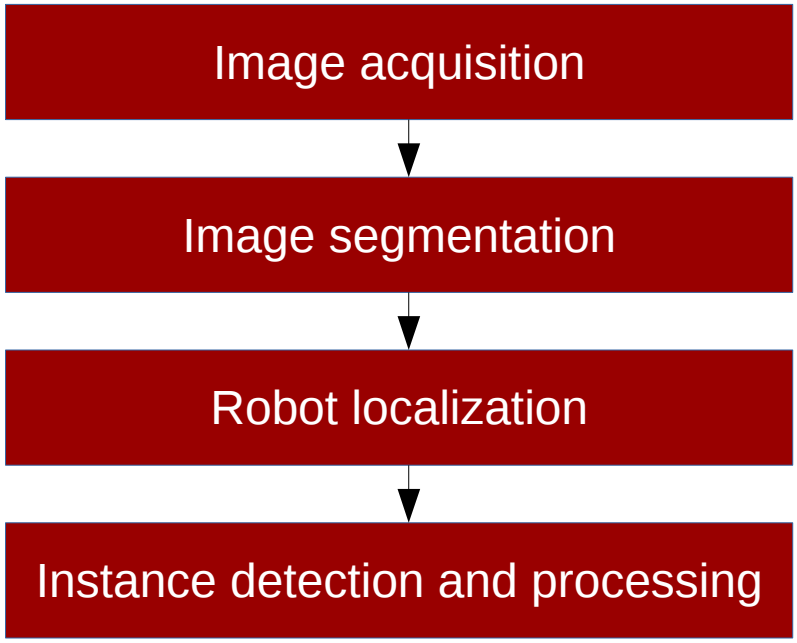


Image segmentation

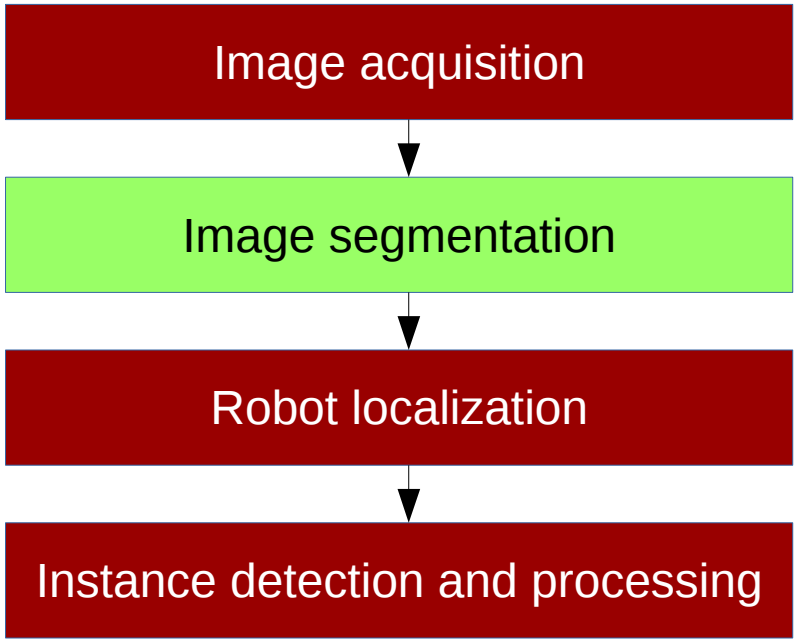
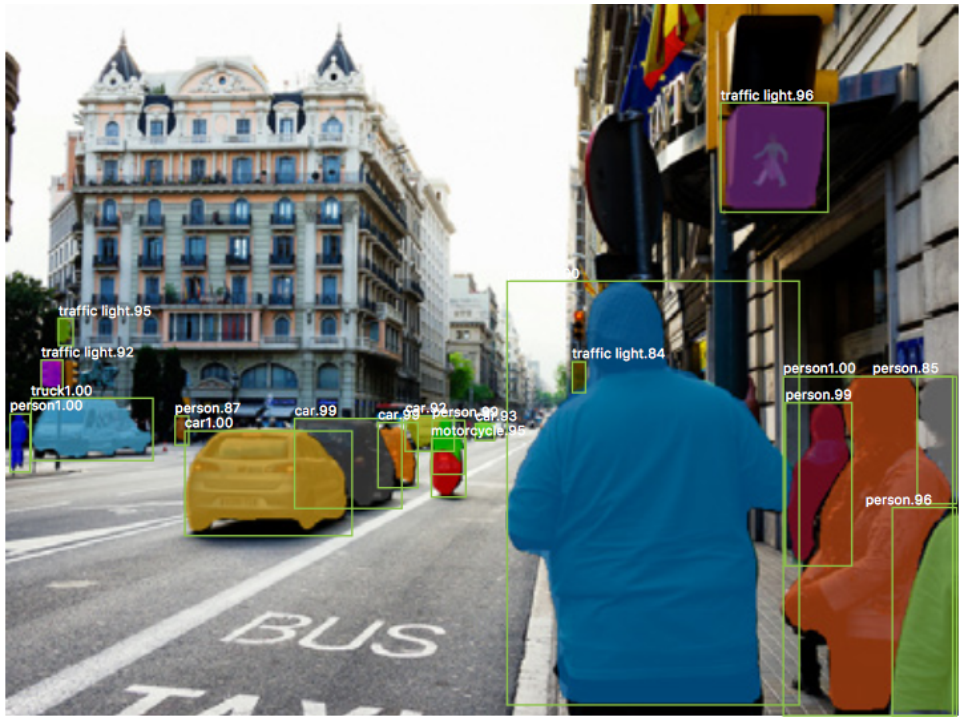


