Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Autonomous Robotic Exploration of Underground Environments

Identifying suitable exploration goals in a sparse point cloud map

Lars Kahlert

Supervisor: Ing. Tomáš Petříček, Ph.D
Field of study: Cybernetics And Robotics
Subfield: Cybernetics And Robotics
August 2021

Acknowledgements

I would like to thank my supervisor Tomáš Petříček his feedback and help with this work. I also like to thank my family for supporting me in my studies. And lastly I would like to thank my friends for proofreading this thesis, and keeping my spirits up in those past few years.

Declaration

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.

In Prague, 13. August 2021

Abstract

Aim of this bachelor project is to develop and test a frontier detection method working on point cloud maps, as an alternative to voxel-based probability mapping, and compare the strengths and weaknesses of such approach to an exploration. Optimal parameters and method behavior is found through tests on simple objects. Additionally, three methods for filtering out false detection of points are proposed. Then the method is tested in a simulated underground environment, and compared with voxel-based frontier detection.

Keywords: point clouds, frontier based exploration, voxel mapping, autonomous exploration

Supervisor: Ing. Tomáš Petříček, Ph.D

Abstrakt

Cílem bakalářské práce je vyvinout a otestovat metodu na průzkum hranice podél známého prostoru pracující s mapami mračen bodů, jako alternativa k voxelovému pravděpodobnostnímu mapování, a srovnat silné a slabé stránky tohoto přístupu k exploraci. Optimální parametry a chování metody získáme skrze testování na jednoduchých objektech. Navíc jsou navženy tři metody pro filtraci falešných detekcí bodů. Následně je metoda otestována na simulovaném podzemním prostředí a porovnána s voxelovou metodou detekce hranice podél známého prostoru.

Klíčová slova: mračna bodů, průzkum podél hraničního prostoru, voxelové mapování, autonomní explorace

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation and goals

Autonomous exploration deals with the problem of efficiently mapping unknown space. This can be used to safely explore environments dangerous for humans, or environments where communication between the robot and human driver cannot be guaranteed, such as underground spaces.

Over time different methods for facilitating exploration were developed. Notably occupancy grid mapping [6], in which a grid map holds values representing the probability that the cell is occupied. With these values being constantly updated with new measurements. Later Yamauchi et al. [1] came with the concept of frontiers, as the areas between explored and unexplored space, with the idea being that the robot would continuously head to in the direction of frontiers, thus expanding explored space, until the whole area is explored.

The goal of this paper is to create, and test a method for identifying suitable exploration goals on a point cloud map, as opposed to voxel grid occupancy maps. This should yield some advantages, mainly simplicity and efficiency, as a point cloud map may already be created during simultaneous localization and mapping (SLAM).

1.2 Related works

Yamauchi et al.[1] proposed the idea of frontiers. Works like [8] deal with autonomous exploration and mapping but work on voxel grids. In [9] different frontier-based approaches are evaluated, thought those also work on grid maps. While there are propositions for utilizing point clouds for terrain assessment, and motion path planning [7], none would consider autonomous exploration.

1.3 Outline

In chapter 2, used terms are explained and the proposed frontier detection algorithm is described. In chapter 3 we test the algorithm on simple geometrical objects and find optimal input parameters. We also try methods for rectifying the algorithms weaknesses. Afterwards in chapter 4 experiments are done in the SubT virtual environment. A comparison with voxel-based frontier detection is made, and pros and cons are discussed. The final results of the project are in chapter 5.

Chapter 2

Method description and definitions

2.1 Term explanations

Frontier - "*regions on the border between space known to be open and unexplored space.*"[1] Used in robotic exploration to determine the best directions to explore.

Frame/Coordinate system - A system of coordinates to uniquely determine the position of points in space.

Point - In the following text we will understand a point as a set of coordinates, describing a position in a given coordinate system. Example for three-dimensional space:

$$\mathbf{P} = [P_X, P_Y, P_Z] \tag{2.1}$$

Point cloud - A set of points in the same coordinate system, usually the output of a sensor, and describing a physical object or space [5] (see fig 2.1).

Voxel - If we split three-dimensional space into a uniform grid. A single "cube" on this grid would be called a voxel.

Voxel-based mapping - For the purposes of this document voxel-based mapping is the process of splitting space into voxels, and setting their occupancy values based on laser measurements. For example, we increase the value for a voxel that contains the end point of a laser measurement, and decrease the value of all voxels intersecting the measure line. Then it is possible to classify voxels, based on their occupancy values, as representing empty, full or unexplored space, thus receiving a map of the environment (see example fig. 2.2).

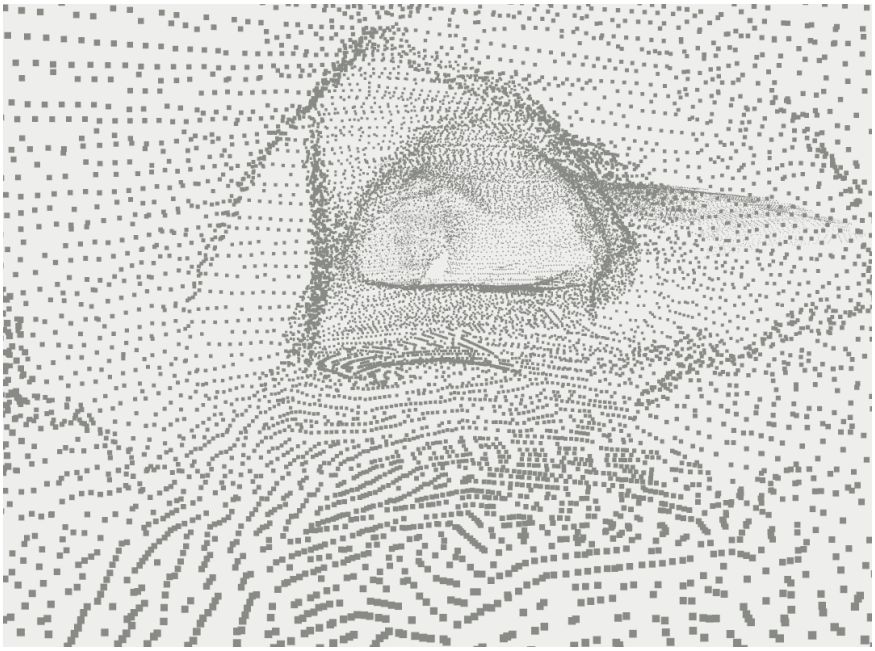


Figure 2.1: Example of a point cloud, mapping an underground environment.

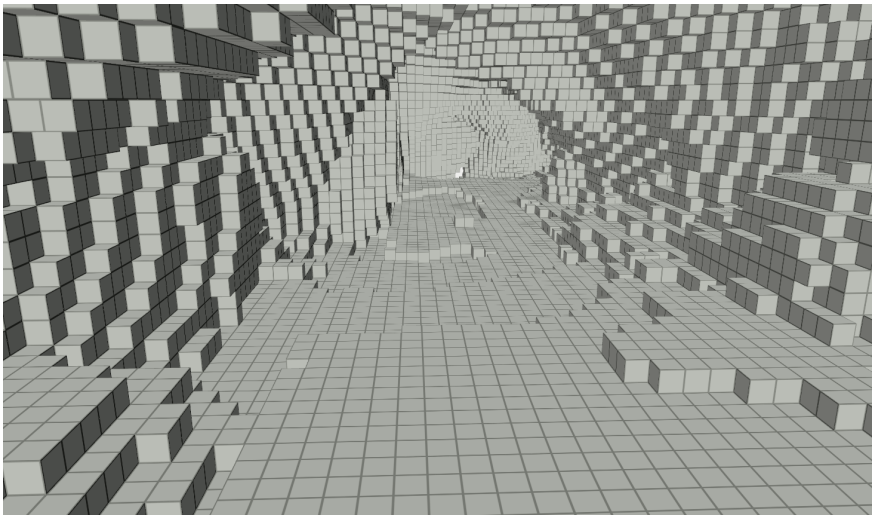


Figure 2.2: Example of a voxel map.

2.2 Mean vector method

The mean vector method is a method designed for finding frontiers on a sparse point cloud.

