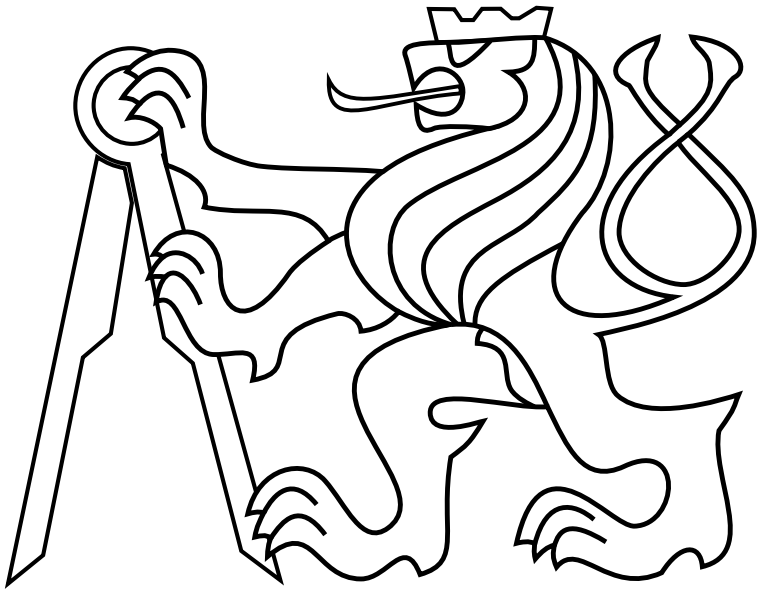
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

MASTER'S THESIS



Zdeněk Rozsypálek

Brick Detection for MBZIRC Competition

Department of Cybernetics

Thesis supervisor: **RNDr. Petr Štěpán Ph.D.**

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne ....
podpis autora práce

Author statement for undergraduate thesis

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date ....
signature

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Rozsypálek Zdeněk** Personal ID number: **457216**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Brick detection for MBZIRC competition

Master's thesis title in Czech:

Detekce cihel pro soutěž MBZIRC

Guidelines:

1) Study methods for analysis of Velodyne spatial depth data and study MBZIRC competition rules.
2) Analyze data from a Velodyne sensor placed on a ground robot and design an algorithm for detection a wall made up of blocks of predetermined sizes. The output of the algorithm should be a list of 3D cuboid positions relative to the robot position.
3) Design an algorithm that would create a map of the wall and allow to combine measurements from multiple robot positions.
4) Test the algorithm on real sensor data.

Bibliography / sources:

[1] Himmelsbach, Michael, et al. "LIDAR-based 3D object perception." Proceedings of 1st international workshop on cognition for technical systems. Vol. 1. 2008.
[2] Dou M., Guan L., Frahm JM., Fuchs H. (2013) Exploring High-Level Plane Primitives for Indoor 3D Reconstruction with a Hand-held RGB-D Camera. In: Park JI., Kim J. (eds) Computer Vision - ACCV 2012 Workshops. ACCV 2012. Lecture Notes in Computer Science, vol 7729. Springer, Berlin, Heidelberg
[3] Ma, L., Kerl, C., Stückler, J., & Cremers, D. (2016, May). CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM. In 2016 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1285-1291). IEEE.

Name and workplace of master's thesis supervisor:

RNDr. Petr Štěpán, Ph.D., Multi-robot Systems, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **14.01.2020** Deadline for master's thesis submission: **22.05.2020**

Assignment valid until:
by the end of summer semester 2020/2021

RNDr. Petr Štěpán, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

Acknowledgements

!!!OUTDATED!!! I would like to express my appreciation to Ing. Tomáš Petříček for his valuable and constructive suggestions during the planning and development of this thesis. I would also like to thank to the Department of Cybernetics of the Czech Technical University and to Michal Němec for the provided hardware. Finally, I wish to thank my family for support throughout my study.

Abstract

BLAH BLAH

Abstrakt

BLAH BLAH

Keywords: Keywords

Contents

List of Figures

Chapter 1

Introduction

Autonomous robotics experienced rapid development in last decades. There are broad ranges of applications from heavy work in industry to autonomous cars or space exploration. The two latter mentioned applications are very challenging, especially because they require many abilities which are specific for the area of mobile robotics. Unlike in industry where are robots usually performing repetitive tasks in unchanging environment, the mobile robot must have some awareness about its surrounding. To enable the robot to perform complex tasks it is usually equipped with various sensors which provides information about the environment. Machine perception is a subfield of autonomous robotics which is dedicated to interpreting the output data of these sensors.

The thesis is focused mainly on processing the data from a lidar (Light Detection And Ranging) sensor. The lidar is a laser based rangefinder which uses passive reflection of detected object. It has wide spectrum of usage and it is also very popular in autonomous robotics. For this application is frequently used spinning version of the lidar which can, thanks to its wide field of view, cover 360° around the robot. In recent years companies like Ouster or Velodyne successfully reduced cost and improved the quality of this technology which led to further increasing interest. The classical applications includes collision avoidance [1], SLAM [2] or detection [3].

The goal of this thesis is to present a detection algorithm for the MBZIRC 2020 (Mohammed Bin Zayed International robotic challenge). The contest aims on development of autonomous robotics and presents simplified problems which must be resolved before the robots can be applied in the real world. All of the participants are further motivated by the prize money for winner.

1.1 MBZIRC Contest

The contest took place in March in Abu Dhabi. Whole competition consisted of three challenges and the grand challenge which connected all challenges together. The first challenge was the only one which was focused solely on UAVs. The goal of the first challenge was to pop multiple big color balloons and to catch small ball carried by the organizer's drone. The other two challenges were designed for both UGVs and UAVs. The second challenge was about building a wall using the robots. Multiple polystyrene bricks were placed in the arena and the robots should have moved these bricks to the destination area and stack them on top of each other to build the wall. Lastly the third challenge was to extinguish fire on the surface of the building model. This task was motivated by inability of firefighters to extinguish fire inside high-rise buildings in UAE. UAVs and UGVs carried tanks full of water and squirt it into the fire dummy. Every team had three rehersals before the contest and then two competition attempts for each one of the three challenges. Only the best teams from all three challenges were nominated into the grand challenge which was limited to just one attempt.

1.1.1 Second Challenge

This thesis is focused on the second challenge, more specifically on the ground robot section of the second challenge, so that we provide more detailed description of this challenge. Each team is given thirty minutes to explore the arena (40×60 meters), localize all interest areas and build the wall. There are four types of the bricks with different colors. All bricks must be very light to enable the UAVs to pick them up. The dimension and colors of the bricks can be seen in the figure 1.1. Team obtains points for every placed brick. Bricks with different colors are rewarded by different number of points. Placing bigger bricks means higher rewards. In addition the UAV bricks are rewarded by twice as many points as the UGV bricks.

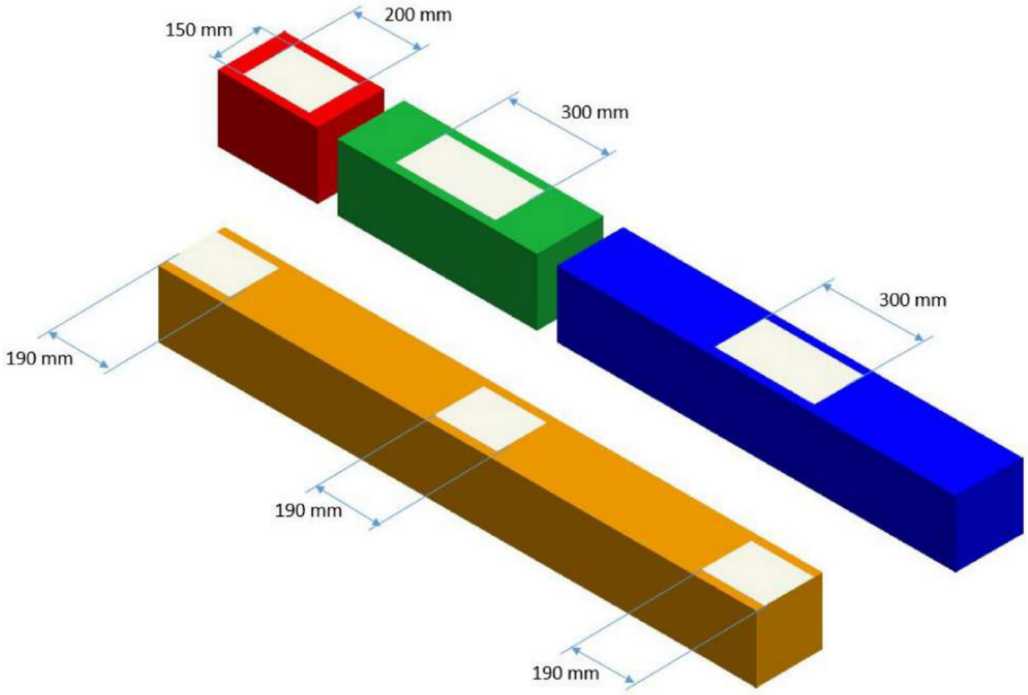


Figure 1.1: Colors and dimensions of the bricks provided by the organizer.

Each brick has thin metal plate on top of it, so that the robots are able to pick them up using electromagnets. At the beginning of the challenge all bricks are placed in beforehand unknown position. Initial position of bricks is unknown but there is a predefined pattern in which are the bricks put together. There are different patterns for the UGV piles of bricks and the UAV piles. The UGV bricks are stacked into the multiple height levels whereas the UAV bricks are stacked into the width and all are put on the ground. Due to the low weight of the bricks it is necessary to put UAV bricks into the rails, otherwise the bricks could be easily blown away by the propellers of the drones. Since the UAV bricks are all on the ground level (in purely horizontal pattern), it is much easier to detect them with the UAV bottom camera than using the lidar. That is why we are further concerned only about UGV bricks. These bricks are stacked in the positions displayed in the figure 1.2.