Image segmentation: Mask-RCNN



Source: https://github.com/matterport/Mask_RCNN

Created by researchers at Facebook AI
Input → RGB image
Output → bounding boxes, object classes, object masks
Achieved +- 10fps on kinect images

Image segmentation: Mask-RCNN

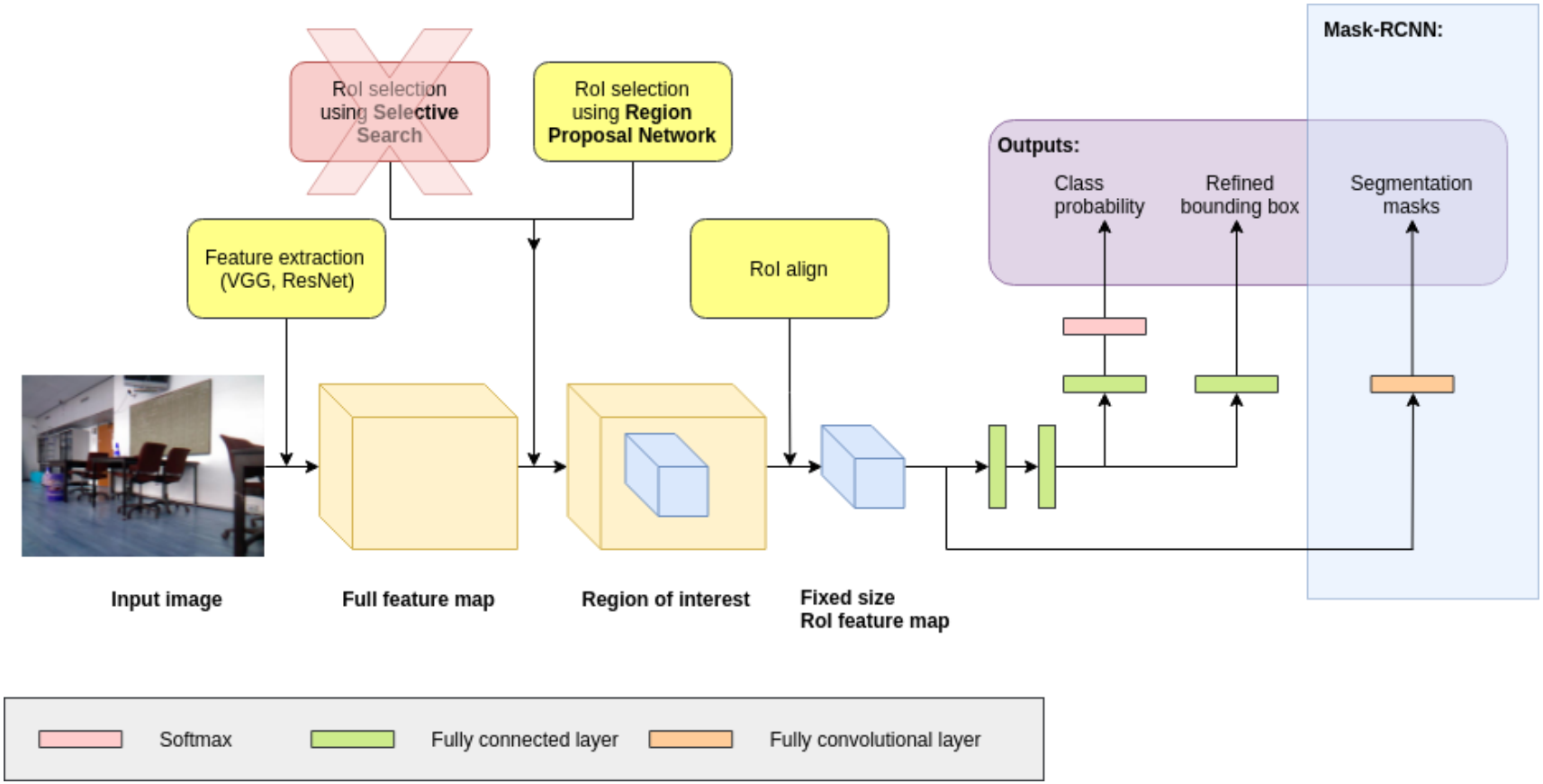


Image segmentation: dataset

Dataset requirements:

- Indoor RGB-D images
- Include required annotations
- Include “office” object classes

SUN-RGBD dataset:

- 10000 RGB-D indoor images
- 37 classes
- Densely annotated

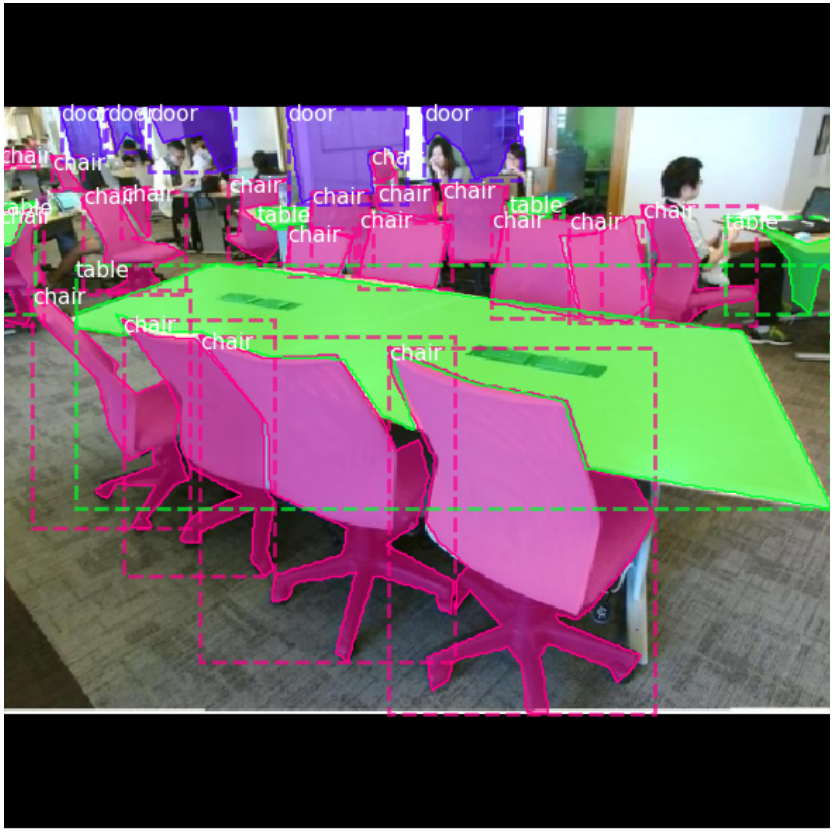
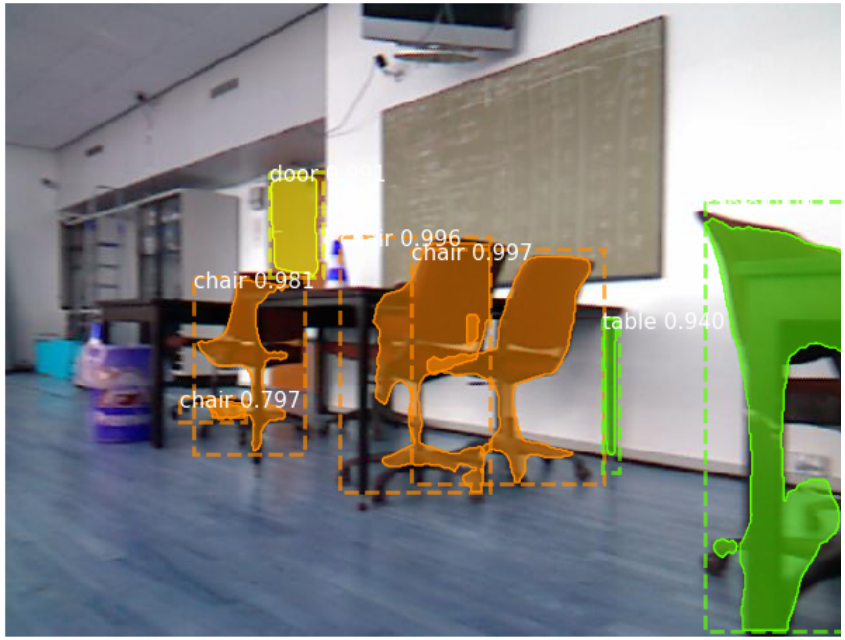
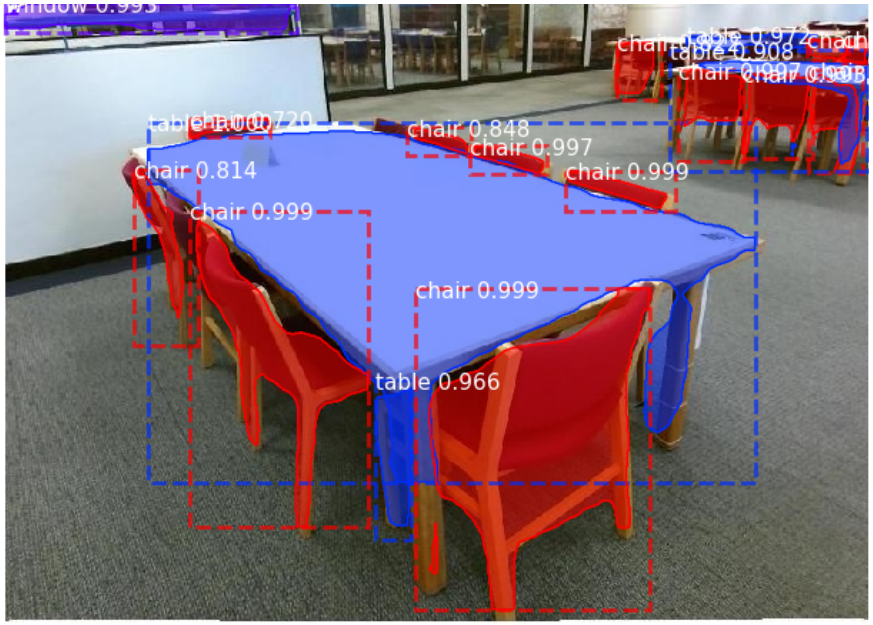
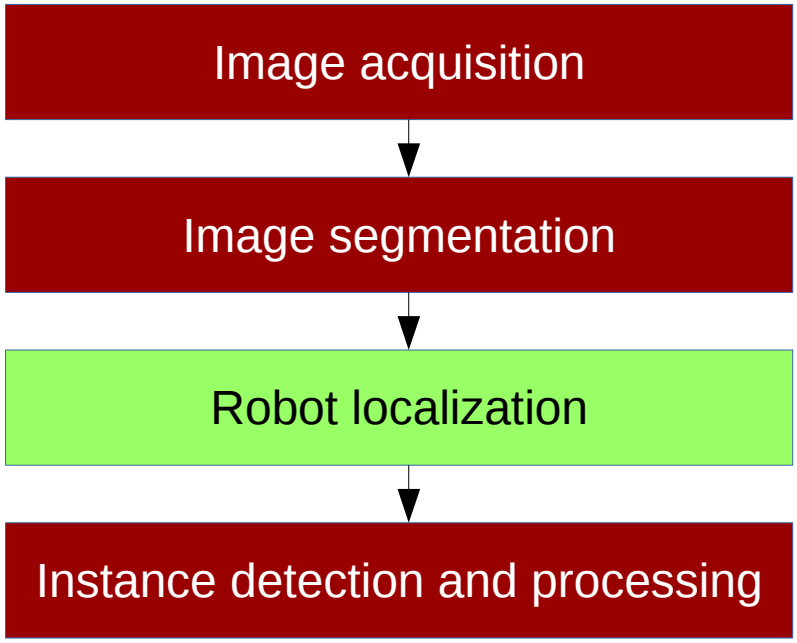