2.2.1 Method input and output

The method has the following inputs:

- Point cloud - A sparse point cloud, for example the output of a lidar sensor.
- Minimal point distance (MPD) - Parameter of the point cloud. Sets the minimal distance two points in the point cloud can have.[1]
- Local neighbourhood radius (LNR) - This input sets the distance from the test point in which other points will be used for calculations. It must be bigger than MPD (see fig. 2.3).
- Threshold - Threshold for classifying point as frontier.(see eq. 2.3)

The method has the following output:

- Set of frontier points, a subset of the input point cloud.

[1] This can determined by the robot sensor, or by the user to limit the number of points.

2.2.2 Method description.

1. Select a point to test. $\mathbf{P_t}$ in the following text.
2. Find local neighbourhood points ($\mathbf{P_i}$). Points closer to $\mathbf{P_t}$ than LNR.
3. Calculate the mean vector ($\mathbf{P_m}$) of this point.

$$\mathbf{P_m} = \frac{\sum_{i=1}^{N} \mathbf{P_i}}{N} \tag{2.2}$$

   Where N is the number of points in LNR.

4. Point is considered a frontier if 2.3 is true.

$$\frac{||\mathbf{P_t} - \mathbf{P_m}||}{\text{LNR}} > \text{threshold} \tag{2.3}$$

In figure 2.3 we can see the effect of the test point position on the resulting distance between the test point and the average vector, with points positioned more on the border having larger distance and corners in particular.

Though we have to keep in mind that the algorithm by itself only detects edges. Any differentiation between edges on the border of unexplored space, and edges given by object geometry has to be done separately (see chapter 3.3.2).

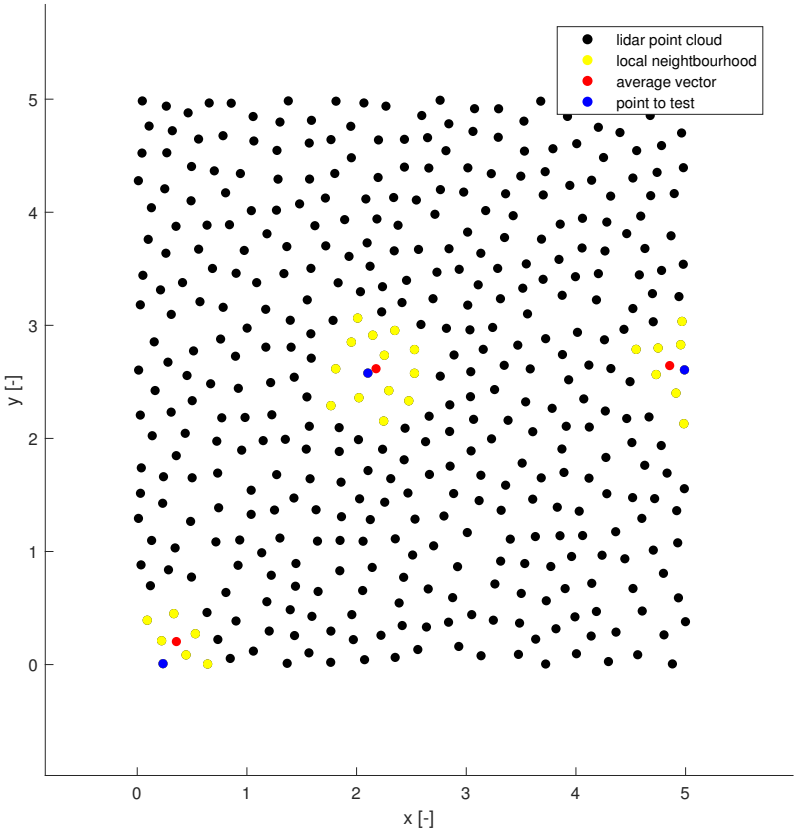


Figure 2.3: Showcase of the mean vector method on a plane. Distance between test point and average vector is bigger for points on the edge of the point cloud.

Chapter 3

Testing on simple objects.

In this chapter we will look at the method behaviour on simple geometrical objects and try to find optimal values for method inputs.

3.1 Test conditions

All testing in this chapter has been done on MATLAB version r2020b. If not specified otherwise values in table 3.1 apply. True frontiers were selected as points closer to the edge than MPD.

Parameter	MPD	LNR	T
Default value	0.1	0.35	0.22

Table 3.1: Default values used for testing.

3.2 Effects of changing local neighbourhood radius

Any changes to LNR will always be considered in relation to MPD, and will be expressed in radius in minimal distance units r.

$$r = \text{LNR}/\text{MPD} \tag{3.1}$$

In figure 3.1, are shown the effects of changing r values, while keeping a fixed threshold, can have on the classification of points (see eq. 2.3). With smaller radii using less points for calculations and thus being more affected by random noise. We see that values between one and two don't consider enough points to be useful for detection. At r = 2 corner points are detected, although still sparse. At values 2.5 and higher we get solid edge detection, with higher

values increasing the width of classified points area, and decrease the chance of misclassifying surface points[1] and increases computational complexity.

3.3 Frontier detection

Now we can test the ability to detect frontiers. For this we take some simple incomplete shapes. For the sake of comparison any point closer to the edge of the object than MPD shall be considered a true frontier.

On figure 3.2 we can see the algorithm output for r = 3.5 and threshold = 0.22 for different shapes. The algorithm reliably detects object edges for flat and rounded surfaces. On the cube (3.2b) and tunnel (3.2d) shapes we can also see the problem with the algorithms inability to differentiate between object geometry edges and point cloud edges 3.1.

3.3.1 Finding optimal threshold values

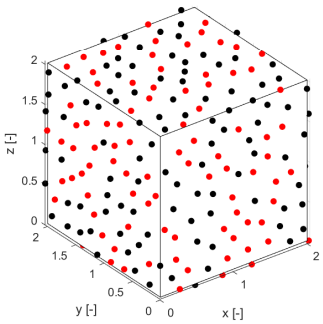
Now that we quantify the detection quality. We can draw a receiver operating characteristic, also known as a ROC curve. This curve plots the true positive rate against the false positive rate (Green and red points respectively in figure 3.2), for the changing value of the classifying threshold, thus showing us the classification capabilities for a given threshold. On figures 3.3 we can again clearly see the effects of changing r, with values less than 1.5 having little classifying ability and little change with raising values over 2.7.

The best threshold values, according to Youden's J statistic [2] are marked in the graph and in table 3.2. For low r values there is a lot of noise and a higher threshold value minimizes the false positives in the faces of objects. At around $r = 2$ the noise in the points in object faces is low enough that the number of false negatives at frontier zones gains statistical significance. Thus a lower threshold value maximizes the true positives at frontiers. With increasing r values the optimal threshold value also rises to counteract the expanding positive classification zone, that would otherwise occur (see fig. 3.1). For the expected use value of $r \approx 3$ a threshold value of 0.22 seems reasonable.

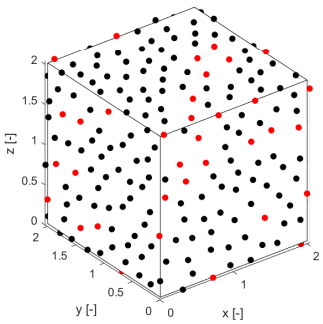
3.3.2 False positive detection.

By now we have encountered the problem of distinguishing between object edges and frontiers multiple times. In this section we look at three possible methods for detecting false positives.

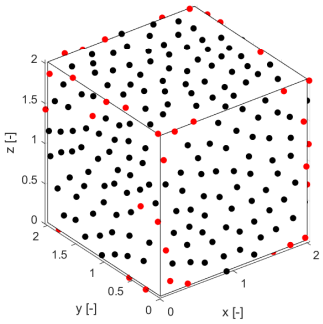
[1] Although given the random placement of points a chance for misclassification always remains.