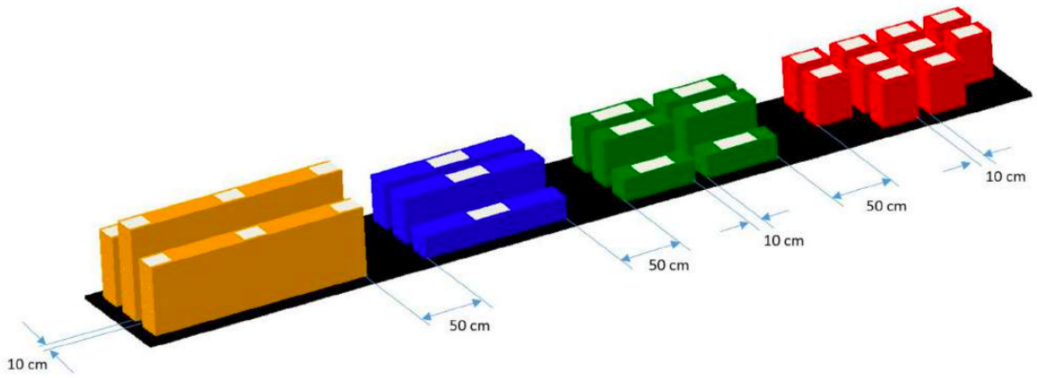
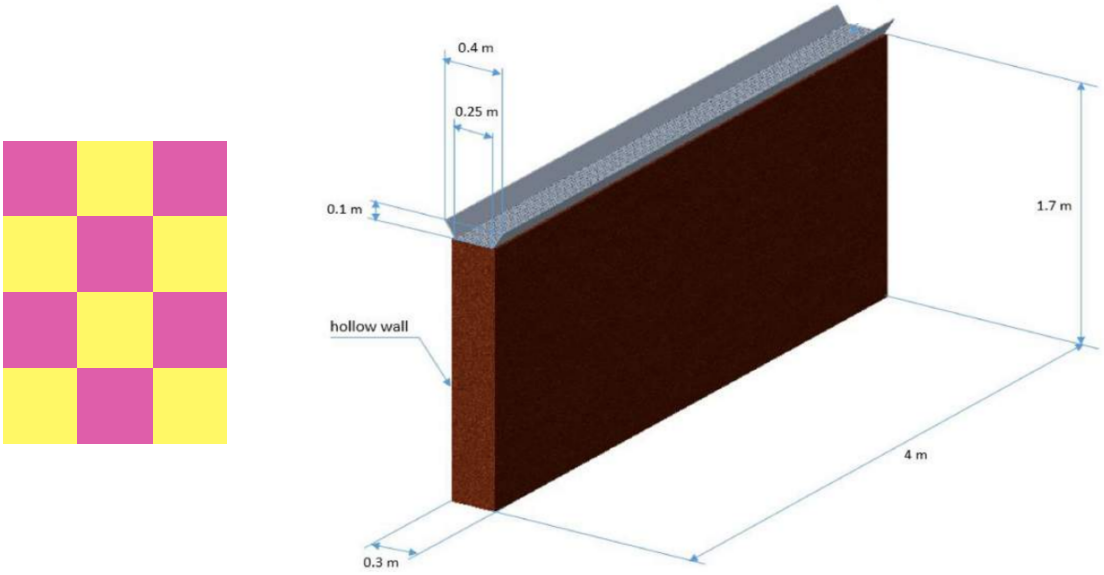


Figure 1.2: The positions of the bricks at the beginning of the second challenge.

Another objects of interest are destinations where the bricks should be placed. The robots must look for them too during the exploration. UGV bricks destination is marked by checker pattern. Detecting the pattern was very challenging because exact shape was not known till second rehersal. The final form of the pattern is shown in the figure 1.3a. Although we are not concerned about the UAV bricks, the destination of the UAV bricks is a vertical object, so it is much easier to detect it from the ground using the lidar. The UAV bricks destination is basically a wall as can be seen in the picture 1.3b. Bricks should be placed on top of this wall. The metal plate on top of each brick shifts the center of mass to the top and make the brick very susceptible to rolling. That is why are the auxiliary handles mounted on the top of UAV destination wall. At the beginning of the challenge is each team given the instructions which describe how the wall should look like at the end. When the built wall does not fit the instructions, the team gets penalty and gains less points for inaccurately placed bricks.



(a) UGV destination pattern.

(b) UAV destination

Figure 1.3: Description of target places for UAVs and UGVs. Each square in the UGV pattern has 10cm. Pattern consists of two 4×0.4 meter segments which are connected into the L shape. Whole UAV destination consists of five similar segments arranged into the M shape with right angles.

1.2 Equipment

For the sake of completeness is it necessary to describe what exact equipment was available. We used **Clearpath Husky A200** which is wheeled robot designed for outside robotics. The robot was equipped with many additional devices. As a computer, which is running all code and controlling the robot ,was used **Intel NUC**. To manipulate the bricks was the **Kinova robotic arm** mounted on top of the Husky robot. Two **12V electromagnets** are attached to the end-effector to enable the arm to grip the bricks. It would be very hard to grip the bricks without any feedback loop to the hand. For visual servoing and proper gripping we placed **Intel Realsense** camera close to the end of the arm. It is also possible to obtain feedback from electromagnets thanks to hall effect sensors and decide whether the brick is gripped correctly. For the localization, collision avoidance and detection was used **Velodyne VLP-16** lidar sensor. Lastly for moving the bricks around the arena we created a handmade cargo area which can contain up to six bricks and attached it to the rear bumper. It was not possible to carry more bricks mainly because of restrictions on robot's size and also due to the limited range of Kinova arm. Whole setup is captured in the figure 1.4.

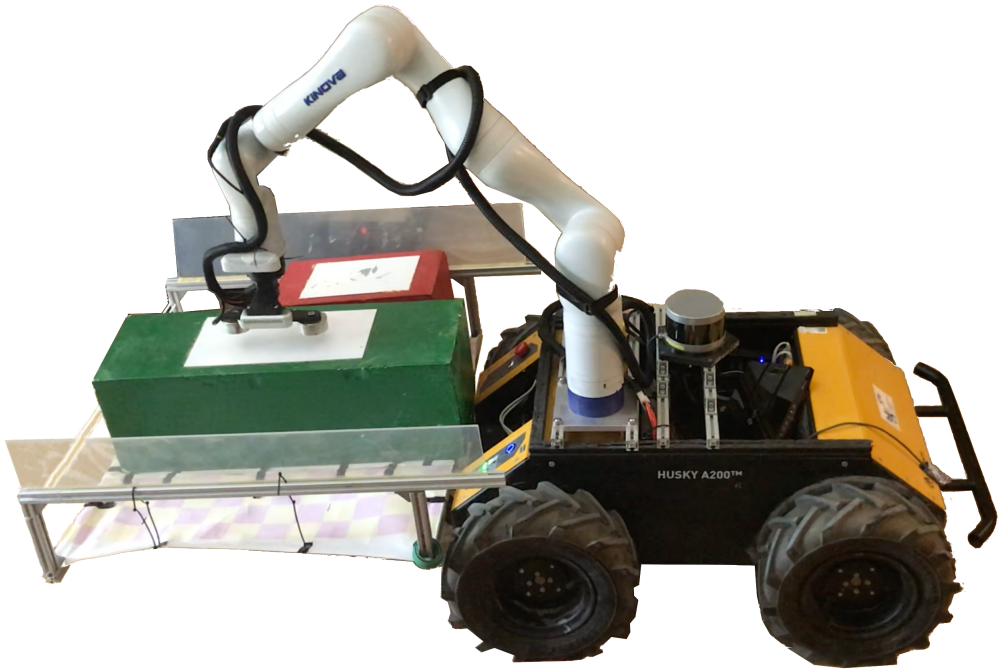


Figure 1.4: Clearpath Husky A200 adjusted for the second challenge.

1.2.1 Velodyne VLP-16

This thesis deals mainly with lidar data therefore following subsection will provide more detailed description of lidar sensor. Inside the VLP-16 puck is rotating infrared laser of class one which measures the distance using the time of flight principle. Lidar is powered by 12V power supply and the data are transferred via UDP packets over the ethernet. Parameters of the Velodyne lidar are listed in the table 1.1.

Layers	16
Range (m)	100
Vertical FOV (°)	±20
Vertical resolution (°)	2
Horizontal FOV (°)	360
Horizontal resolution (°)	0.1
Frequency (Hz)	5
Precision (m)	±0.03

Table 1.1: Parameters of VLP-16 lidar sensor.

1.3 Software

Operation system of the Intel NUC is Ubuntu 18.04. All important subroutines for controlling the robot are run by Robot Operating System (ROS) [4]. For localization is used particle filter from ROS AMCL package. Particle filter is Monte Carlo method for localization on a known map and it is particularly useful for sensor fusion. In our case is AMCL fusing odometry data and lidar measurements to estimate the robot position in the map.

Chapter 2

Methods

In this section of thesis are described methods which can be applied to detection of the bricks. Only brief description is provided and all methods are presented in general form.

2.1 Lidar detection range analysis

It is useful to know possible range of detection based on lidar sensor. This range influences the way how the robot explores the arena and also influences choice of used methods. Higher the detection range is, lower number of waypoints is necessary to explore whole arena. There is a limited time for exploration, because brick pickup and brick placement takes a lot of time. Speed of pickup and placement is limited mainly by the speed of Kinova arm. We estimate the maximal range using the figure 4.1.