Image segmentation: results

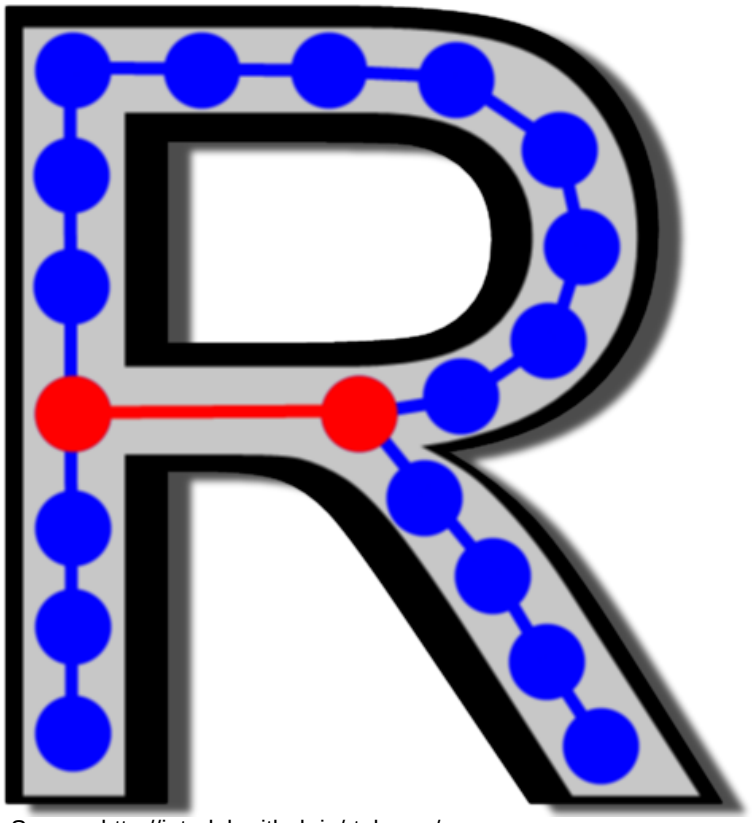




Robot localization



Robot localization: RTAB-Map

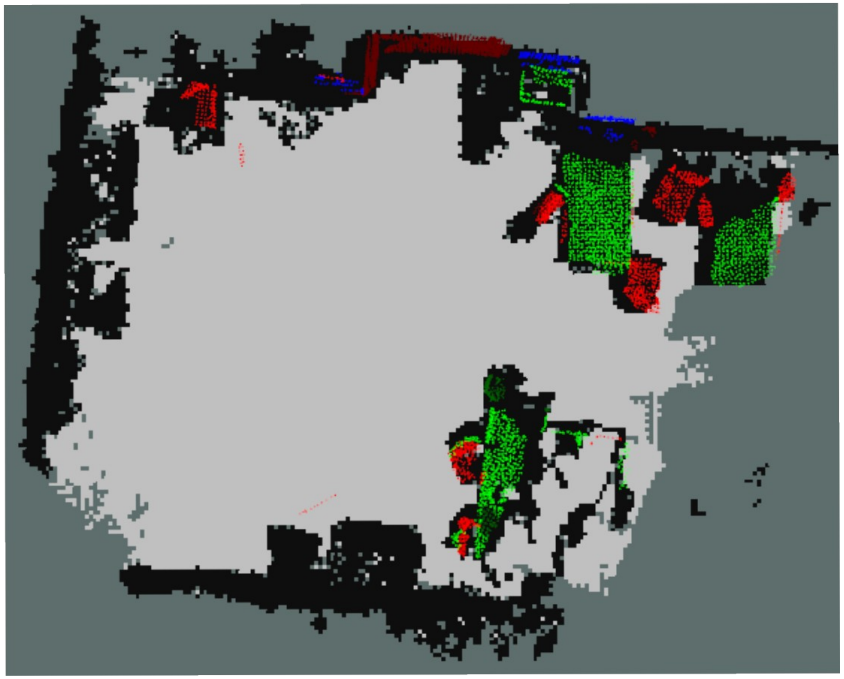


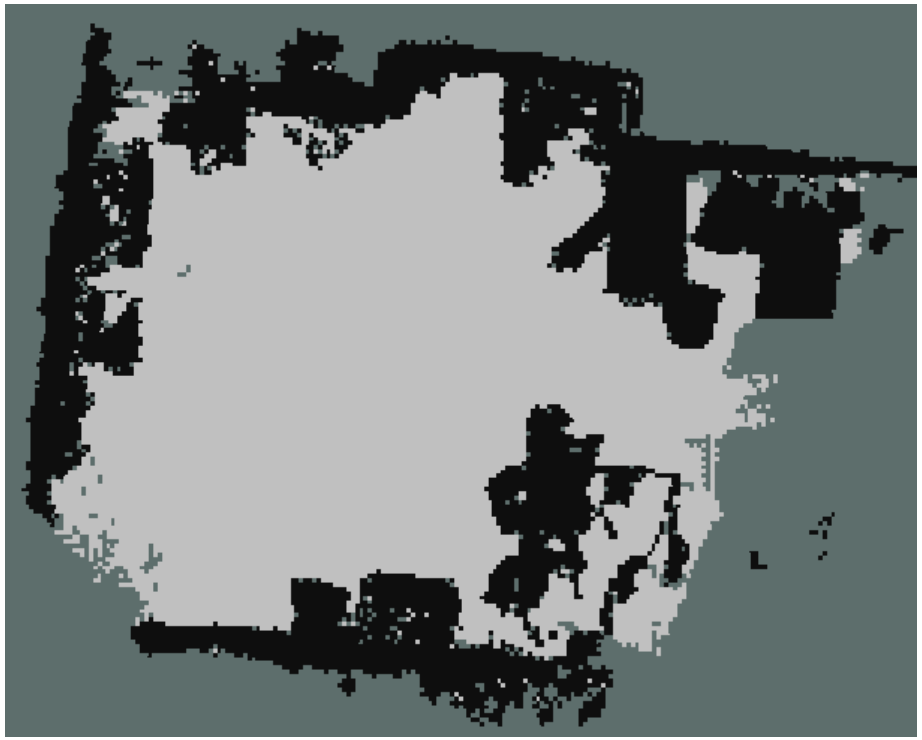
Source: http://introlab.github.io/rtabmap/