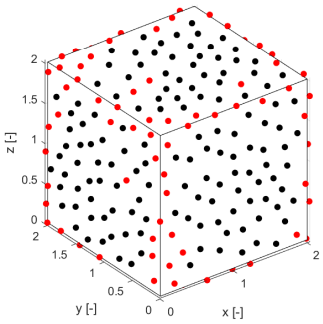
(a) : r = 1.1



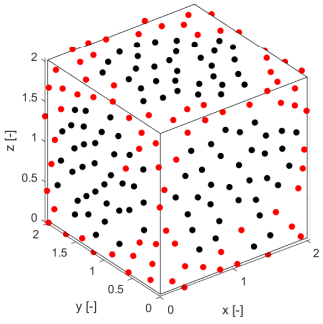
(b) : r = 1.5



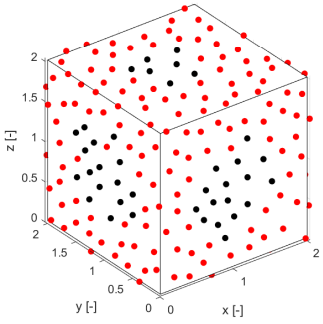
(c) : r = 2



(d) : r = 2.5

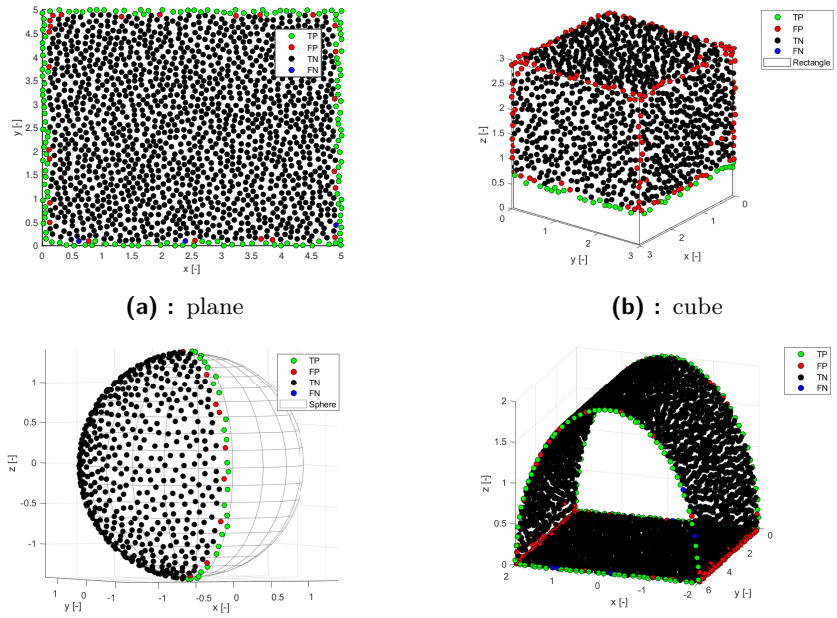


(e) : r = 3



(f) : r = 4

Figure 3.1: Display mean vector algorithm on a cube for various values of r (see eq. 3.1) and a fixed threshold value.



(a) : plane **(b) :** cube

(c) : sphere **(d) :** tunnel

Figure 3.2: Display of frontier selection capabilities of mean vector algorithm.TP - True Positive, FP - False Positive, TN - True Negative, FN - False Negative

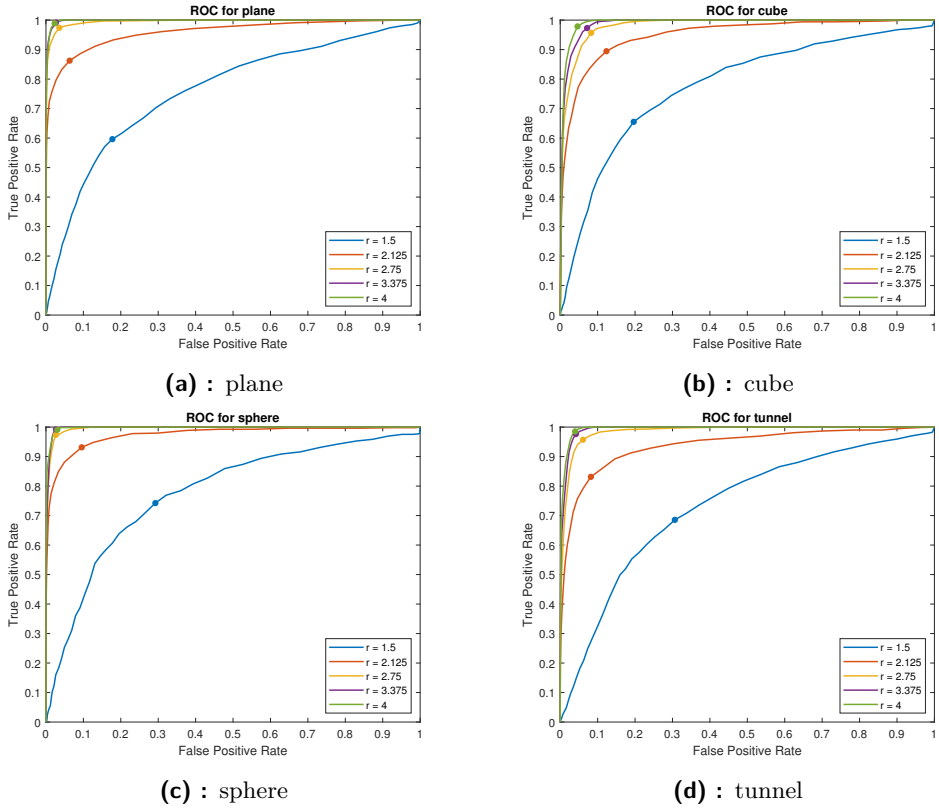
r	1.5	2.12	2.75	3.38	4.0
Plane	0.23	0.18	0.17	0.20	0.23
Cube	0.33	0.25	0.24	0.24	0.25
Sphere	0.23	0.18	0.21	0.23	0.26
Tunnel	0.24	0.20	0.19	0.21	0.24
Mean	0.26	0.21	0.20	0.22	0.25

Table 3.2: Optimal threshold values for a given LNR/MPD ratio

Number of points in LNR

This method uses the number of points in the local neighbourhood. With frontier points having less points in their LNR (see 3.4). So, for each frontier detection we check the number of points in LNR, and if the number is bigger than a point threshold we consider it a false positive and change it accordingly.

Now the problem of selecting the right cutoff point arises. A reasonable assumption would be that frontier points have half the number of points in their local neighbourhood, by virtue of being in the middle between explored (filled) and unexplored (empty) space. We can approximate the number of points by first calculating the point density $\rho_{points} = \frac{1}{MPD^2}$ times the area of the local neighbourhood $A = \pi \text{LNR}^2$. So our point threshold is:



(a) : plane **(b) :** cube

(c) : sphere **(d) :** tunnel

Figure 3.3: ROC curves for simple objects, with highlighted best classification threshold values (see table 3.2).

$$T_{points} = \frac{\pi \mathrm{LNR}^2}{2\mathrm{MPD}^2} \tag{3.2}$$

On figure 3.5 is an example of the method effects. If we compare these with the classification in figure 3.2, we see that it got rid of most of the false classified points, although as seen in the cube example, corners are still false positives.

Distance from fitted plane

This method works on the principle of detecting object edges and filtering them out. We do this by fitting the local neighbourhood points with a plane. Projecting the average vector on to this plane, and measuring the length of this projection. With edge points having larger distance (see fig 3.6).

Now we have to find a the decision threshold (T_f). The effects of different thresholds can be seen on figure 3.7. Values too small start detecting any sort of curvature on the point cloud, while values to high stop detecting edges and the algorithm doesn't fulfill it's purpose. Also because this method works on