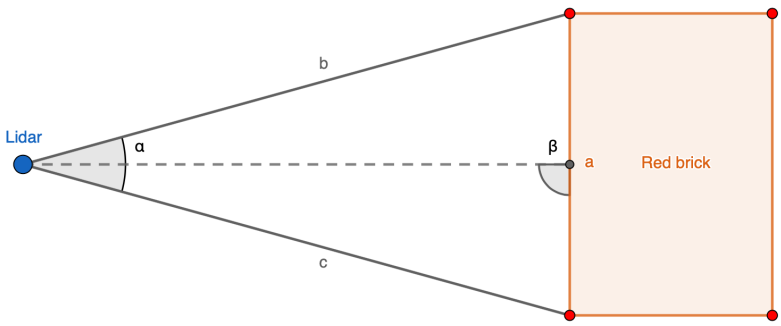


Figure 2.1: Visualization of rays hitting the red brick.

Angle α is the resolution of lidar known from table 1.1. Only the maximal range is calculated thus angle $\beta = 90°$. To obtain the distance between points on the brick the cosine theorem can be used.

$$a^2 = b^2 + c^2 - 2bc\cos\alpha. \tag{2.1}$$

Because β is right angle we can write $b = c$ and thus:

$$a = \sqrt{2b^2\left(1 - \cos\alpha\right)}. \tag{2.2}$$

Now we want to know how many rays N would hit the brick from given distance b with lidar angular resolution α and size of the brick a.

$$a = \sqrt{2b^2\left(1 - \cos\left(N\alpha\right)\right)} \tag{2.3}$$

$$N = \frac{\arccos\left(1 - \frac{a^2}{2b^2}\right)}{\alpha} \tag{2.4}$$

Finally we can plot a function of number of rays N with respect to distance to object b. This analysis can be done similarly for vertical and horizontal resolution.

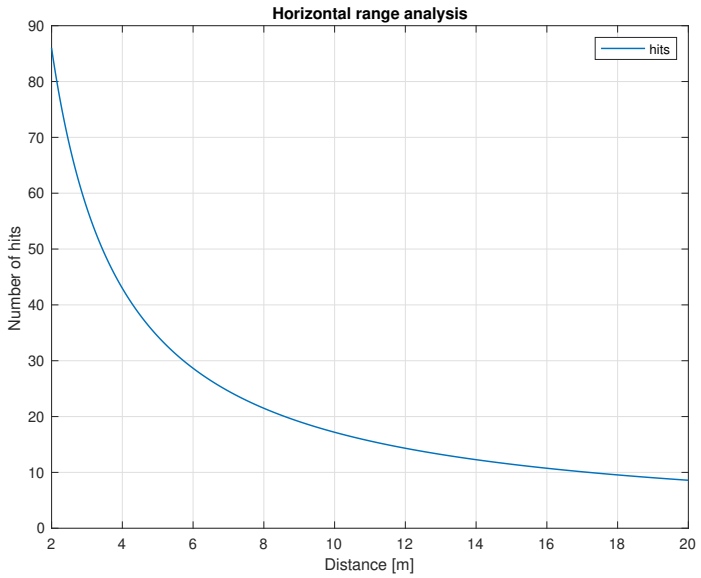


Figure 2.2: Number of hits of the smallest brick from given distance.

In the figure 2.2 is clearly visible that the horizontal resolution of the lidar is not limiting factor of the range. Even from 10 meters is lidar able to hit red brick more than 15 times. On the other hand the figure 2.3 shows that pile of bricks with height 40 cm would be hit by less than two lidar layers from distance bigger than 6 meters. Furthermore this is the best case scenario analysis where β is right angle which happens rarely in reality.

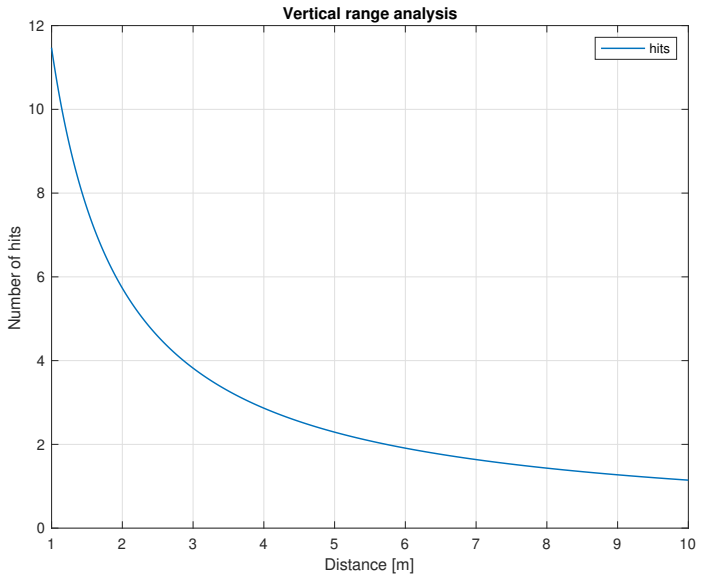


Figure 2.3: Number of hits of two stacked bricks from given distance.

2.2 Lidar data processing

To detect individual bricks it would be handy to extract straight lines from lidar data. Several methods can be used to achieve this goal. One of the most popular algorithms for lines extraction is currently split and merge algorithm. Initially was this algorithm proposed for image segmentation by Horowitz and Pavlidis [5]. Simple version of this algorithm for pointcloud processing is described in algorithm 1. There are many implementations of this algorithm which differ mainly in a way how they compute some particular steps of the algorithm. For example just the method of fitting a line to cluster can vary a lot. Very often is used the method of least squares, but as simple method as connecting endpoints of the cluster could be used. When the latter method is applied, algorithm is usually refered as Iterative End Point Fit (IEPF) [6]. For the cluster creation are the points iterated in each layer one by one. When the distance of subsequent points is too high we split the cluster. Every cluster is then further recursively split based on the most distant point from the fitted line. In a comparison to other line extraction algorithms is the split and merge algorithm one of the best performing in terms of precision and computational complexity [7].

```
Data: pointcloud
Result: line_segments
initialize constants C, S;
clusters = find_clusters(pointcloud, C);
while clusters is not empty do
    cluster = clusters.pop();
    line = fit_line(cluster);
    point = most_distant_point(cluster, line);
    if distance(point, line) > S then
        c1, c2 = split_cluster(cluster, point);
        clusters.push_back(c1, c2);
    else
        line_segments.push_back(cluster[start], cluster[end]);
    end
end
merge_parallel(line_segments);
```

Algorithm 1: Lidar data segmentation using split and merge algorithm. **C** is clustering distance and **S** is splitting distance.

2.3 EM algorithm

Expectation-maximization (EM) algorithm is iterative process which can find parameters of certain statistical model based on incomplete data. One of the most used statistical description for the EM algorithm is the Gaussian mixture model. This model is particularly useful because it emerges in many real world situations and it is easy to maximize. As the name of algorithm suggests it repeats expectation and maximization step. Each iteration of the algorithm should improve the likelihood of the model until the terminating criterion is met. The termination criterion can be simply number of iterations or the algorithm can be stopped when the model is not improving anymore. Although we are discussing mainly the Gaussian distribution, EM algorithm can be also used for other distributions from exponential family [8]. Usage of EM algorithm for classification of 3D lidar data is not completely novel approach. Gaussian mixtures was already used for terrain recognition [9].

2.3.1 Maximization

For maximization step is used maximal likelihood estimate weighted by γ from expectation step. For parameters of Gaussian distribution $\mathcal{N}(\mu, \sigma)$ and number of samples N looks maximization as follows:

$$\mu = \sum_{n=1}^{N} \gamma_n x_n \tag{2.5}$$

$$\sigma = \sum_{n=1}^{N} \gamma_n (x_n - \mu)^2. \tag{2.6}$$

Very important assumption for the convergence of the algorithm is that its likelihood with respect to estimated parameter must be concave. This can be easily proved by computing the second derivative of likelihood function. For example for mean value μ it is easy to show that second derivative of likelihood is always negative which means that the function has no local optima.

$$\mathcal{L} = \prod_{n=1}^{N} \mathcal{N}(x_n, \mu, \sigma) \tag{2.7}$$

$$\frac{\partial \log \mathcal{L}}{\partial \mu} = \frac{1}{\sigma^2} \sum_{n=1}^{N} (x_n - \mu) \tag{2.8}$$

$$\frac{\partial^2 \log \mathcal{L}}{\partial \mu^2} = \frac{-N}{\sigma^2}. \tag{2.9}$$

2.3.2 Expectation

The expectation step is done simply by evaluating conditional probability of current parameters given the sample:

$$\gamma_n = P(\mu, \sigma | x_n), \tag{2.10}$$

which is calculated using the Bayes theorem and the law of total probability:

$$\gamma_n = \frac{P(x_n|\mu, \sigma) P(\mu, \sigma)}{P(x_n)} \tag{2.11}$$

$$\gamma_n = \frac{P(x_n|\mu, \sigma) P(\mu, \sigma)}{\sum_{k=1}^{N} P(x_k|\mu, \sigma) P(\mu, \sigma)}. \tag{2.12}$$

There is only one class so the priors can be evaluated $P(\mu, \sigma) = 1$ and the equation simplifies to final form:

$$\gamma_n = \frac{\mathcal{N}(x_n, \mu, \sigma)}{\sum_{k=1}^{N} \mathcal{N}(x_k, \mu, \sigma)}. \tag{2.13}$$

2.3.3 Algorithm

How to implement general version of EM algorithm on sampled data is shown in algorithm 2. All important calculations for Gaussian distribution are described in previous subsections. It is not clear where to start iterating. It is possible to start both with the expectation and with the maximization step, but both parts are dependent on the result of

the other one. Here we start with the maximization step so during the initialization we set $\alpha_n = 1$. If some prior information about parameters of the model is available, they can be set during the initialization and the algorithm can be started with expectation step. This informed initialization can highly reduce the number of iterations and sometimes even an outcome of the algorithm.