- Real-Time Appearance-Based Mapping
- RGB-D Graph-Based SLAM
- Loop closure detection
- Can be used with no laser rangefinder
- Part of the ROS – easy to use

Robot localization: outputs

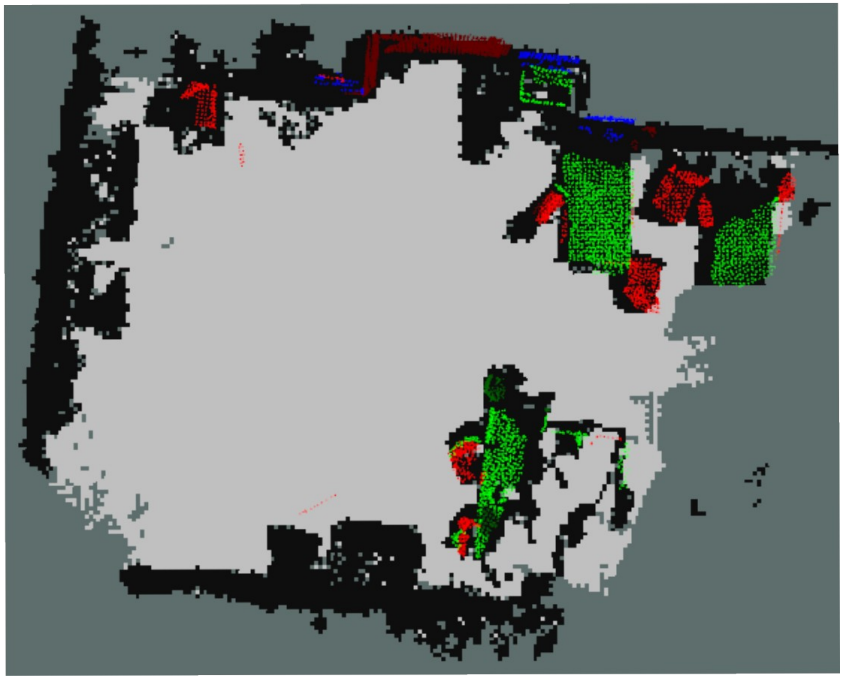
- Global obstacle costmap
- Global semantic pointcloud