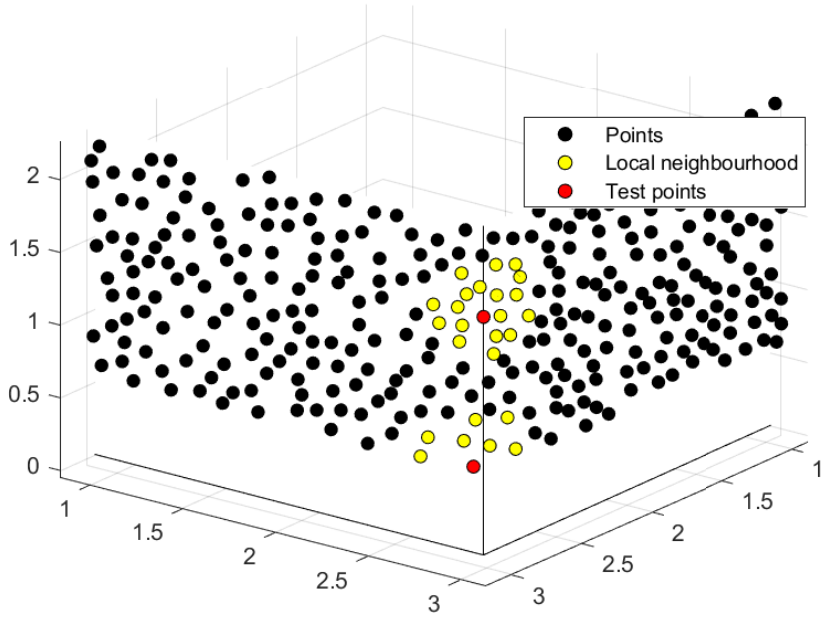


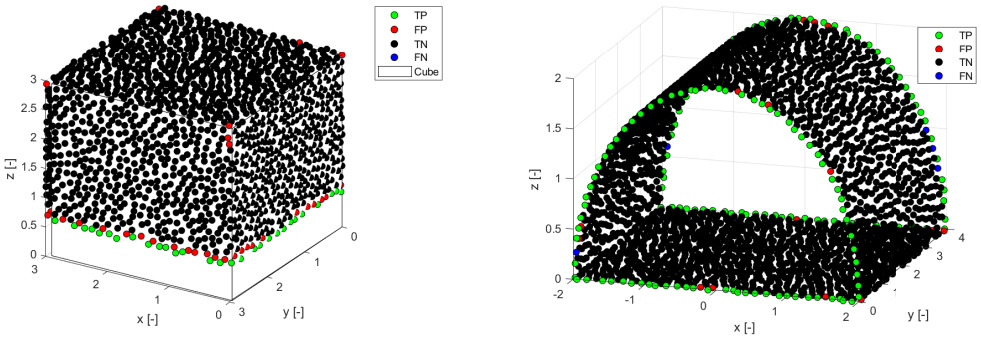
Figure 3.4: Example of Number of Points method. Frontier points have less points in LNR.

detecting edges it also falsely detects any edges on the frontier (see fig. 3.7c). This could be problematic when using the method in rough terrain.
The ideal values are strongly dependent on the terrain shape and method parameters, but for this case (chapter 3.1) values between 0.02 and 0.08 seem usable.

Projection on plane

As we can see on figure 3.2 the method itself works well on planes. We can use this knowledge, and classify points from a planar perspective. We do this by projecting the local neighbour points on a fitted plane and applying the mean vector method on those projected points (See example fig. 3.8). This removes any false positives occurring because of object geometry.

Being pretty much the same method we can expect the optimal threshold values to be similar to those of a plane. The ROC curves for the method are on figure 3.9, if we compare those with the graphs in figure 3.3 it's evident that the characteristics now more closely resemble those of a plane or sphere. Values are in table 3.3, and although slightly lower are mostly similar to those of a plane. On figure 3.10 the effects of the method can be seen. All of the "false edges" have been removed, without impairing the true frontiers.
While this method works great at removing false frontiers, this comes at a



(a) : cube **(b) :** tunnel

Figure 3.5: Effect of point number based FP detection. For $T_{points} = 19$

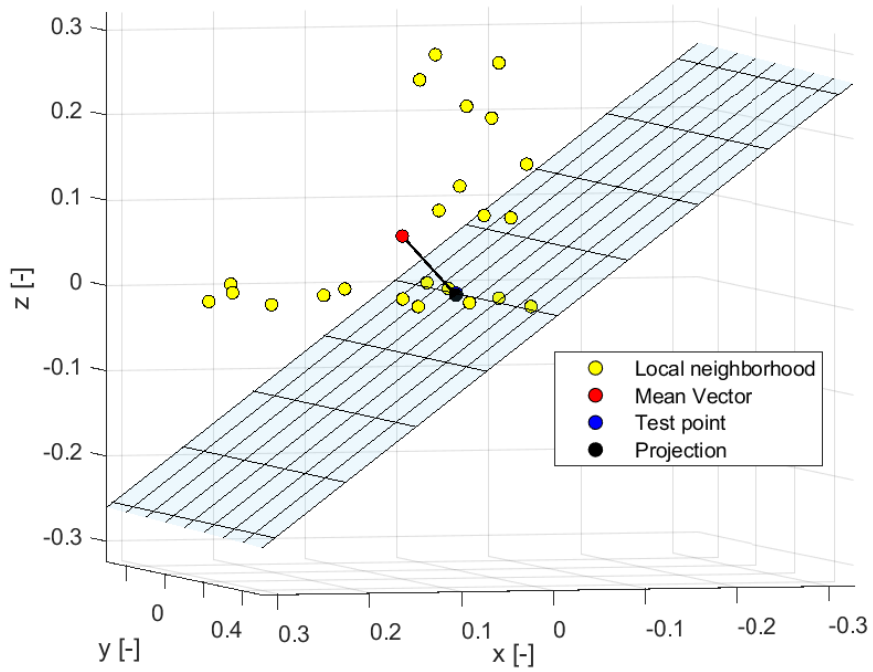
great cost in computational complexity and time. With the added need for plane fitting and point projection.

r	1.5	2.12	2.75	3.38	4.0
Cube	0.24	0.19	0.19	0.20	0.20
Tunnel	0.21	0.19	0.19	0.19	0.21

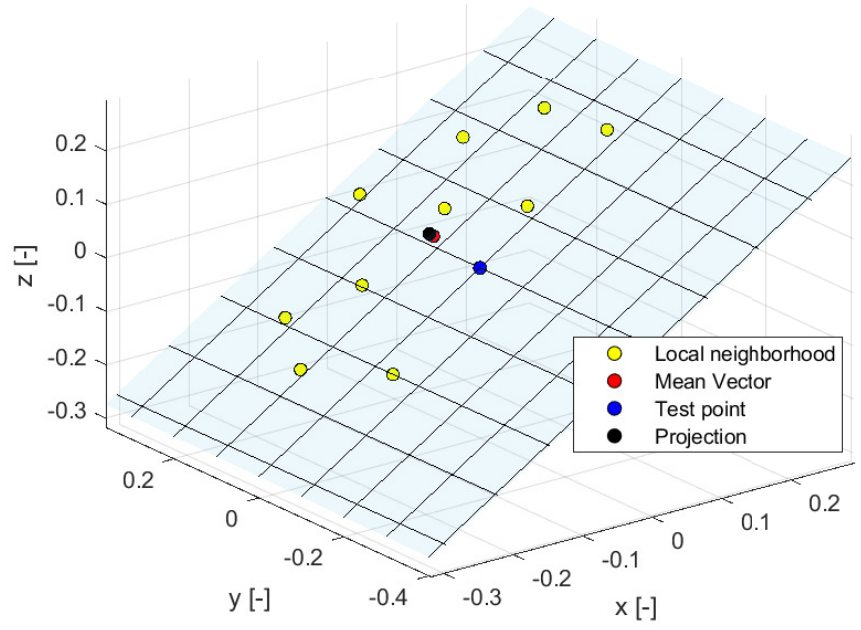
Table 3.3: Optimal threshold values for plane fitting method.

Comparison

Number of points method has the advantage of low computational requirements, with LNR points having already been found for the mean vector algorithm. Its disadvantage is the assumption that the point cloud has a uniform density. Also false detection for objects with forms that would naturally invoke sparser point clouds, for example rock spikes or grates.
Distance from fitted plane method, on the other hand, has generally a more thorough false positive detection. But this at computational cost and the possibility to misclassify frontier points. Although this missclassification only occurs on sharp edges, depending on threshold. This method seems to be better for locations with flat or curved surfaces.
The effectiveness of both methods is dependent on the explored environment terrain. The plane method has the best false positive detection rate, but this comes at the cost of speed and computation.

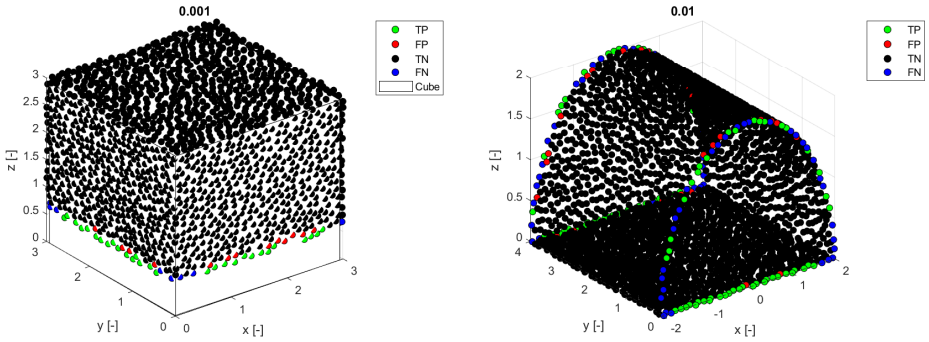


(a) : False frontier



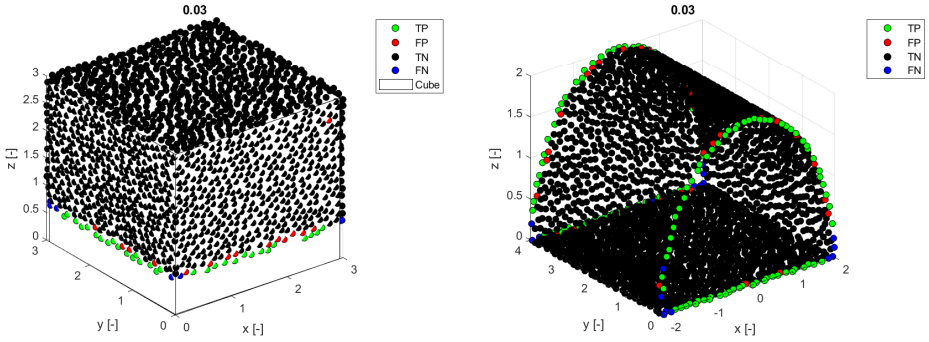
(b) : True frontier

Figure 3.6: Example of fitting method.