Data: x
Result: parameters θ
set all $\alpha_n = 1$;
while *not stopping_criterion* **do**
    θ = maximization(x, α);
    α = expectation(x, θ);
end

Algorithm 2: Pseudocode shows how to implement the EM algorithm. **x** is the observed data.

2.4 RANSAC

Random sample consensus (RANSAC) is an iterative method which can estimate parameters of hypothesis given the data. It was first presented by Fischler [10] with application in scene and image analysis, but it can be used for fitting arbitrary hypothesis. The biggest advantage of this algorithm is its robustness to outliers. Major drawback of this method is very high time complexity when fitting hypothesis to noisy data with large number of samples. Whole iterative process is described in algorithm 3.

Data: x
Result: best parameters θ^*
initialize θ, θ^*, C, C^*;
while *not stopping_criterion* **do**
    samples = draw_samples(x);
    θ = find_parameters(samples);
    C = compute_cost(x, θ);
    if $C > C^*$ **then**
        $C^* = C$;
        $\theta^* = \theta$;
    end
end

Algorithm 3: Pseudocode shows how to implement the RANSAC algorithm. x is the observed data, C is the maximized cost and θ are the parameters of the hypothesis.

Number of drawn samples in **draw_samples** must be equal or higher than the number of degrees of freedom of the hypothesis. After drawing the samples method **find_parameters** assigns the correspondences between sampled data and the hypothesis. The correspondences are used to obtain the parameters of the hypothesis. Then is the algorithm evaluating quality of the hypothesis by applying the hypothesis to whole dataset. This can be done by arbitrary cost function. Common practice is to define some metrics in our domain and use a threshold value to obtain the number of samples which fits the hypothesis. These samples are often referred as inliers. Stopping criterion is usually met when the probability of sampling better hypothesis is lower than a specified threshold. This section describes just the basic version of the algorithm. Many improvements to RANSAC algorithm was proposed since 1981 [11].

2.4.1 Tentative Correspondences

The tentative correspondences can help us to choose better samples from the data to generate the better hypothesis. It is necessary to define some function witch measure the similarity between data and hypothesis. The data which has higher similarity to the hypothesis are then chosen with higher probability. It is also possible to completely ban correspondences with low similarity. Given the typical application in scene analysis is similarity usually computed by comparing keypoint descriptors.

2.5 Global model and transformations

One of the goals of this thesis is develop a global model which can efficiently store and update the positions of interest points. This global model is in the map coordinate frame. The localization of the robot is absolutely essential for precise global model. Every detection must be transformed from the coordinate frame of the sensor to coordinate frame of the map. Transformation can be easily done using matrix multiplication:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \vec{t}, \tag{2.14}$$

where R is 3×3 the rotation matrix and t is a 1×3 translation vector between coordinate frames. Similarly can be the transformation done in homogenous coordinates by merging translation and rotation into one matrix T:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} & R & & \vec{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{2.15}$$

Different framework for computing the transformations is using a quaternions. The quaternions are currently the standard for transformations in computer graphics and robotics. The biggest advantage of quaternions is that they are more efficient and does not suffer from gymbal lock and ambiguity of rotation. Arbitrary rotation and scaling can be expressed as quadruple of numbers in quaternion framework. The rotation between coordinate frames $B \rightarrow A$ is computed with quaternions as:

$$q_A = q_T q_B q_T^*, \tag{2.16}$$

where q_A is quaternion in coordinate frame A, q_B is quaternion in coordinate frame B, q_T is quaternion representing the transformation between these coordinate frames and q_T^* is its conjugate. There is available a library within the ROS which can handle all these transformations in different forms [12].

2.5.1 Symbolic map

When are all detections transformed into the map frame we can add them into a symbolic map. The symbolic map is storing the positions of all interest points and makes up the global model of the arena. Every object added to symbolic map has a float number which indicates confidence of detection. When there is a new detection within certain range from an object already stored in the symbolic map, new object is not added but only the confidence is increased. This approach creates a clusters of interest points of different types. All interest points can be polled from the symbolic map and robot can make decisions based on confidence of such an interest point. The symbolic map also checks whether the inserted object is located inside the arena otherwise is the object rejected. (CITACE)

2.5.2 Lidar to camera registration

As can be seen in the figure 1.1 (where the bricks are defined) besides the dimensions another important feature of the bricks is their color. Although, the color manifests itself little bit in reflectivity of the surface which can be detected by lidar, it is not possible reliably distinguish the colors using only the lidar sensor. Until there is a gap between the individual bricks, it is possible to detect brick using just the spatial data. Ideally the robot should be stacking bricks next to each other without any significant gap. Without any information about the color is then impossible to decide whether is the robot detecting one large brick or several small bricks put together. So for this part of detection is necessary to color the pointcloud. This can be done by using the image from Intel RealSense camera and projecting 3D lidar pointcloud to the camera plane. For this purpose is used the pinhole camera model. To describe such a model is used intrinsic camera matrix K which consists

of intristic camera parameters [13]

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.17}$$

where f_x, f_y are focal lengths and c_x, c_y stands for optical center of the camera.

Firstly is necessary to transform whole pointcloud from lidar coordinate frame to camera frame. For this purpose are used so called extrinsic camera parameters, which describes where in lidar coordinate frame is camera placed. This type of transformation is discussed at the beginning of this section. Secondly we can use the camera intrinsic parameters to calculate the projection. Note that it is needed to work in homogenous 2D coordinate system.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{2.18}$$

Using this matrix equality is possible to decide which coordinates u, v on image plane corresponds to 3D point x, y, z from pointcloud and assign color of a certain pixel to the point. However this works only for the simplest pinhole camera model without any distortion of image. If the lens has non-negligible distortion, this distortion must be included in the camera model. For description of distortion are used the distortion coefficients.

Chapter 3

Application of methods

In following part of thesis is described how to apply discussed methods to our problem. It is necessary to make several adjustments and also combine some of these methods to ensure reliable detections.

3.1 Detection pipeline

Detection is divided into three parts. All three parts are discussed in next three subsections. detection pipeline is visualized in the figure 3.1.

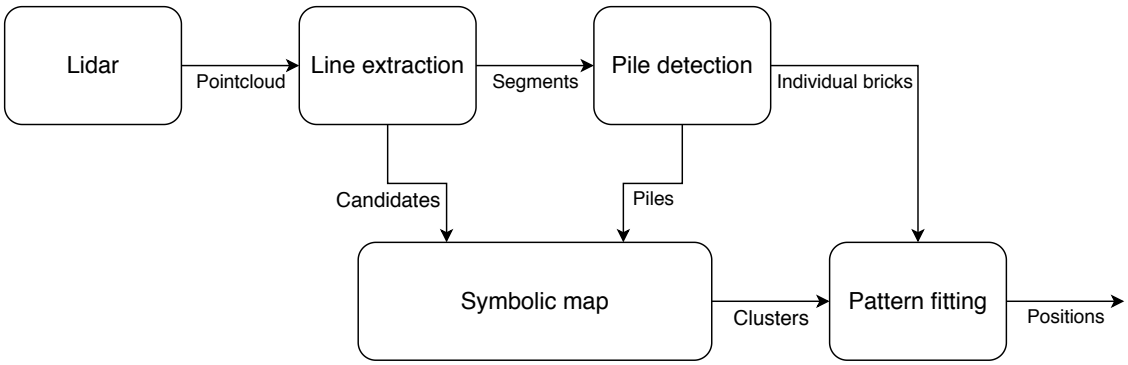


Figure 3.1: Visualization of subsequent steps of detection.

It can be seen that the symbolic map has two types of inputs. The main advantage of this approach is that it extends the lidar detection range. As we discussed in previous section, used lidar has low vertical resolution - only 16 layers. Thus the bricks are often

visible in only one layer of lidar scan. If we use only one scan, there is a high probability of false positive measurements. On the one hand we can decrease the occurrence of false positives by adding other lidar layers into the detection process. But on the other hand two layers are available only from distance smaller than 6 meters and that decreases the detection range. Therefore we exploited both approaches. One layer line segmentation for generating the candidates with low confidence and multilayer pile detector providing high quality estimates.

3.2 Line segmentation

For line segmentation is used IEPF algorithm very similar to the one described in algorithm 1. Only the final merging parallel segments is omitted, because it can connect two bricks into one. After retrieving the segments, a filtering based on the segment size is done. It is possible to assign the color to the segment because each brick type has unique dimensions. Example of lidar measurement with extracted and filtered lines is in the figure 3.2. Algorithm performance is influenced by correct setup of constants C and S (clustering and splitting distance). If we choose to high C, it could happen that the algorithm joins two bricks into one segment. As described in figure 1.2 the distance between two bricks of same color is 10 cm. Therefore the clustering distance must be always less than 10 cm. If the clustering distance is too low, one brick can be unintentionally divided into many segments and without merging at the end are these segments useless. It is necessary to bear in mind that the lidar precision as shown in table 1.1 is ± 3 cm so any clustering with distance smaller than 6 cm will be highly affected be sensor noise. The found segments are transformed into the map frame and passed to symbolic map as low confidence detections. Further are segments passed to pile detector which can filter out false positive detections.