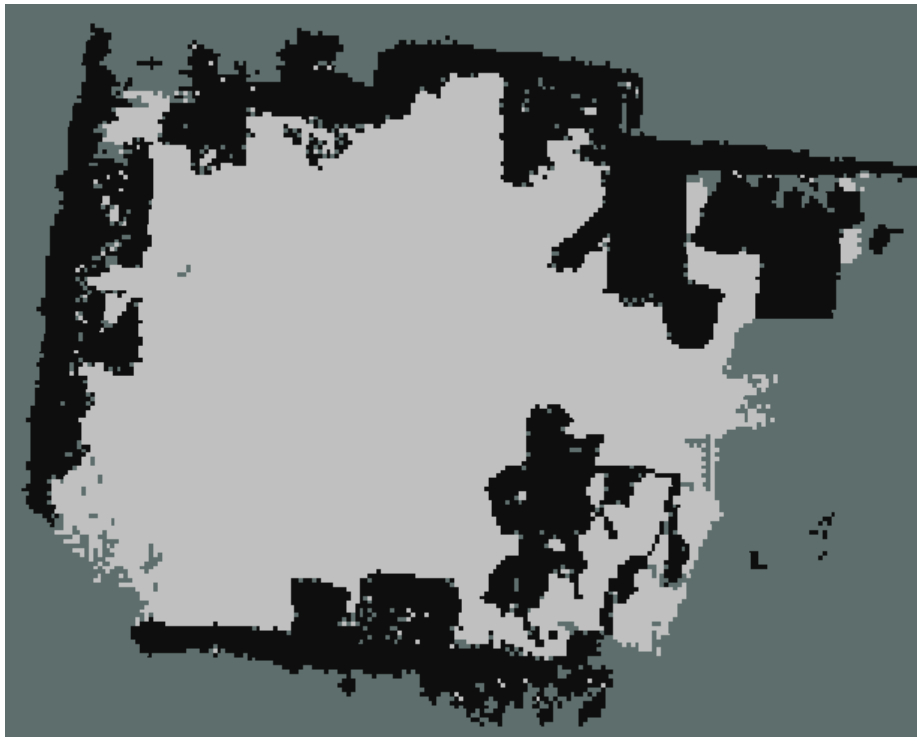




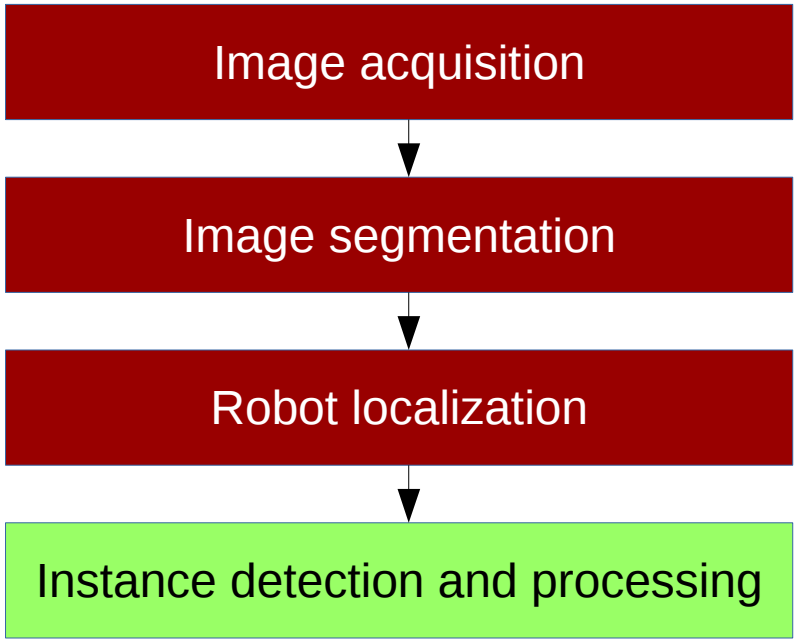
Robot localization: outputs

- Global obstacle costmap
- Global semantic pointcloud



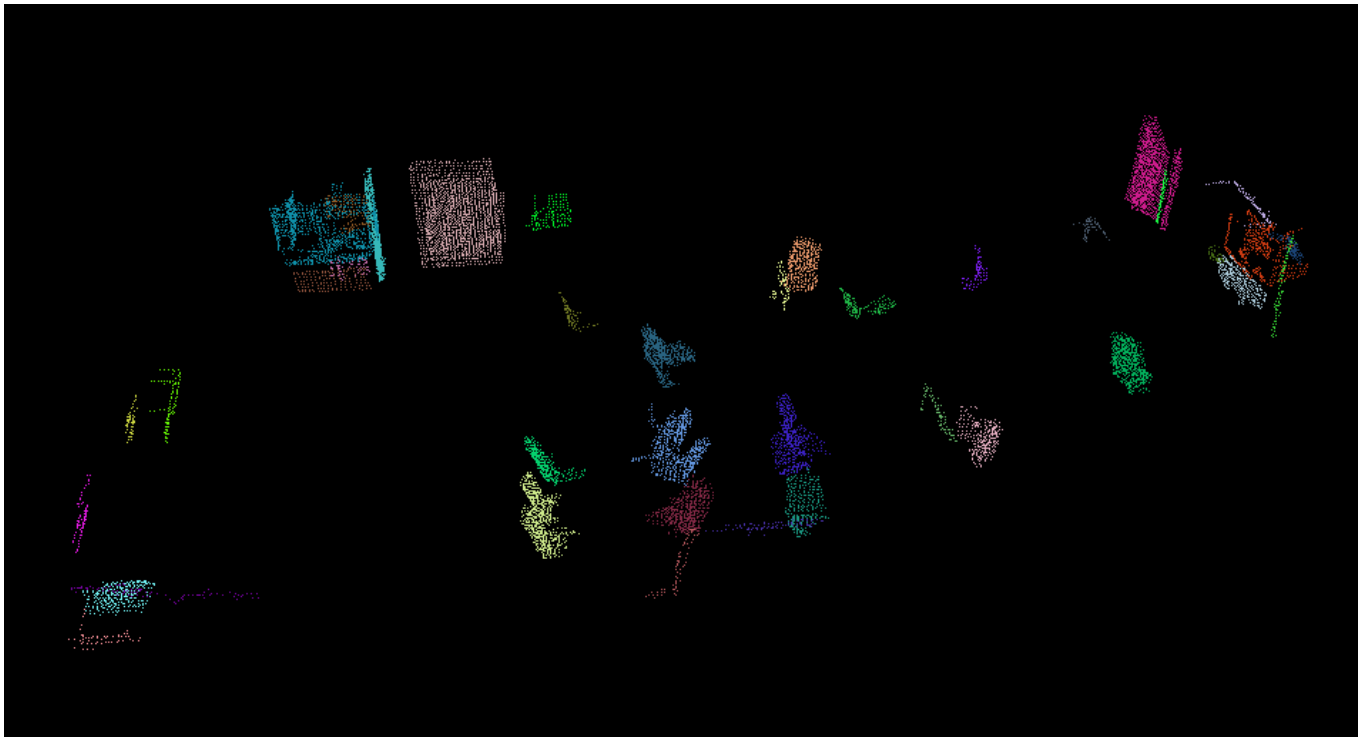


Instance detection and processing

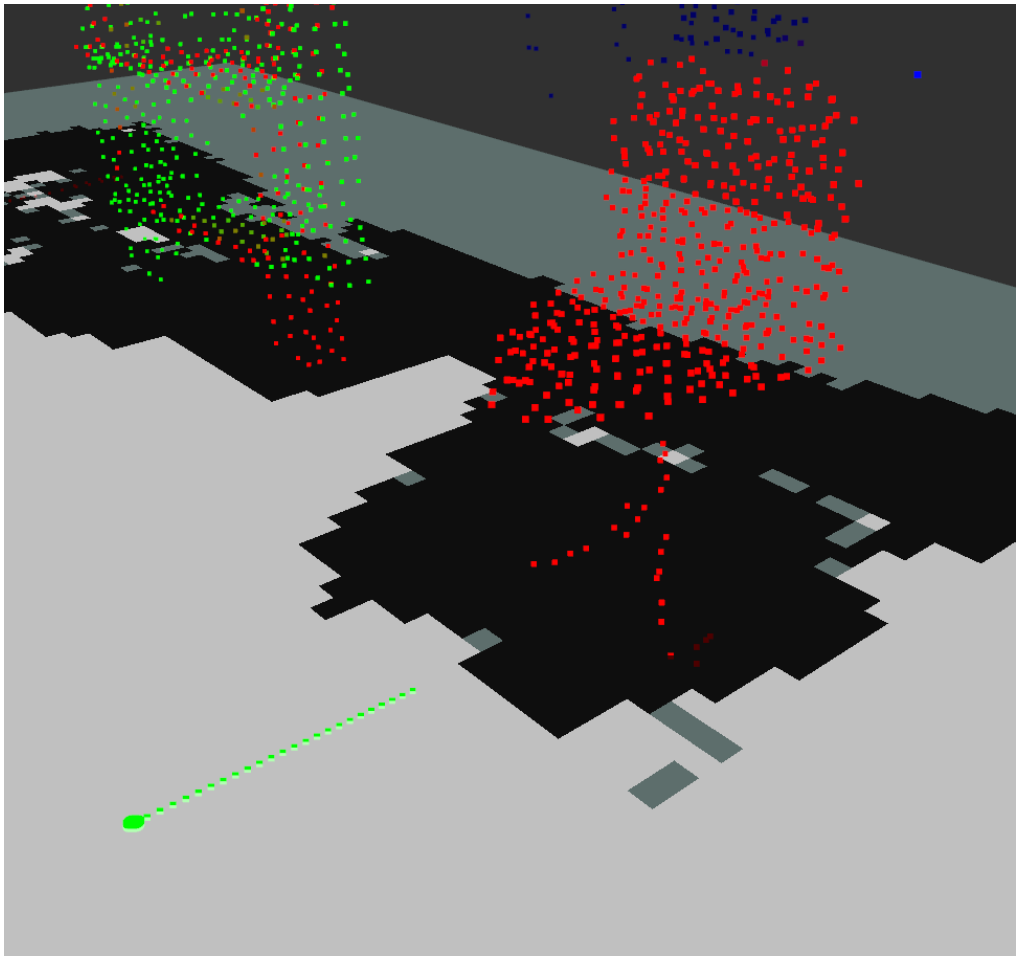


Instance detection: overview

- Detect single instances using pointcloud processing algorithm
- Save locations and class of each detected instance
- Keep information about the shape (pointcloud) of each instance



Instance processing: chair class



Final experiment: description

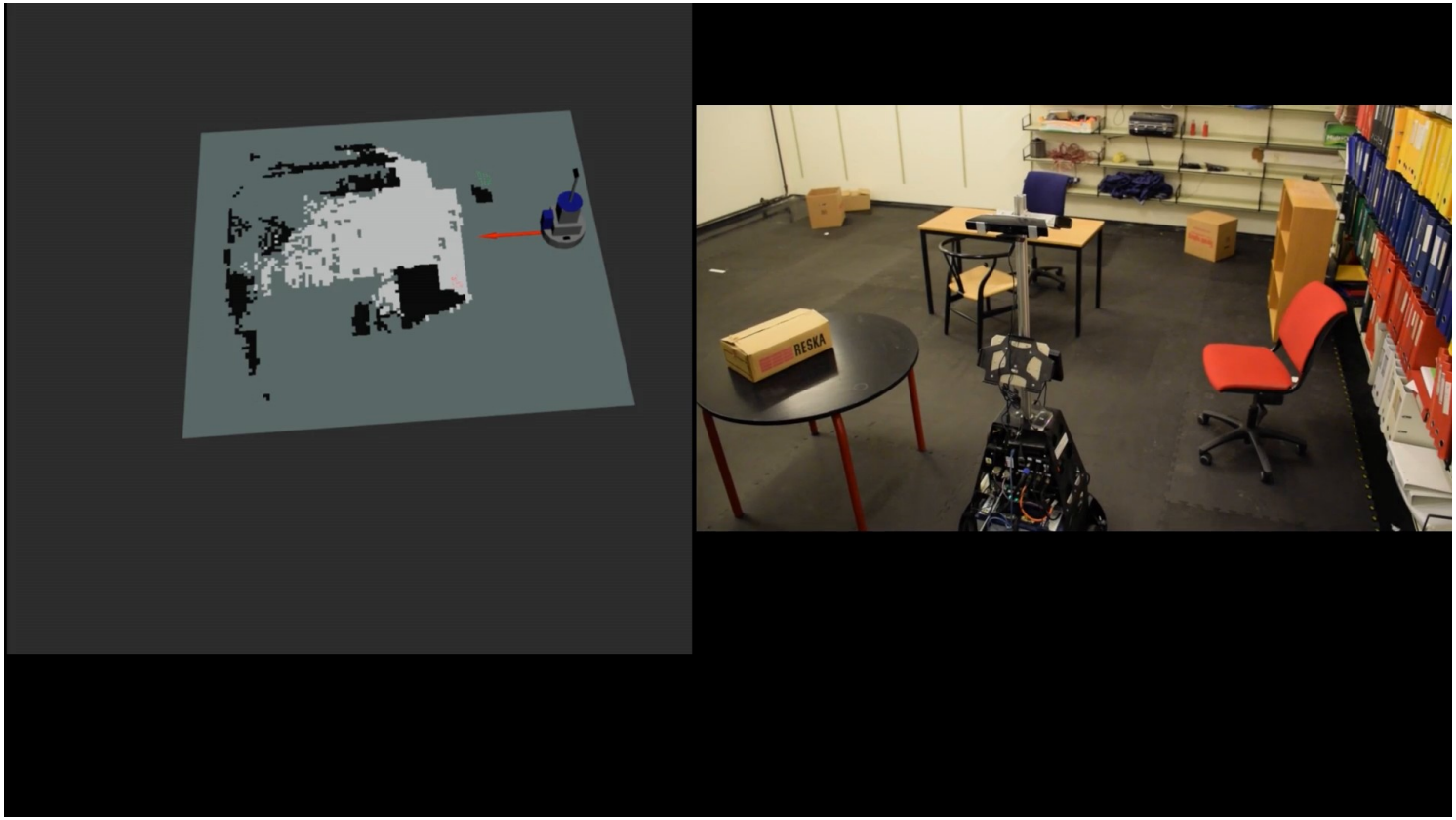
Stage 1:

- Robot controlled manually to map the environment
- Results of the obstacle detection compared with ground-truth

Stage 2:

- Robot was sent to approach multiple objects in the environment

Final experiment: video



Final experiment: results

Table 7.1: Placement of the objects during the final experiment.

Object	x[m]	y[m]	Detected?
Chair #1	1.25	-0.85	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 7.2: Detected objects and their placement.

Object	x[m]	y[m]	Number of points	Notes
Chair #1	1.2	-0.95	740	Matched
Chair #2	3.18	1.13	200	Matched
Chair #3	1.94	1.21	112	Matched
Chair #4	2.57	-0.98	87	Shelves missclassified as a chair
Table #1	2.36	0.94	541	Matched
Table #2	0.57	1.18	299	Matched
Table #3	3.17	-1.04	259	Shelves missclassified as a table
Table #4	5.13	1.05	85	Missclassification
Table #5	4.98	0.81	63	Missclassification
Window #1	5	1	317	Missclassification
Window #2	1.29	-1.46	102	Missclassification
Box #1	0.33	1.31	167	Matched
Box #2	4.87	0.4	128	Detected a background box
Box #3	4.27	-0.3	113	Matched
Box #4	5.22	0.63	82	Detected a background box
Box #5	4.95	1.81	52	Detected a background box
Shelves #1	1.62	-1.47	1409	Detected background shelves
Shelves #2	4.96	0.35	68	Detected background shelves

Future work

- Perform a more detailed hyperparameters tuning process for the neural network
- Prepare an extended dataset with more training examples
- Implement different instance processing algorithms for different classes
- Verify the performance with different SLAM algorithms (suitable for dynamic environment)

Thank you for your attention