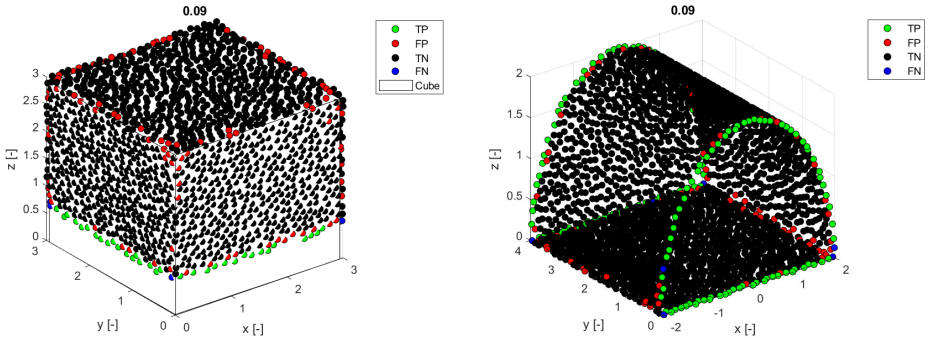
(a) : $T_f = 0.01$

(b) : $T_f = 0.01$



(c) : $T_f = 0.03$

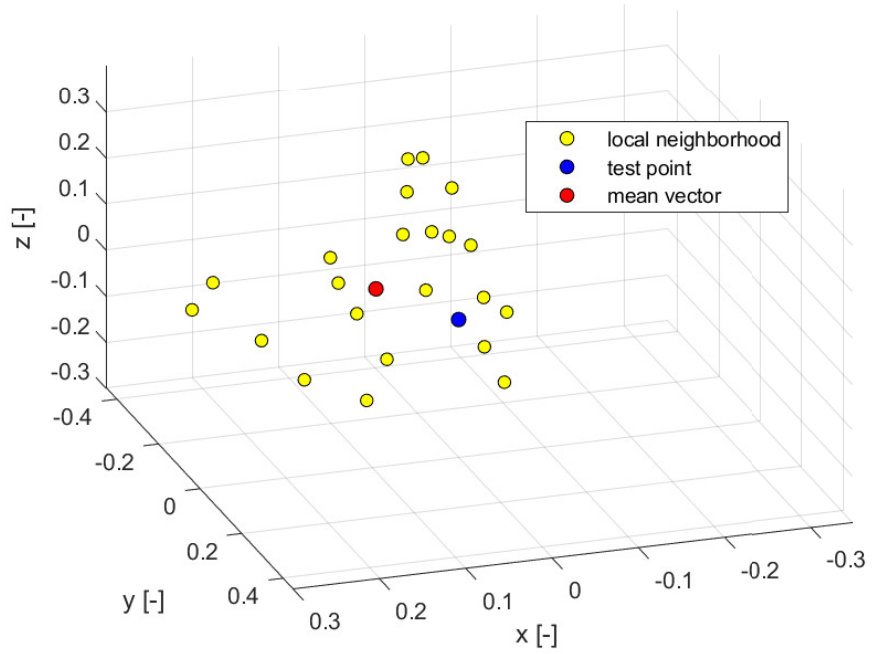
(d) : $T_f = 0.03$



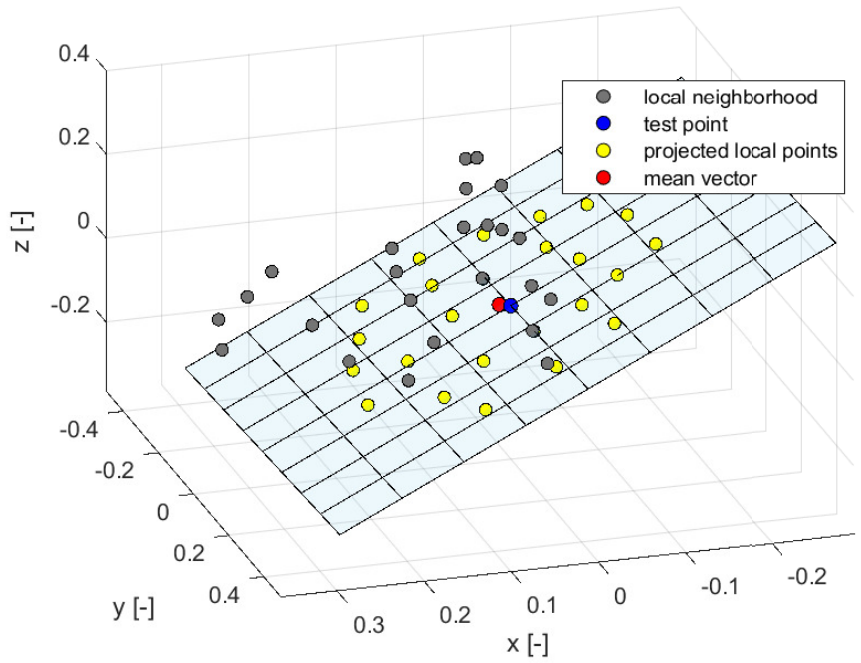
(e) : $T_f = 0.09$

(f) : $T_f = 0.09$

Figure 3.7: Example of effect of fitting method threshold (T_f) changes

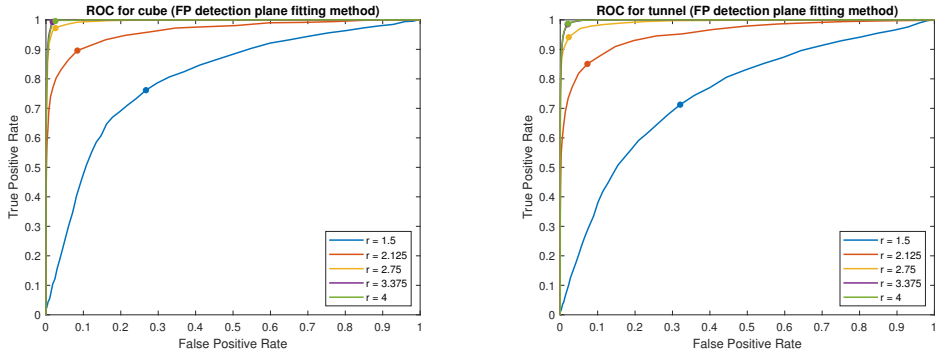


(a) : without projection



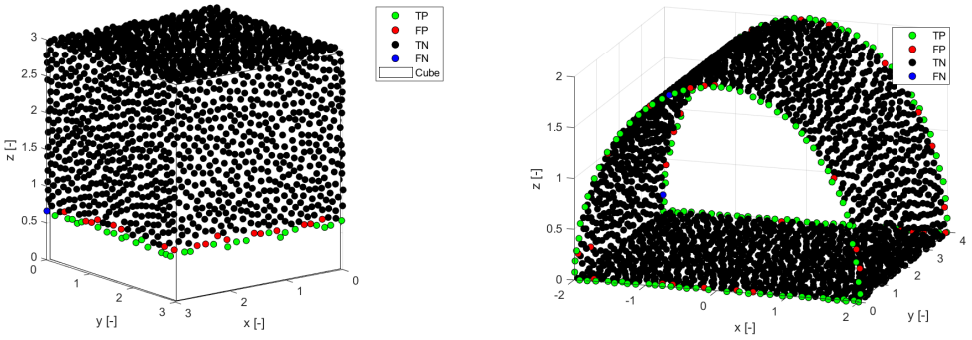
(b) : with projection

Figure 3.8: Example of plane fitting method.