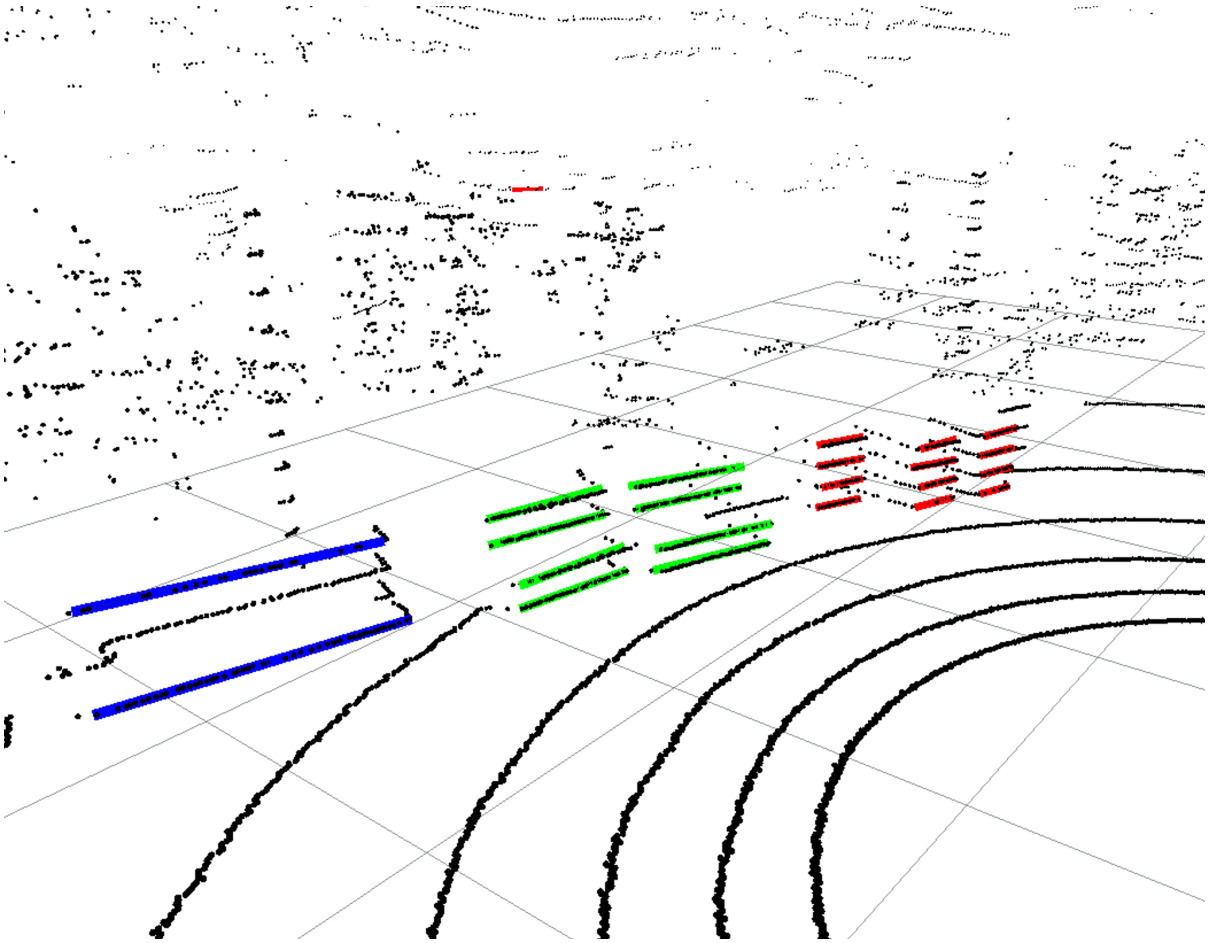


Figure 3.2: RViz visualization of line segmentation. In the figure is evident that the bricks are well detected. This detection is done from ≈ 3m. There is visible one false positive detection behind the brick piles on a tree.

3.3 Pile detection

The pile detector uses one of the simplest version of EM algorithm. As a model for the pile is applied 2D multivariete Gaussian distribution with variance fixed to one. Although, this model is not precise description of the detection probability, it has other advantages already discussed in previous chapters. It is easy to work with and it converges to the global optimum very well. Only the mean value is optimized and it should converge into the center of the pile. The stopping criterion for the algorithm is solely number of iterations. The robot is realtime system and there are strict demands for meeting a deadline. There are further requirements for a hypothesis to be declared as pile after the optimization is done. We look around the proposed center in one meter radius and we inspect all bricks found in this area. All the following conditions must be fulfilled for segments in the pile:

- There are at least two unique heights (z positions).

- There are at least two unique places ((x, y) positions).

- Difference between maximal and minimal height is less than the pile height.

- Pile center is inside the arena.

When these conditions are met then the hypothesis is declared as pile and pushed into symbolic map with high confidence. When one of these conditions is violated then all the segments in this hypothesis are deleted and the algorithm runs again until there are no more segments within the hypothesis radius. Second condition does not apply to the orange pile. Whole procedure is shown in algorithm 4.

```
Data: segments
Result: pile_position
while True do
    pile_position = fit_em(segments);
    pile_segments = segments_in_pile(pile_position, segments);
    if pile_segments.size() < 2 then
        return None;
    end
    if is_pile(pile_segments) then
        return pile_position;
    else
        segments.delete(pile_segments);
    end
end
```

Algorithm 4: Algorithm to obtain pile centers.

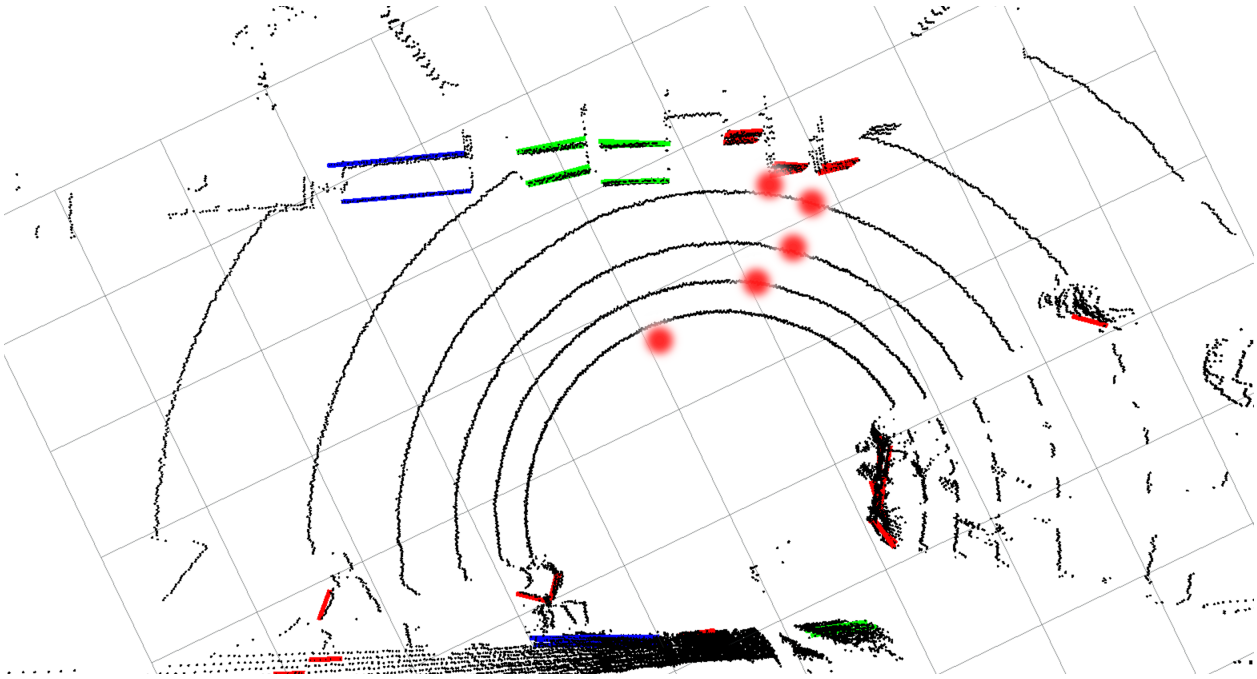


Figure 3.3: Several steps of EM algorithm for the red pile. Colored lines are the visualized segments same as in the figure 3.2. Red dots show subsequent positions of mean value of multivariete Gaussian. Although, there are many false positives especially in the bottom of the picture, the pile model ensures that the algorithm converges to correct place.

3.4 Pattern fitting

Ah shown in the figure 3.1 the last step of detection pipeline is pattern fitting. It is also visible that there are two types of input into this last step. Each input is used for different type of pattern fitting. In previous two sections it was described how to generate candidates with different levels of confidence. This section describes how the candidates can be used for generating actual positions and how information about the spatial distribution of bricks can be exploited for the detection. From figure 1.2 is known exact position of each pile and even each brick. Pattern in which are bricks stacked has three degrees of freedom - position (x, y) and rotation ϕ. Goal of the pattern fitting is to find values of these parameters in the map frame.

3.4.1 Brick fitting

The first type of fitting is using detected 3D positions of individual bricks obtained in pile detector. In the pattern is described position of each brick which can be detected by

lidar sensor. Next step is to generate hypothesis which align this pattern with the measured positions of bricks. For this step is utilized the RANSAC algorithm. Firstly we draw two different bricks from detected set of bricks. Secondly the correspondences are used to find the transformation - two correspondences are enough to generate the hypothesis (rotation matrix R and translation vector t). The brick types (colors) and brick z positions are used as the correspondences. Also we know that the distance between corresponding pairs should match, otherwise it would be incorrect correspondence. Lastly we measure the cost of the hypothesis. When the bricks are stacked into the pattern with a reasonable precision, it is possible to fit the pile with average brick error down to $\pm$3cm. After such a observation we set the inlier distance to 5cm. Algorithm is stopped after certain number of iterations. The result is passed only if all detected bricks are inliers.

Hypothesis ambiguity

It is obvious that we cannot properly test the hypothesis when there are not enough bricks found. In addition we can see the pile from behind which renders a two different patterns to match. For this reason is required at least five detected red bricks to even start the brick fitting. As shown in the figure 3.4 even with four detected red bricks can be hypothesis easily fitted incorrectly. All conditions which can start fitting the hypothesis from front view are listed bellow:

- 5 red bricks detected.
- 3 green bricks detected.
- 4 bricks of at least two colors detected.

For view from behind is sufficient only the last condition.

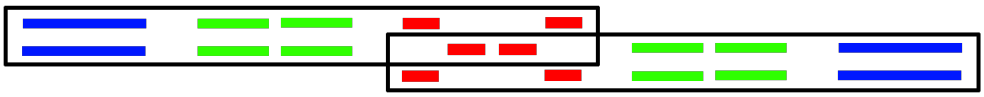


Figure 3.4: In black rectangles are visualized two possible hypothesis which can be generated by four red bricks in the intersection of rectangles. This is top view of detection from the front, so it is not visible that all red bricks are in two layers.

3.4.2 Cluster fitting

The second type of fitting polls clusters from symbolic map and utilizes their confidence and spatial distribution. For RANSAC algorithm would be hypothesis using only

four clusters too sparse to fit reliably. Because of this reason is the cluster fitting done by the EM algorithm. As in pile detection is used multivariete Gaussian distribution to represent the pile but this time the model takes in account relationships between positions of the piles. We define the probability model as follows:

$$P(\vec{x}_m) = \mathcal{N}_m(\vec{x}_m; \vec{\mu} + k_m\vec{v}; \mathbf{\Sigma}), \tag{3.1}$$

where m is pile (color) index and M is absolute number of colors, $\vec{x}_m$ is measurement of color m, $\vec{\mu}$ is mean value of whole model, $\mathbf{\Sigma_m}$ is covariance matrix, k_m is scalar multiplier unique for each pile and $\vec{v}$ is defined as:

$$\vec{v} = \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix}, \tag{3.2}$$

where ϕ is rotation of model. This definition of model ensures that all piles (Gaussians) are in the line with distance defined by multiplier k. Now we want to obtain the position and rotation of such a model based on real measurements. There is a closed form solution for maximizing both terms which can be found using maximum likelihood estimate. For mean value is derivation very similar to multivariete Gaussian:

$$\frac{\partial \log \mathcal{L}}{\partial \vec{\mu}_m} = \mathbf{\Sigma}_m^{-1} \sum_{n=1}^{N_m} (\vec{x}_{m_n} - \vec{\mu} - k_m\vec{v}) \tag{3.3}$$

$$\vec{\mu_m} = \sum_{n=1}^{N_m} \frac{\vec{x}_{m_n} - k_m\vec{v}}{N_m}. \tag{3.4}$$

Further is necessary to derive MLE for rotation ϕ which is hidden inside vector $\vec{v}$. For simplicity is now the matrix equation split into one part for each of two dimensions.

$$\frac{\partial \log \mathcal{L}_x}{\partial \phi_m} = \frac{1}{\sigma_x^2} \sum_{n=1}^{N_m} (x_{m_n,x} - \mu_x - k_{m,x}\cos\phi)\,(-k_{m,x}\sin\phi) \tag{3.5}$$

$$\frac{\partial \log \mathcal{L}_y}{\partial \phi_m} = \frac{1}{\sigma_y^2} \sum_{n=1}^{N_m} (x_{m_n,y} - \mu_y - k_{m,y}\sin\phi) k_{m,y}\cos\phi \tag{3.6}$$

Likelihood derivative is now set equal to zero to find the extremes for each dimension.

$$\cos\phi_m = \frac{1}{k_{m,x}N_m} \sum_{n=1}^{N_m} x_{m_n,x} - \mu_x \tag{3.7}$$

$$\sin\phi_m = \frac{1}{k_{m,y}N_m} \sum_{n=1}^{N_m} x_{m_n,y} - \mu_y. \tag{3.8}$$

Now it is possible divide one equation by the other, use basic relationship of trigonometric functions and derive final expression for maximizing the rotation ϕ:

$$\phi_m = \arctan\left(\frac{\frac{\sum_{n=1}^{N_m}(x_{m_n,y}-\mu_y)}{k_{m,y}N_m}}{\frac{\sum_{n=1}^{N_m}(x_{m_n,x}-\mu_x)}{k_{m,x}N_m}}\right). \tag{3.9}$$

Although the expression can be further simplified, in this form it is easier to weight each data sample by the expectation α. In addition is necessary to consider contribution of each color to pile model. This can be done in two ways. One possibility is to set equal contribution to each data sample. In our case this could lead to bias towards the most detected color (usually red). Thus we set equal contribution to each color instead of data sample. Final form used in maximization step looks as follows:

$$\vec{\mu} = \sum_{m=1}^{M}\frac{\sum_{n=1}^{N_m}\vec{\gamma}_{m_n}(\vec{x}_{m_n}-k_m\vec{v})}{M} \tag{3.10}$$

$$\phi = \arctan\left(\frac{\sum_{m=1}^{M}\frac{\sum_{n=1}^{N_m}\gamma_{m_n,y}(x_{m_n,y}-\mu_y)}{k_{m,y}}}{\sum_{m=1}^{M}\frac{\sum_{n=1}^{N_m}\gamma_{m_n,x}(x_{m_n,x}-\mu_x)}{k_{m,x}}}\right). \tag{3.11}$$

Method is demonstrated in the figure 3.5. Expectation is calculated simply by evaluating the probability of sample $P(\vec{x}_m)$ as in equation 3.1. Results can be further improved when the confidence of sample is used as the prior probability.

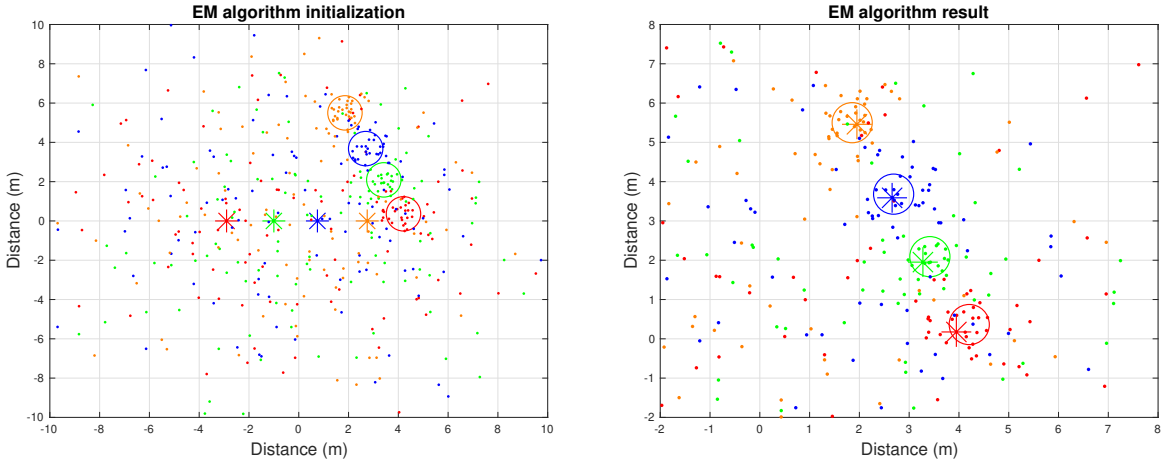


Figure 3.5: In the pictures is visualised pattern fitting using EM algorithm. On the left is algorithm initialized with parameters $\vec{x} = (0, 0)$ and $\phi = 0$. Algorithm estimate is marked by stars. Points are generated by sampling multivariete Gaussian distribution. Model was sampled with parameters $\vec{x} = (3, 3)$ and $\phi = 2$ to create desired pattern of points. Positions of sampled piles are marked as circles. At the end of process are parameters found correctly.

Convergence

The unimodality of model's likelihood was disrupted by adding the rotation parameter to Gaussians. Possible local optimum is shown in the figure 3.6. Now there are no guarantees that the algorithm will converge into the global optimum. This is common issue of EM algorithm when dealing with more complex problems. Many solutions of this issue was proposed. Very advantageous is that local optima are usually significantly worse in terms of likelihood than the global optimum. One of methods how to avoid local optima is the deterministic annealing which influences how the expectation is used in maximization step [14]. Another way how to escape local optimum is to apply the perturbations to parameters of model. In our case it could be for example rotating the model by 180°. Different approach would be population based em algorithm with multiple initializations or informed initialization. The latter is easily applicable in our case because the confidences of clusters could be used in the expectation step. Because we have the formulas for gradient it is also possible to use arbitrary gradient ascend method instead of EM algorithm to maximize likelihood of the model.

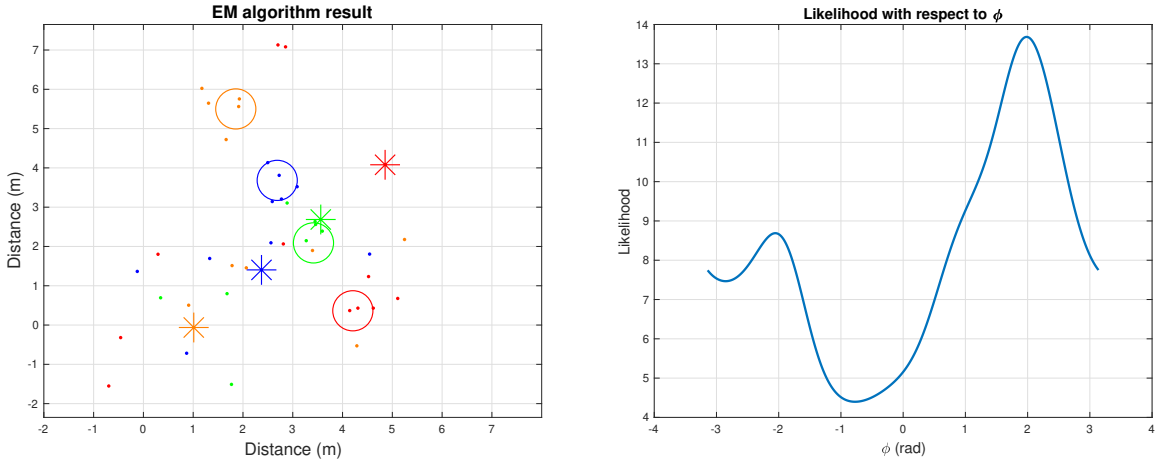


Figure 3.6: When there is not enough samples the EM algorithm is more susceptible to ending in local optima as shown on the left. Right picture shows how different rotations of model with fixed mean influences the likelihood and it also shows that there is a local maximum. The likelihood is maximal when $\phi = 2$ which is optimal rotation of model.

3.5 Arena exploration

It is vital to emphasize on proper arena exploration. If all the interest points are not found, the robot can not proceed in completing of the challenge. Except the initial brick position is necessary to find also the destination where the bricks should be placed. This place is marked by checker pattern in the figure 1.3a. Only sensor which can detect the destination pattern is camera. The big advantage is that the camera is mounted onto the arm so that it can be raised into height and turn around. As can be seen in the figure 3.7 the most limiting factor of the detection range is currently the destination pattern search.

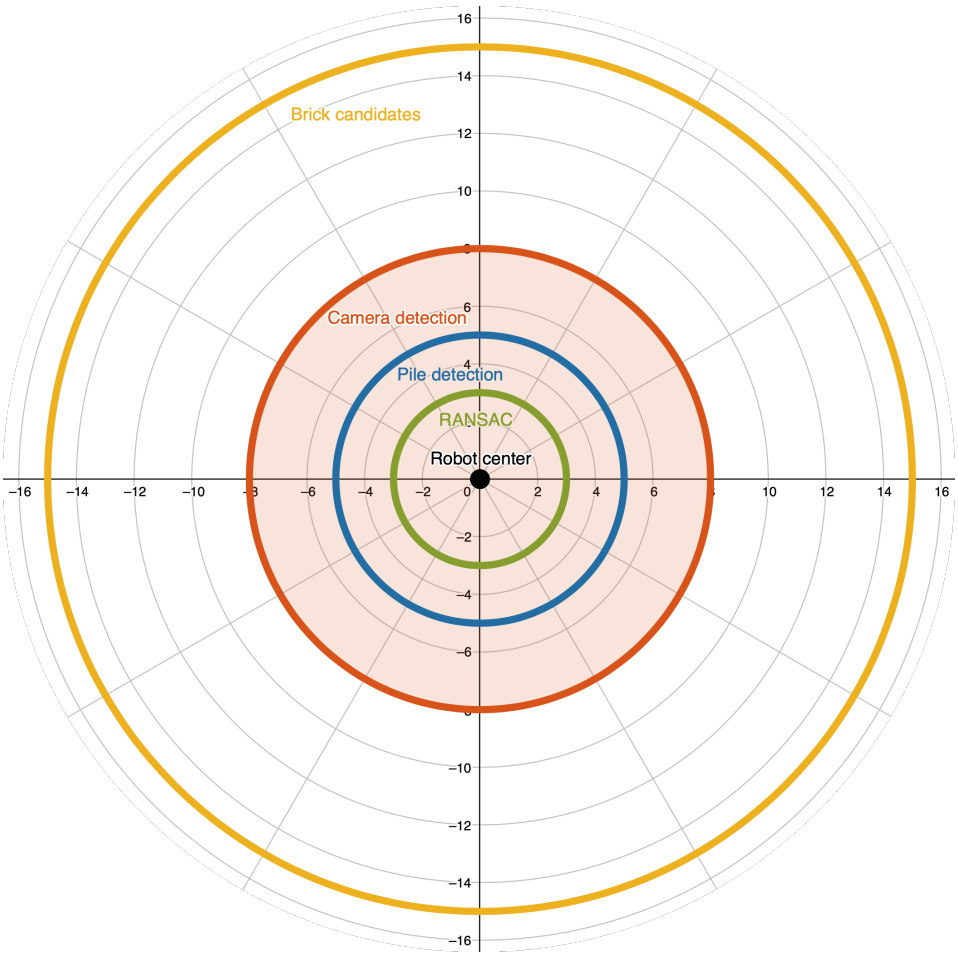


Figure 3.7: In the picture we see the range of each type of detection. Fit the complete hypothesis using RANSAC method (green) requires high number of segments in piles, so the range is around 3m. For detection of the pile (blue) is required just few segments - the detection range is roughly 5m. Camera can obtain candidates for checker pattern (red) from maximal distance of 8m, that is the limit distance for are exploration. From even bigger distance can be generated candidates for bricks (yellow) using the lidar sensor. The upper boundary can be even higher but we set it manually to 15m to reduce the number of false positive detections.

The waypoints of exploration movement should be generated so that the area of whole arena is covered by circles with 8 meter radius. Generated waypoints in the arena are visualized in the figure 3.8. On each waypoint the robot must stop, raise the arm and look around with camera. There are several reason while camera detection cannot be done while moving. First of all the arm can not be raised to the highest possible position while the robot is moving, otherwise it could get stuck or damaged. In addition the motion blur of image would reduce the detection range even further. Although, during camera detection the robot must be still, the lidar detections can be done while moving, which

further improves the detection. Whole map exploration is described in the algorithm 5.

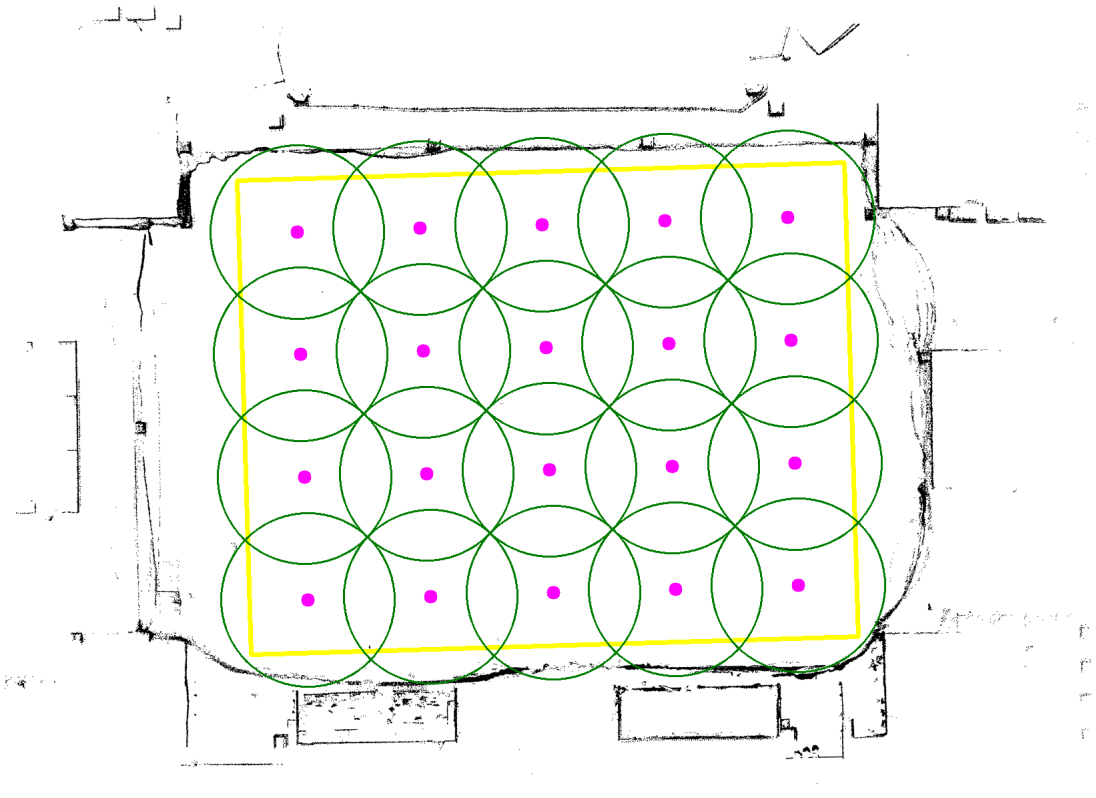


Figure 3.8: Purple waypoints are places where the robot should stop and look around. Green circles are camera ranges from the corresponding place. In this figure are the waypoints generated so that there is no unexplored area. That is usually not necessary since the objects of interest have non-negligible size. It is thus possible to further reduce the number of waypoints.

```
Data: waypoints
done = False;
while not done do
    waypoint = waypoints.pop();
    start_lidar_detection();
    go_to(waypoint);
    stop_lidar_detection();
    raise_arm();
    do_camera_detection();
    if has_all_objects or waypoints.empty() then
        done = True;
    end
end
if not has_all_objects then
    go_to(get_strongest_candidate());
end
```

Algorithm 5: Algorithm to explore whole map. The waypoints are ordered prior to this algorithm. How to order them is beyond the scope of this thesis. In algorithm is visible that the method **start_lidar_detection** is non-blocking service which just turns on the detector, whereas the method **do_camera_detection** is blocking service which waits until the arm looks around with camera and finishes the detection. This method is also the most time consuming part of the loop.