(a) : cube **(b) :** tunnel

Figure 3.9: ROC curves for classification with plane fitting method. Values in table 3.3.



(a) : cube **(b) :** tunnel

Figure 3.10: Example of results with plane fitting method.

Chapter 4

Experiments in SubT simulator.

Testing will be done in the DARPA subterranean challenge (SubT) virtual competition simulator (see https://github.com/osrf/subt/wiki). This simulator provides a robot, sensor and environment that can be used for testing the method in sufficiently realistic scenario.

4.1 Test environment

The tests will be done on the virtual cave and urban environment. An example of simulation can be seen on figure 4.1.
For testing purposes, any kind of autonomous localization or exploration will be omitted. Instead, we will purely focus on the algorithms ability to identify suitable exploration goals on a sparse point cloud map. For this goal, the real time position of the robot from the simulator will be used. Though it is important to note that determining robot location is a whole problem in itself (localization). Also any disturbances or inaccuracies in localization or mapping will project itself on the point cloud map, which is going to have a negative impact on the frontier detection method.

Robot

The robot used for testing is a large skid-steer UGV. Its only sensor used in our experiments is the long range (30m) lidar.

4.1.1 Code structure

The simulator operates on the robot operating system (ROS), in which code is split into nodes, which interact by sending (publishing) and receiving

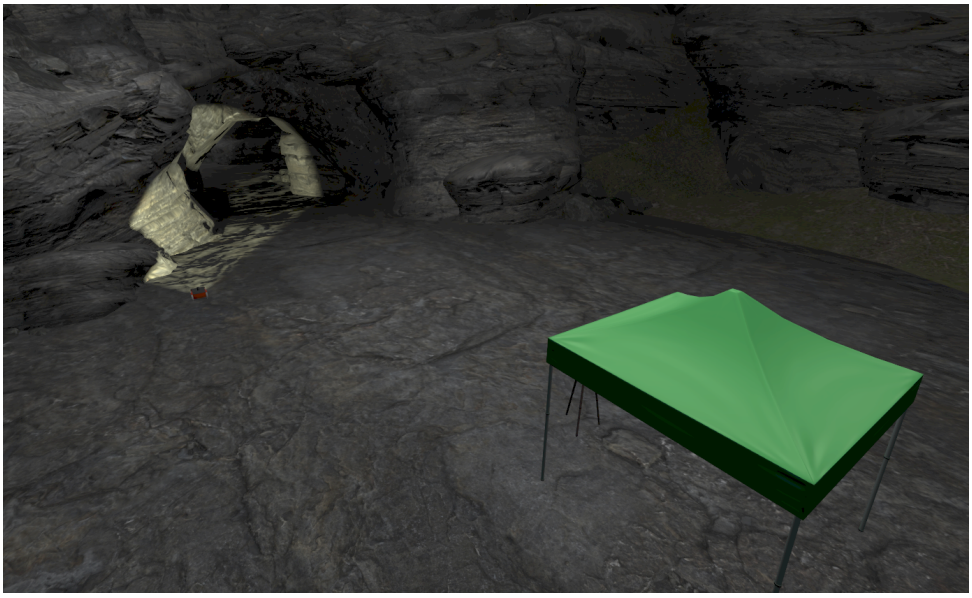


Figure 4.1: Example of the SubT virtual test environment.

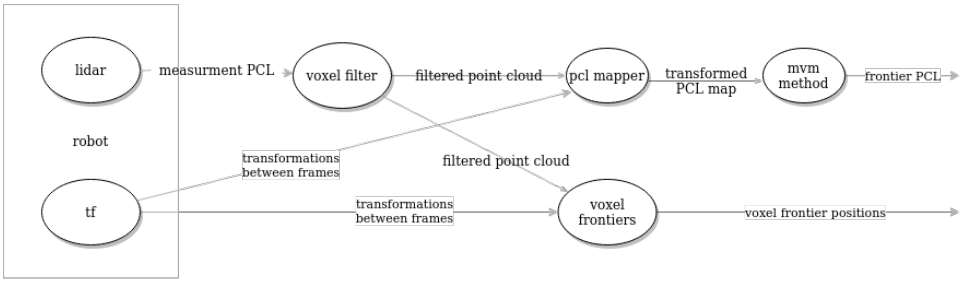


Figure 4.2: Code structure

(subscribing to) messages. This is demonstrated on figure 4.2 where the structure for our testing can be seen, as well as the robot providing the measurements and transformations between coordinate frames.

Voxel filter

This node subscribes to the robot lidar measurements and reduces the amount of measure points. This works by replacing all points in a voxel with their centroid. This reduces the amount of data, but keeps the relevant surface information.

Point cloud mapper

This node takes the filtered points, and transforms them from the sensor frame to the world frame. After this it creates a map by adding new points if

there are no points closer than a given minimal point distance. Thus creating a point cloud map of the environment with a minimal point distance.

Mean vector method

This node subscribes to the point cloud map and applies the mean vector method (as described in chapter 2.2). The parameters gained in chapter 3 give good results so there was no need to alter them. False positive filtering based on the number of points (see chapter 3.3.2) was added to get rid of object edges. Operations on point clouds are done using the C++ Point Cloud Library [3], with FLANN (Fast Library for Approximate Nearest Neighbours) [4] providing k-nearest neighbour search for finding points in local neighbourhood. It outputs the classified frontier points as a point cloud.

Voxel frontiers

This node takes the filtered point cloud from voxel filter, and crates an internal voxel map, on which frontier voxels are identified. Yamauchi desribes frontiers as "*regions on the border between space known to be open and unexplored space.*"[1] Because we are working on point clouds, which are measured points (i.e, occupied space), so if we are comparing those methods we have to consider full voxels. We thus define a voxel frontier as: *A full voxel which directly neighbours an empty voxel, which directly neighbours an unexplored voxel.* With a neighbouring voxel meaning the ones which touch a side (6-neighbourhood). The code implementation is based on a header only voxel map library[1]. The node outputs the centers of frontier voxels as a point cloud.

4.2 Results

4.2.1 Point cloud method

Figure 4.3 shows the results of mapping and frontier detection in a cave environment. As we see, the detection ability looks good, with most of the points closest to on opening being classified. But there are things to look out for, most of which we already encountered in the previous chapter. In figure 4.4a we see that even in a cave environment, which we could consider very round in contrast to urban areas, we can't do without some form of filtering out false positives. Also we have to remember specific filtering methods can have their own weaknesses, for example number of points method, which doesn't filter out long narrow objects, see gazebo legs in figure 4.4b.

[1] https://bitbucket.org/tpetricek/voxel_map/src/master/

4.2.2 Voxel method

A showcase of voxel mapping in a cave environment is on figure 4.5. There are a lot of false positives but frontiers around openings are detected. It seems that on curved surfaces there is a chance that the measuring line intersects a voxel on the outside of the map[2], thus classifying it as empty. Which from our definition (chapter 4.1.1) leads to the voxels around it being set as frontiers. So it seems the method has a hard time dealing with curved surfaces[3]. But as shown on figure 4.6 there are no problems on flat surfaces.

4.2.3 Method comparison

Discussion of pros and cons

The first thing noticeable when comparing the two methods is the simplicity of point cloud based mapping, with only the need for a point map, and nearest neighbour search. With the point cloud map already being the output of simultaneous localization and mapping (SLAM) methods, the implementation of the point cloud method seemed far easier.

The simplicity of the method also concerns the maintenance of the map, as adding points to a cloud is easier than having to constantly update occupancy values for voxel maps. Though in this sense the voxel map is more robust to faulty measurements. With the point cloud mapping algorithm in its current form faulty measurement (or transformations) would be permanently added to the map. Because of this, some form of a point matching algorithm should be used in mapping.

Single outliers aren't a problem for both methods, as part of voxel-based exploration is splitting frontiers into separate clusters, with the amount of points in a cluster predicting the possible reward of exploration (i.e, new unexplored space). Thus false classification in small enough numbers would be ignored, this is particularly beneficial for number of points FP detection, which does not detect thin edges.

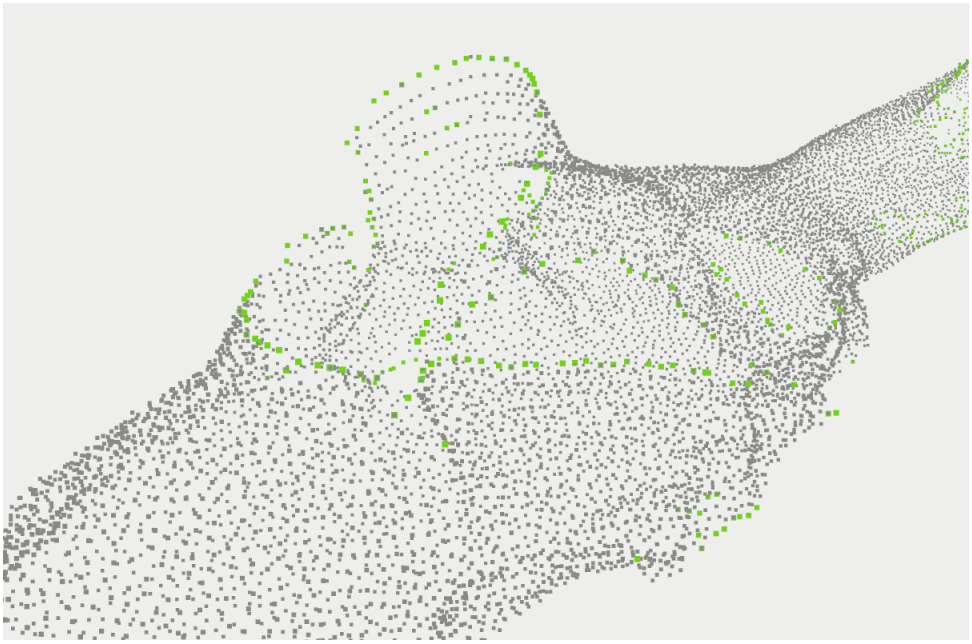
It would be interesting to see if sensor inaccuracies could be remedied by applying the plane projection method discussed in chapter 3.3.2, although the methods with plane fitting come at the cost of simplicity, and quickness, which are otherwise the methods greatest strengths.

At least for this abstracted implementation the frontier detection ability of the frontier algorithm seems good. But before any concrete statement can be made about the viability of the algorithm in autonomous exploration, more

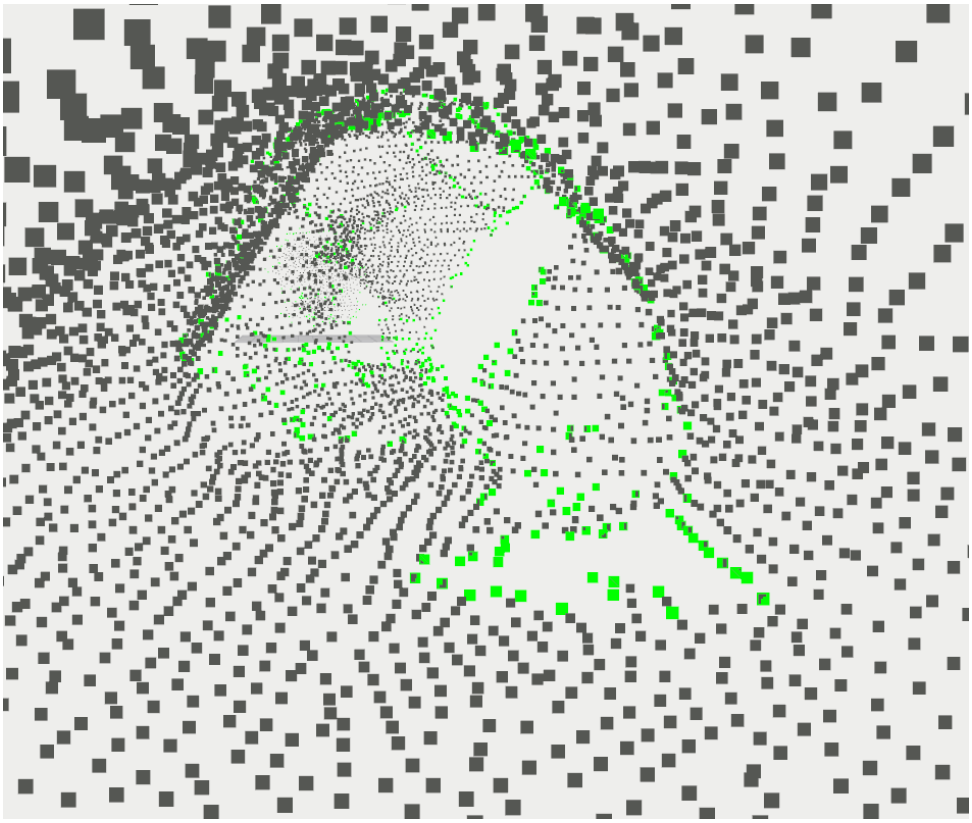
[2]Remember that for a voxel to count as free it is enough for the measuring line to intersect it at any point.

[3]Or at least my implementation of it does.

tests would need to be done with SLAM and autonomous navigation, based on frontier goals.

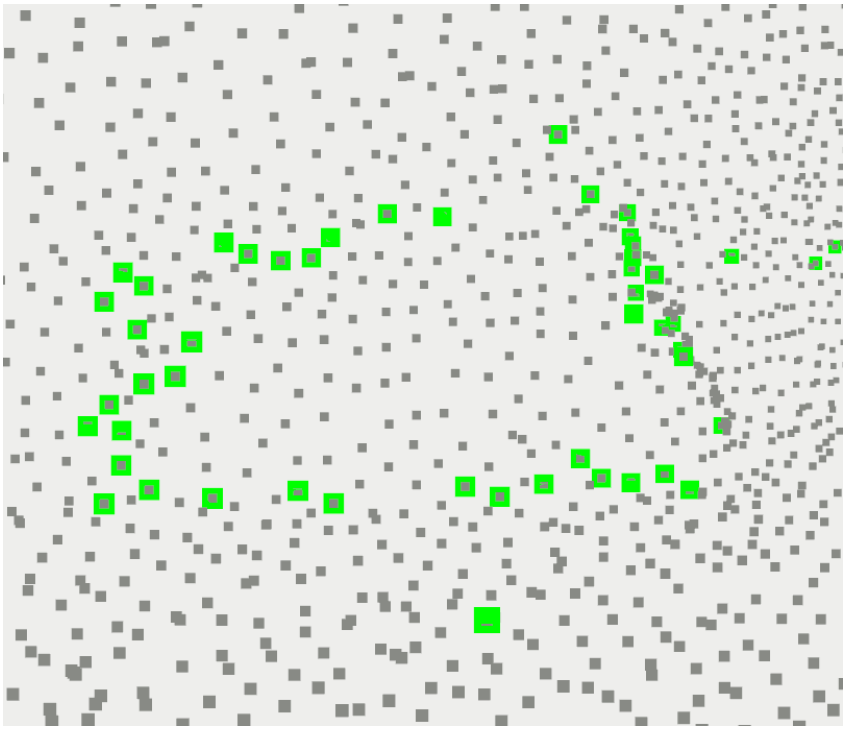


(a) : Outside view.

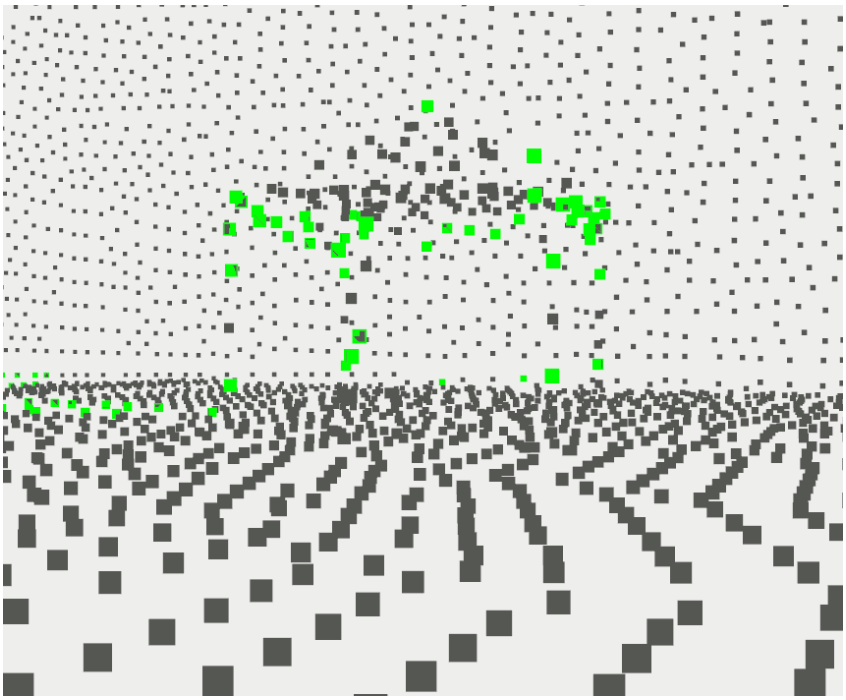


(b) : Inside view.