3.6 Stacked bricks detection

It is obvious that, when the bricks are stacked close to each other without any significant gap, it is impossible to distinguish which bricks the lidar detects. This issue was already addressed in previous chapters. It is desired to know what color particular lidar point has. This is done by camera to lidar registration. Since we know the relative position of camera to lidar and also the intristic camera matrix, it is not problem to assign color to each point. The colors are further utilized during the clustering. We can now split clusters not only using the spatial data but the splitting can be based on color difference of subsequent points. Modified clustering is visible in algorithm 6. Note that the color distance is computed from whole cluster mean to a new point. When the new point is added, the color of cluster should be recalculated. This is important to reduce camera noise influence on final form of clusters. When the running mean technique is used, the clusters are split only in sharp color transitions. How the pointcloud coloring and final detection looks like is shown in the figure 3.9. Since the robot stacked all of these bricks, their relative position should be known. Whole stacking is predefined sequence of movements and thus we can add each placed brick into the list of stacked bricks with its relative position to place where stacking started. These known positions can be used to generate hypothesis which can be

fitted with the RANSAC algorithm in the same way as during the pattern fitting. When we use colors for the segmentation, it is important to choose right color space. During the experiments was much easier to distinguish between green and blue color in LAB color space than in HSV color space. The robot is not capable of carrying orange bricks so that it was not necessary to segment also the orange color which is problematic, because it can be easily confused with red color. This part of detection is not discussed any further because it was more important to localize initial position of UGV bricks and placing more than one load of bricks was not achievable in given timeframe.

```
Data: points
Result: clusters
initialize constants C, T, min_size ;
cluster = [];
clusters = [];
foreach pt in points do
    if cluster.empty() then
        cluster.push_back(pt);
    else
        if distance(pt, cluster[end]) < C and color_distance(pt, cluster) < T then
            cluster.push_back(pt);
        else
            if cluster.size() > min_size then
                clusters.push_back(cluster);
            end
            cluster.clear();
        end
    end
end
```

Algorithm 6: Spatial and color clustering. Input points must be ordered layer from lidar pointcloud. Constant **C** is clustering distance, **T** is color clustering distance and **min_size** is minimal cluster size - this has high influence on range of brick candidate generation.

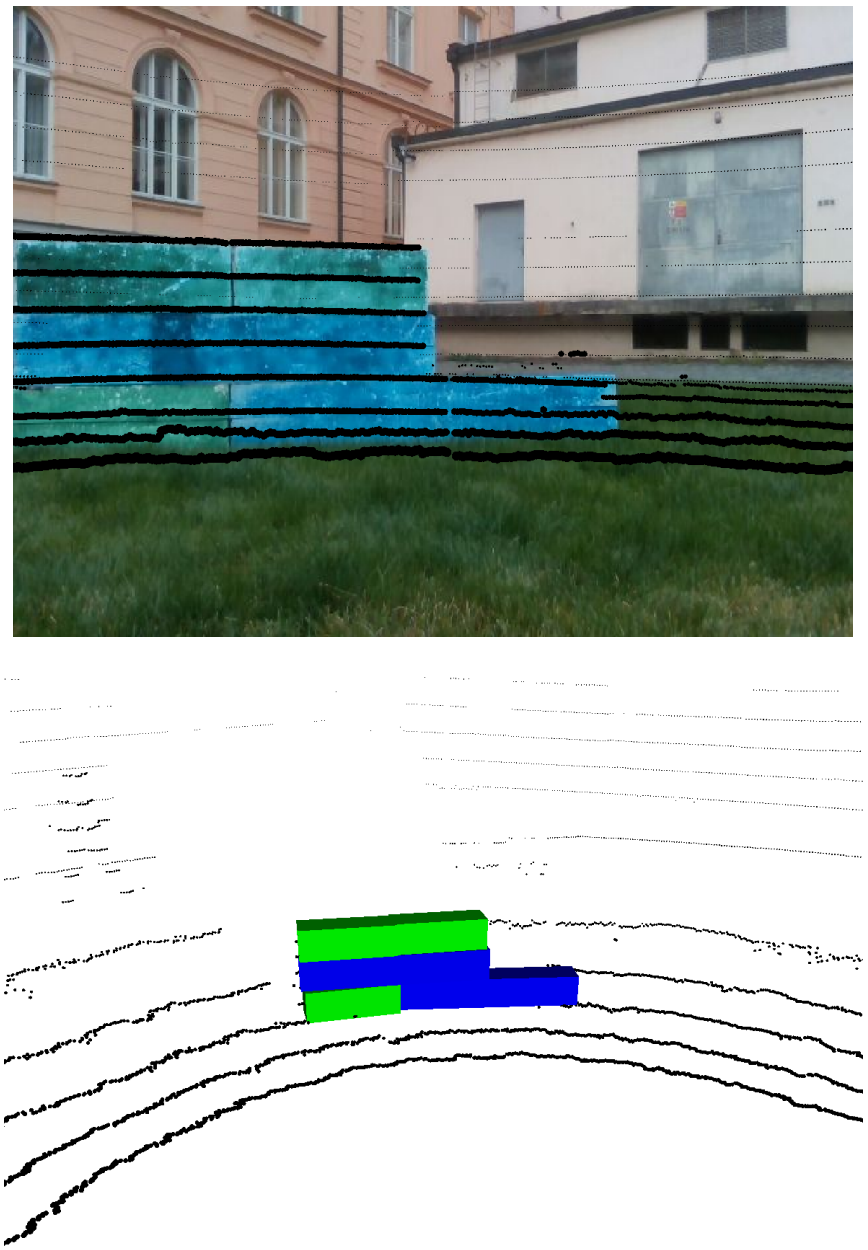


Figure 3.9: Detection of bricks using color based clustering. At the top is image from camera pointing at built wall. Black dots are points captured by the lidar sensor. At the bottom is shown final detection in map frame. All of the bricks are correctly detected, although they are poorly painted and the environment is quite challenging, especially with the green grass on the ground.

Chapter 4

Experiment

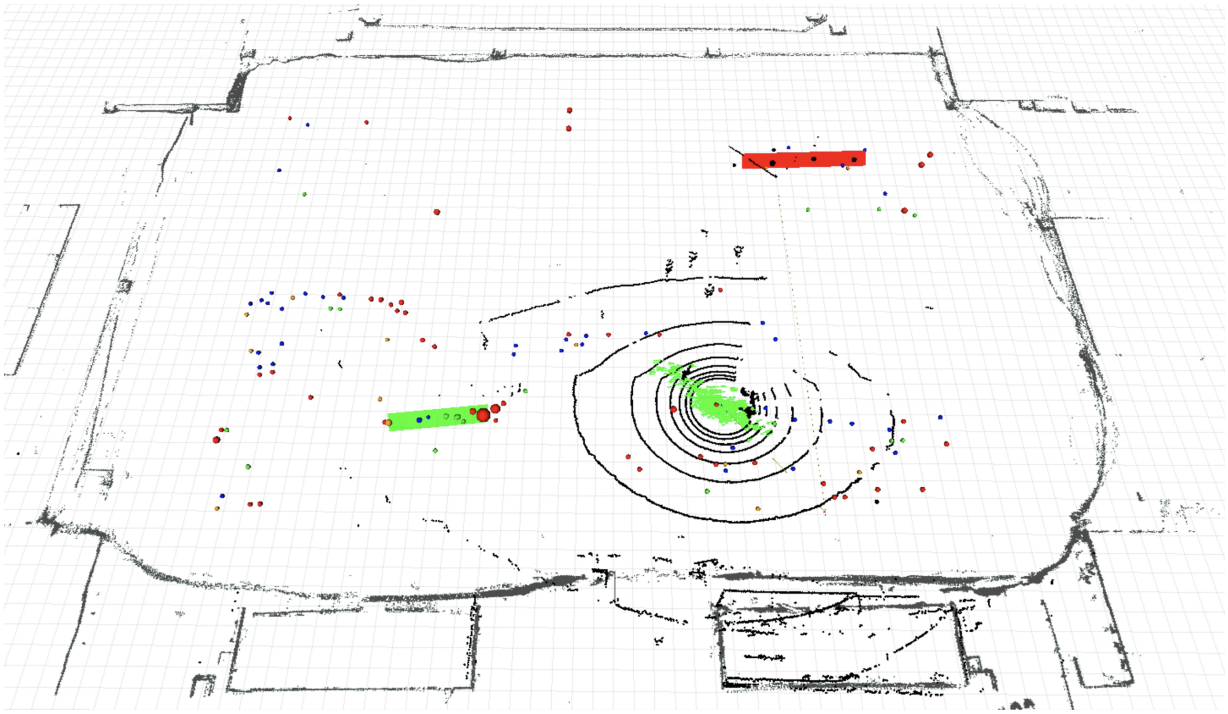


Figure 4.1: Blah Blah

Chapter 5

Conclusion

BLAH BLAH

5.1 Future work

BLAH BLAH

Bibliography

[1] Alessandro Gardi Roberto Sabatini. Lidar obstacle warning and avoidance system for unmanned aircraft. 2007.

[2] Johannes Meyer Stefan Kohlbrecher. Hector open source modules for autonomous mapping and navigation with rescue robots. 2017.

[3] Johannes Meyer Stefan Kohlbrecher. Lidar-based 3d object perception. 2008.

[4] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.

[5] Theodosios Pavlidis Steven L. Horowitz. Picture segmentation by a tree traversal algorithm. 1974.

[6] Axel Kaske Ali Syadat. An optimized segmentation method for a 2d laser-scanner applied to mobile robot navigation. 1997.

[7] Stefan Gachter Viet Nguyen. A comparison of line extraction algorithms using 2d range data for indoor mobile robotics. 2006.

[8] N. M. Laird A. P. Dempster. Maximum likelihood from incomplete data via the em algorithm. 1977.

[9] Nicolas Vandapel Jean-Francois Lalonde. Natural terrain classification using three-dimensional ladar data for ground robot mobility. 2006.

[10] Robert C. Bolles Martin A. Fischler. Random sample conseunsus: A paradigm for model fitting with applications to image analysis and automated cartography. 1981.

[11] Jan Matas Ondrej Chum. Optimal randomized ransac. 2008.

[12] Tully Foote. tf: The transform library. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Open-Source Software workshop, pages 1–6, April 2013.

[13] Andrew Zisserman Richard Hartley. *Multiple view geometry in computer vision*. Cambridge University Press, 2017.

[14] R. Nakano N. Ueda. Deterministic annealing em algorithm. 1998.

.1 CD Content

In Table 1 are listed names of all root directories on CD.

Directory name	**Description**
thesis	the thesis in pdf format
ctu_thesis	latex source codes
lidar-gym	OpenAI gym environment

Table 1: CD Content

.2 List of abbreviations

In Table 2 are listed abbreviations used in this thesis.

Abbreviation	Meaning
EM	Expectation maximization

Table 2: Lists of abbreviations