Figure 4.3: Showcase of point cloud mapping and frontier detection (green) in cave environment.



(a) : Falsely classified edges around a rockfall. (Without FP detection.)



(b) : False classification of object edges. (Number of points FP detection.)

Figure 4.4: False classifications of the mean vector method.

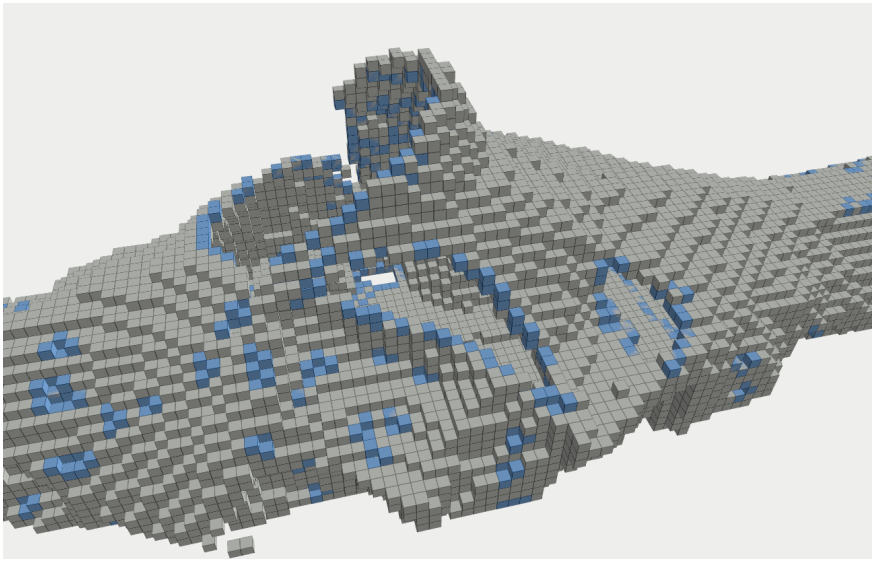


Figure 4.5: Showcase of voxel frontiers in cave environment

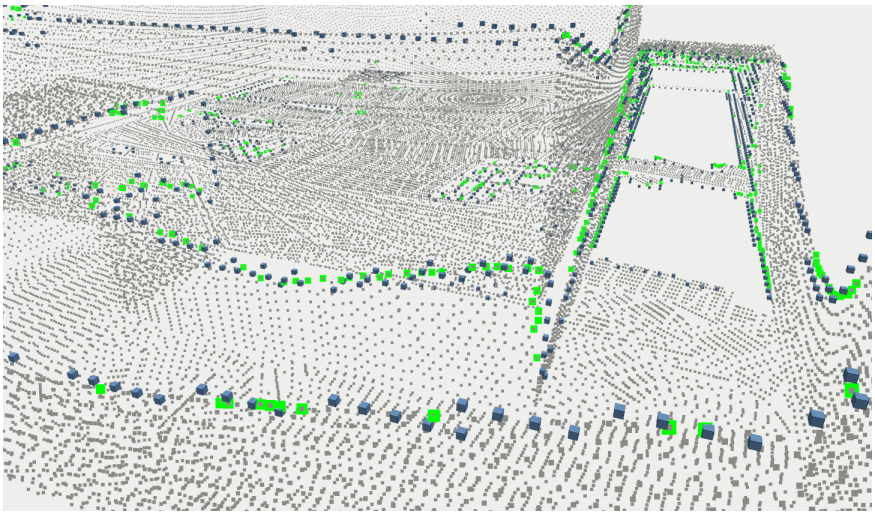


Figure 4.6: Comparison of frontier classification ability of mean vector method (green) and voxel mapping (blue), in an urban environment.

Chapter 5

Conclusion

In this paper we introduced the mean vector algorithm for detecting frontiers on point clouds. The algorithm function depends on the minimal point distance of the point cloud map, the local neighbourhood radius which determines the area around the test point, which is used for calculations, and the classification threshold. Through tests on simple objects, optimal parameters were determined. Radius in minimal distance units r (see eq. 3.1) should be kept between three and four. Threshold values are slightly dependent on the value of r, but for the expected values a threshold of 0.22 seems reasonable.

The tests showed that not only does the method detects edges between explored and unexplored space (frontiers) but also edges on object geometry, so we proposed and tested three methods for filtering out those false detections. The number of points method, which is simplest and fastest one but does not filter out long thin objects for example poles or rock spikes. The distance from fitted plane method, which worked by filtering out all object edges but this also included those that were on the frontier. The projection on plane method, which was the most reliable method of the three, filtering out all false detections without impairing true frontiers but this comes at the cost of added complexity and computation time.

Afterwards the mean vector algorithm was implemented in ROS and tested in a simulated underground environment. For these abstracted tests the frontier detection ability of the algorithm seems to be working well. But before any concrete statement can be made about the viability of the algorithm in autonomous exploration, more tests would need to be done with SLAM and autonomous navigation, based on frontier goals

Appendix A

Bibliography

[1] Yamauchi, Brian. "A frontier-based approach for autonomous exploration." Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'. IEEE, 1997.

[2] Youden, William J. "Index for rating diagnostic tests." Cancer 3.1 (1950): 32-35.

[3] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 1-4, doi: 10.1109/ICRA.2011.5980567.

[4] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in International Conference on Computer Vision Theory and Application VISSAPP'09). INSTICC Press, 2009, pp. 331-340.

[5] ——, "What are Point Clouds".https://tech27.com/resources/point-clouds/ (accessed august 12 2021)

[6] Hans Moravec and A. E. Elfes Conference Paper, Proceedings of the 1985 IEEE International Conference on Robotics and Automation, pp. 116 - 121, March, 1985

[7] Krüsi, P. et al. Drivingon Point Clouds:MotionPlanning,TrajectoryOptimization,and Terrain Assessmentin GenericNonplanarEnvironments.J. Field Robotics,34: 940-984,2017.

[8] Surmann, H., Nuechter, A., & Hertzberg, J. (2003). An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. Robotics and Autonomous Systems, 45(3–4), 181–198.

[9] D. Holz, N. Basilico, F. Amigoni and S. Behnke, "Evaluating the Efficiency of Frontier-based Exploration Strategies," ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), 2010, pp. 1-8.

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kahlert Lars** Personal ID number: **478066**

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

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Autonomous Robotic Exploration of Underground Environments

Bachelor's thesis title in Czech:

Autonomní robotická explorace podzemních prostor

Guidelines:

Consider the task of autonomous robotic exploration of underground environments.
Focus on identifying suitable exploration goals in a sparse point cloud map, as opposed to a voxel grid occupancy map, discuss related challenges and the pros and cons of such an approach.
Design, implement and evaluate a method for solving this task.
Perform experiments in the DARPA Subterranean Challenge simulator (see https://subtchallenge.world/, https://github.com/osrf/subt/).

Bibliography / sources:

[1] Rouček T. et al. DARPA Subterranean Challenge: Multi-robotic Exploration of Underground Environments. In: Modelling and Simulation for Autonomous Systems. Springer, Cham, 2020.
[2] Krüsi, P. et al. Driving on Point Clouds: Motion Planning, Trajectory Optimization, and Terrain Assessment in Generic Nonplanar Environments. J. Field Robotics, 34: 940-984, 2017.
[3] B. Yamauchi. A frontier-based approach for autonomous exploration. In: Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. Monterey, CA, USA, 1997.

Name and workplace of bachelor's thesis supervisor:

Ing. Tomáš Petříček, Ph.D., Vision for Robotics and Autonomous Systems, FEE

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

Date of bachelor's thesis assignment: **25.01.2021** Deadline for bachelor thesis submission: **13.08.2021**

Assignment valid until: **30.09.2022**

Ing. Tomáš Petříček, Ph.D. Supervisor's signature	prof. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature