Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

NDT SLAM Respecting Visibility

Matěj Boxan

Supervisor: Ing. Vladimír Smutný Ph.D.
Field of study: Cybernetics and Robotics
May 2020

Acknowledgements

I would like to express my gratitude to my supervisor Mr. Vladimír Smutný for his patience and persistent help during the whole work. I also appreciate many useful suggestion from his colleague, Mr. Pavel Krsek. Finally, I would like to thank my family and my girlfriend for their never-ending support during my studies.

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 the university theses.

Prague, 20. May 2020

Abstract

This thesis is dedicated to improvements of existing NDT (Normal distribution transform) SLAM algorithm. The import of static structures from CAD drawings of the building, which the original algorithm exploits, lead to duplication of certain walls. The proposed solution is the use of visibility algorithms, known from computer graphics. Positive consequence of integration of visibility is an overall speedup of the algorithm.

Another improvement of the previous implementation is the fusion of odometry data provided by the mobile robot, while taking into account their uncertainty. The motivation is to increase reliability and robustness mainly in long corridors.

The thesis contains a description of used methods, their implementation and evaluation of experimental results, especially in comparison to the original algorithm.

Keywords: localisation, mapping, mobile robot, lidar, SLAM, visibility, NDT, normal distribution transform, ROS, odometry, CAD drawing

Supervisor:
Ing. Vladimír Smutný Ph.D.

Abstrakt

Tato bakalářská práce se zabývá vylepšením existujícího algoritmu pro NDT (Normal distribution transform) SLAM. Import statických struktur z CAD výkresu budovy, které původní řešení využívá, mělo za následek duplikaci některých stěn. Jako řešení se nabízí algoritmy zkoumající viditelnost, známé z počítačové grafiky. Dalším důsledkem jejich integrace je časové zrychlení původního NDT SLAM algoritmu.

Dalším vylepšením původního řešení je větší zapojení dat z odometrie robota, s přihlédnutím k jejich nejistotě. Motivací je zvýšení spolehlivosti lokalizace a mapování, zejména v dlouhých chodbách.

Práce obsahuje popis principů použitých metod, jejich implementaci a zhodnocení výsledků provedených experimentů, především jejich srovnání s původním algoritmem.

Klíčová slova: lokalizace, mapování, mobilní robot, lidar, SLAM, viditelnost, NDT, transformace normálního rozdělení, ROS, odometrie, CAD výkres

Překlad názvu: Lokalizace mobilního robotu metodou NDT za využití viditelnosti

Contents

Figures

Tables

Chapter 1

Introduction

Simultaneous localisation and mapping is one of the basic underlying tasks of mobile robotics. While the robot needs to determine its position in the environment, at the same time it updates the map of the environment with data acquired from various sensors, such as LIDARs and video cameras. Both localisation and mapping are required for other crucial features, including navigation and multi-robot cooperation.

State of the art solutions using point cloud matching are memory consuming, especially in larger environments. A proposed solution to this is based on Normal distribution transform (NDT). The environment is divided into a grid, similar to the one used for occupancy map. Each cell in the grid contains a normal distribution function, representing the probability of position and size of an obstacle. Scan registration is then used to update the existing NDT grid, by computing a score based on the distances between dedicated cells and points obtained from the LIDAR sensor in the form of a point cloud.

In larger environments, looking for the optimal transformation between the point cloud and NDT grid of the whole map requires longer computation time, while the robot is carrying only limited resources which are needed elsewhere. It is thus suitable to exploit only information about the part of the map that the robot currently observes, or use other methods, such as ray-tracing or visibility algorithms known from computer graphics, to further decrease the number of possible transformations.

One of proposed uses for mobile robots are warehouses, where robots could be employed for transportation of goods. Dynamically changing environment of a warehouse can be well represented as an NDT map. Certain parts of the building in standard situation don't change, such as supporting frames. Because the majority of modern buildings were at some stage planned in the form of CAD drawings, these prior information could be used for creation of the base of the NDT map. This import is strongly influenced by parameters of the grid, particularly by the size of the cells. This may lead to duplication of certain walls, which causes problems when fitting a laser scan to the existing map, as the robot cannot see the wall facing in the opposite direction.

Majority of today's land mobile robots provide odometry data, either from wheel encoders, or estimated from properties of motors. Although the measurements are indirect and strongly influenced by the environment the robot moves in (such as slippery floor) and the state of the robot itself (pressure in wheels), odometry can still be beneficial, if appropriately combined with other methods of localisation. These methods and especially scan matching are rather unreliable in long corridors. For this reason, even though not explicitly mentioned in the assignment, the need for the use of odometry emerged during work on this theses.

These issues create a motivation to introduce visibility and odometry as a new part to the existing NDT algorithm.

Chapter 2

State of the art

2.1 Normal distribution transform

Normal distribution transform is an alternative approach for storing digital map data in a grid, firstly proposed by Biber and Straßer in [2]. Each cell is represented by its mean

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_x \\ \mu_y \\ \mu_\theta \end{bmatrix} \tag{2.1}$$

and covariance matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} & \Sigma_{x\theta} \\ \Sigma_{yx} & \Sigma_{yy} & \Sigma_{y\theta} \\ \Sigma_{\theta x} & \Sigma_{\theta y} & \Sigma_{\theta\theta} \end{bmatrix}, \tag{2.2}$$

that are built from each measurement $\{\mathbf{x_i}\}_{i=1}^{N}$ as

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x_i} \quad \text{and} \quad \boldsymbol{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x_i} - \boldsymbol{\mu})(\mathbf{x_i} - \boldsymbol{\mu})^T. \tag{2.3}$$

A normal distribution is therefore assigned to each cell in the grid, modelling the probability of measuring a point obtained from a laser scan. Those can be easily calculated, are continuous and differentiable and can be further used to match another laser scan into the existing map.

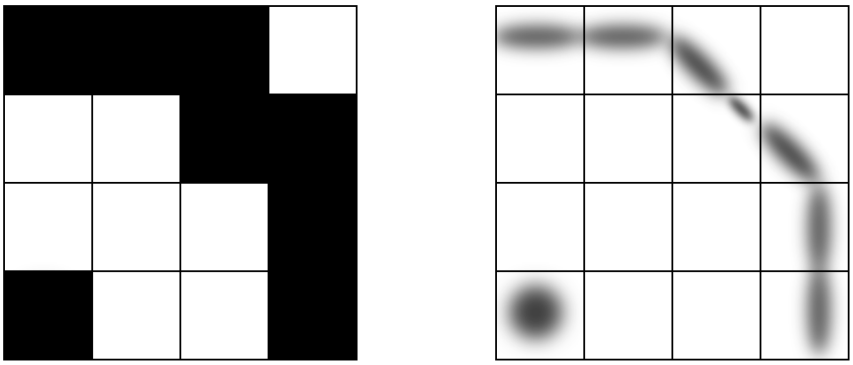
The probability of measuring a specific point x in a cell c is modelled by the normal distribution

$$p_c(\mathbf{x}) \sim exp(-\frac{(\mathbf{x} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})}{2}). \tag{2.4}$$

Unlike the standard occupancy grid, which represents information whether or not a cell is being occupied, NDT provides information describing the

probability of measuring a laser scan sample for each position inside of the cell. Using this method, we can work with more precise information about obstacle position, orientation and size, as the probability distribution of obstacles is normal and not uniform. Unlike storing point clouds, the memory consumption is constant over time.

Biber and Straßer proposed usage of 4 overlapping grids, in order to minimise effects of discretization. Similar approach was chosen by Schulz, Hanten and Zel in [12], who opted for 4 overlapping grids in 2D space and 8 overlapping grids in 3D space. The grids are shifted by half the resolution into different directions. The likelihood of a point $\mathbf{x}$ is then obtained as the mean of the four likelihood $p_c(\mathbf{x})$ values of the corresponding overlapping cell grids.



(a) : Occupancy grid **(b) :** NDT grid

Figure 2.1: Comparison of Occupancy and NDT grids

In dynamic environments, with people and other robots moving around, it is important to distinguish between free and occupied space, especially for planning algorithms. Normal distribution transform can be expanded to the Normal distribution transform occupancy map (NDT-OM) which explicitly describes information about the occupancy state, monitors free space and changes in occupation of dynamic objects. This extension described in [11] has been implemented by Jelínek [6]. The NDT-OM performs ray tracing between the sensor origin and the point of the ray's reflection from an obstacle. Cells that are intersected by the cast ray are updated with low occupancy probability and their covariance is changed accordingly, while the last cell is updated as occupied. Similar approach was proposed by Einhorn and Gross in [5], where they explicitly store the occupancy value of cell c as o_c and the final probability distribution expressing that a point belongs to c is $o_c \cdot \mathcal{N}(\boldsymbol{\mu_c}, \boldsymbol{\Sigma_c})$. Schulz, Hanten and Zel proposed alternative solution called kd-ONDT, where the ray casting is performed between the LIDAR position and the means of the measurement distributions. Cells among the ray are once again updated free and then the measurement distributions are combined with existing distributions in the last cell. Unlike the NDT-OM, maximum likelihood values of the ray in the cells to be updated is not covered, to ensure independence of the new measurement from the previous ones.

Biber and Straßer described the conversion between a recorded point cloud and the NDT grid [2]. This conversion is called PCL-to-NDT matching and it has been implemented and brought to our environment by Nováček [8], who describes it in greater detail. Other possible scenario of scan matching is PCL-to-PCL, standing for Point cloud to Point cloud. This is a standard technique for scan matching, solved by the Iterative closest point (ICP) algorithm [7]. The ICP algorithm aims to find the best possible transformation between two laser scans, in order to minimise their distance. The last possible setting is NDT-to-NDT matching, which can be once again interpreted as minimising the distance between two NDT grids.

2.2 Score function

The goal of a scan alignment is to recover parameters $\mathbf{p} = [t_x, t_y, \theta]^T$ of transformation T between two robot coordinate frames. The transformation T in 2D is

$$T(\mathbf{p}, \mathbf{x}) = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \mathbf{x} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \tag{2.5}$$

where t_x and t_y are translation parameters and θ is the angle of rotation. Optimising the parameters $\mathbf{p}$ is our next focus. Firstly, each point from the measurement $\{\mathbf{x_i}\}_{i=1}^{N}$ is transformed with the transform $T(\mathbf{p}, \mathbf{x})$, that is $\mathbf{x'_i} = T(\mathbf{p}, \mathbf{x_i})$. Then, the covariance matrix $\mathbf{\Sigma_i}$ and mean $\boldsymbol{\mu_i}$ are looked up in the NDT grid from a cell corresponding to the point $\mathbf{x_i}'$. Score $s(\mathbf{p})$ is computed as

$$s(\mathbf{p}) = \sum_i exp(-\frac{1}{2}(\mathbf{x'_i} - \boldsymbol{\mu_i})^T \mathbf{\Sigma_i^{-1}} (\mathbf{x'_i} - \boldsymbol{\mu_i})). \tag{2.6}$$

The parameters $\mathbf{p}$ are considered optimal if the value of sum $s(\mathbf{p})$ is maximal. Score $s(\mathbf{p})$, respectively its negation $-s(\mathbf{p})$, represents the optimisation criterion for Newton's algorithm, which iteratively finds parameters $\mathbf{p}$ that minimise $-s(\mathbf{p})$ by solving the equation

$$\mathcal{H}\Delta\mathbf{p} = -\mathbf{g}. \tag{2.7}$$

Details about the hessian matrix $\mathcal{H}$, gradient $\mathbf{g}$ and the derivatives of $s(\mathbf{p})$ can be found in the work of Nováček [8]. With respect to the existing literature, the term score function will be further used for the score $s(\mathbf{p})$.

2.3 Visibility

Visibility computation is an important part of computer graphics theory. Nevertheless, it is also discussed in other fields, such as telecommunications, architecture, computer vision and robotics. The aim of visibility algorithms

is to determine which objects and part of the scene are visible from the current location of the observer. Bittner and Wonka [3] provide a summary of existing visibility algorithms from the point of computer graphics based on the problem domain. We will further focus only on 2D algorithms, as the studied problem is also two dimensional.

There are many domains of visibility that computer graphics deals with. Among visibility along a line, visibility from a point, visibility from a polygon and others, only point to point visibility concerns us, because it is the closest approach to our own problem. Starting with the robot's location in the grid, maximum range and angle of vision, the task is to find all cells visible between the robot's position and the sector of the circle sensors on the robot can observe. In order to determine which cells are visible, a line between the start and final points needs to be drawn.

2.3.1 Bresenham's line algorithm

Bresenham's line drawing algorithm is a Digital differential algorithm (DDA) which approximates line between two points. As it only uses addition, subtraction and bit shifting of integers, it is widely used in today's processing units. The use of incremental error avoids the need for floating points, rounding and division operation [4]. Although it was originally intended for drawing lines on pixel screens, it can be as well used for 2D grids. Only a shortened version of the algorithm for slope between 0 and 1 is presented in Algorithm 1. It can be easily generalised to other octants.

Algorithm 1: Bresenham's line algorithm

```
1  function drawLine (x1, y1, x2, y2)
   Input  : Coordinates of two 2D points
2  dx = x2 - x1
3  dy = y2 - y1
4  step = 1
5  x = x1
6  y = y1
7  error = dx
8  for i from 0 to dx do
9  |  x += step
10 |  error += 2*dy
11 |  drawPoint(x, y)
12 |  if error > 2*dx then
13 |  |  y += step
14 |  |  error -= 2*dx
15 |  end
16 end
```

From the point of computer graphics, any line drawing function must reasonably approximate a line on a limited resolution of a computer screen, because only horizontal, vertical and diagonal lines with slope equal to ±1 can be drawn precisely. Secondly, it must be fast [1]. Both these characteristics, which are met by the Bresenham's line drawing algorithm, are also required in our environment.

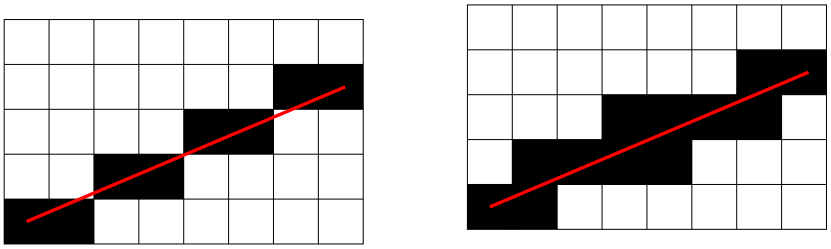
2.3.2 Supercover DDA line algorithm

A slight modification of the Bresenham's line algorithm, which returns all grid points crossed by a line, may be useful in cases when we need to determine occurrence of an obstacle between two points, such as robots location and the edge of its field of vision. The fact that the Bresenham's line algorithm is defined on 8-neighbourhood limits its ability to detect obstacles, especially in situations where the wall is not continuous, or in corners. Once again, only shortened version of the algorithm for the first octant is included in Algorithm 2.

Algorithm 2: Supercover DDA line algorithm

```
1  function drawLine x1, y1, x2, y2
   Input  : Coordinates of two 2D points
2  dx = x2 - x1
3  dy = y2 - y1
4  step = 1
5  x = x1
6  y = y1
7  error = dx
8  errorPrev = 0
9  for i from 0 to dx do
10 |  x += step
11 |  error += 2*dy
12 |  drawPoint(x, y)
13 |  if error > 2*dx then
14 |  |  y += step
15 |  |  error -= 2*dx;
16 |  |  if error + errorPrev < 2*dx then drawPoint(x-step, y) ;
17 |  |  else if error + errorPrev > 2*dx then drawPoint(x, y-step) ;
18 |  |  else drawPoint(x-step, y) drawPoint(x, y-step) ;
19 |  end
20 |  errorPrev = error
21 end
```

Where the Bresenham's line algorithm doesn't return all points the line crosses, conditions on lines 16-18 of Algorithm 2 check the adjacent cells and the result covers all the crossed points. This is demonstrated on Fig. 2.2.



(a) : Bresenham's line algorithm

(b) : Supercover Bresenham's line algorithm

Figure 2.2: Comparison of DDA algorithms

2.3.3 Midpoint circle algorithm

Midpoint circle algorithm enables rasterization of a circle to a discrete grid. It works similarly as the previous algorithms, that implies that it uses only addition, subtraction and bit shifting of integers. It draws 8 octants at once starting from the multiples of 90° and traverses both direction, until it reaches a multiple of 45°. The algorithm can be further adjusted to supercover all crossed pixels, or to draw only a sector of a circle.

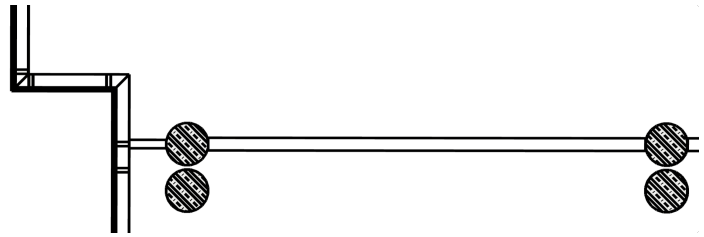
Algorithm 3: Midpoint circle algorithm

```
1  function drawCircle cx, cy, radius
   Input  : Coordinates of the centre of the circle, radius
2  r = radius;
3  x = -r;
4  y = 0;
5  error = 2-2*r;
6  do
7      Point(cx-x, cy+y);
8      Point(cx-y, cy-x);
9      Point(cx+x, cy-y);
10     Point(cx+y, cy+x);
11     r = error;
12     if r <= y then
13         y+=1;
14         error += 2*y+1;
15     end
16     if r > x or error > y then
17         x+=1;
18         error += x*2+1;
19     end
20 while x < 0;
```

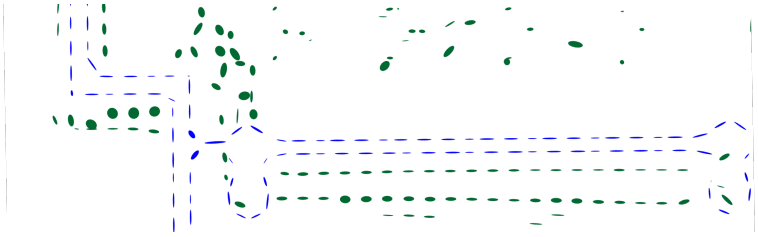
2.4 Our motivation

The motivation for the use of visibility algorithms is twofold. Firstly, fitting the laser scan only on the part of the world the robot can actually observe with its sensors speeds up scan matching. This increase of speed can then lead to more SLAM cycles per second and therefore to more accurate localisation and mapping.

The second reason is associated with the use of CAD drawings. The transform of CAD drawing to NDT map may lead to duplication of certain walls. An example of this can be seen on Figures 2.3. The original CAD drawing 2.3a is transformed to NDT grid (blue ellipses on 2.3b). After several rounds of NDT SLAM, where the robot is moving in the room north of the bottom wall, the bottom wall is duplicated (green ellipses). The cause of this improper behaviour is that the laser scan is firstly fit on the outer part of the wall, which leads to incorrect determination of the robot's position. When the robot observes the wall on the other side of the room, recorded laser scan is not fitted to this wall, which is now too far. Instead, the laser scan is registered into empty cells in front of the wall, which is now duplicated. Further laser scans are fitted to this duplicate, in effect returning wrong robot's position. When the robot once again observes the wall from Fig. 2.3, it's position is misaligned and the laser scan, previously fitted to the outer side of the wall, is registered to empty cells on the outer side of the wall, effectively creating a new wall.



(a) : Original CAD drawing



(b) : The result of NDT SLAM after several rounds in a room.

Figure 2.3: Duplication of walls imported from CAD drawing

Chapter 3

Score function approximation and the use of odometry

The implementation by Nováček [8] performs one step of SLAM after the robot drives a minimum distance, or rotates a minimum angle. Otherwise it uses the odometry pose, acquired by the robot. However, in certain situations, the Newton's algorithm using the score function $s(\mathbf{p})$ as criterion does not converge, or converges to one of many local maximums. In cases like this, it would be effective to use a prior odometry information about the robot's pose to improve the performance and correctness of the algorithm. At the same time, if the score function $s(\mathbf{p})$ with parameters $\mathbf{p}$ from the final iteration of the Newton's algorithm has high values and the robot's position is thus quite likely, we would like to keep this information for future use. The final parameters $\mathbf{p}$ will be further referred to as $\mathbf{p_f}$.

As the score function $s(\mathbf{p_f})$ from 2.6 is not given analytically and its computation is time consuming, we will try to approximate it with a function similar to the multivariate normal distribution

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}}{\sqrt{|2\pi\boldsymbol{\Sigma}|}}. \tag{3.1}$$

The desired function $M(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, that approximates the score function $s(\mathbf{p})$, has several constrains

$$M(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = k\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{3.2}$$
$$M(\mathbf{p_f}, \boldsymbol{\Sigma}) = s(\mathbf{p_f}), \tag{3.3}$$
$$max(M(\boldsymbol{\mu}, \boldsymbol{\Sigma})) = max(s(\mathbf{p})), \tag{3.4}$$
$$\mathcal{H}(M(\boldsymbol{\mu}, \boldsymbol{\Sigma}))|_{\mathbf{p_f}} = \mathcal{H}(s(\mathbf{p})|_{\mathbf{p_f}}). \tag{3.5}$$

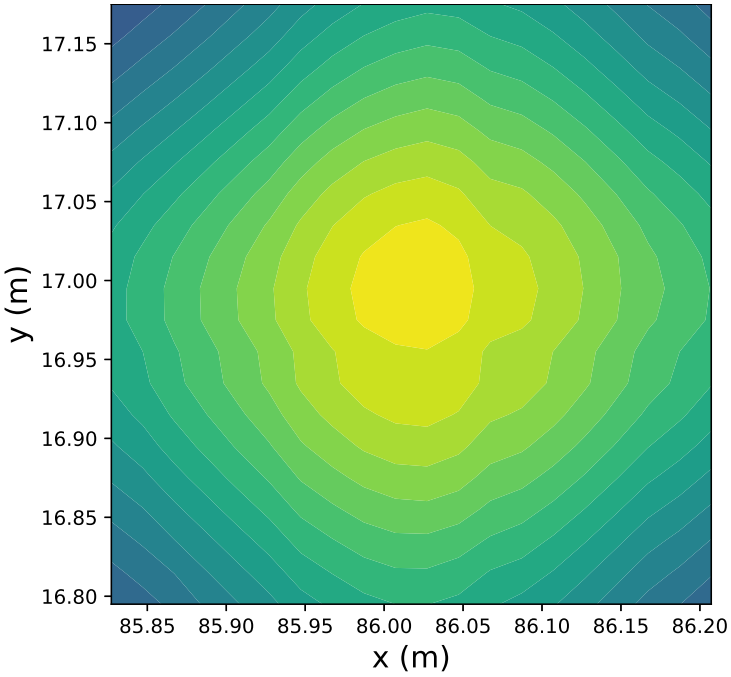


Figure 3.1: An example of planar cut through $s(\mathbf{p})$ in $\mathbf{p_f}$ along axis X, Y

The first constrain 3.2 underlines the similarity to the multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where k is a constant. The second 3.3 and third 3.4 constraints together define the relation between the maximal values of functions $M(\mathbf{x}, \boldsymbol{\Sigma})$ and $s(\mathbf{p})$. As $M(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is similar to a multivariate normal distribution, it has maximum in its mean $\boldsymbol{\mu}$. Subsequently, the score function $s(\mathbf{p})$ reaches it's highest point in $\mathbf{p_f}$, thus

$$max(s(\mathbf{p})) = s(\mathbf{p_f}) = M(\boldsymbol{\mu}, \boldsymbol{\Sigma})|_{\boldsymbol{\mu}} \quad \text{and} \quad \mathbf{p_f} = \boldsymbol{\mu}. \tag{3.6}$$

Finally, 3.5 implies that $M(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ approximates $s(\mathbf{p})$ on its neighbourhood, because the hessian matrices $\mathcal{H}$ of their second derivatives are equal.

By definition, the covariance matrix of a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ must be symmetrical and positive-definitive. Since the score function converges to maximums and it's hessians in these points are therefore negative definitive, we can obtain the covariance matrix as

$$\boldsymbol{\Sigma} = -\mathcal{H}^{-1}. \tag{3.7}$$

From 3.5 and 3.2 we get

$$\begin{aligned} \mathcal{H}(M(\mathbf{x}, \boldsymbol{\Sigma}))|_{\boldsymbol{\mu}} &= \mathcal{H}(k\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})) \\ &= k\mathcal{H}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})) \\ &= k\frac{\boldsymbol{\Sigma}^{-1}}{\sqrt{|2\pi\boldsymbol{\Sigma}|}} \\ &= \mathcal{H}(s(\mathbf{p}))|_{\boldsymbol{\mu}}, \end{aligned} \tag{3.8}$$

describing the approximation of the score function $s(\mathbf{p})$ by the multivariate normal distribution. From 3.2 we get the relation between the score function $s(\mathbf{p})$ and its approximation in $\mathbf{p_f} = \boldsymbol{\mu}$ as

$$\begin{aligned} M(\boldsymbol{\mu}, \boldsymbol{\Sigma})|_{\boldsymbol{\mu}} &= k\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})|_{\boldsymbol{\mu}} \\ &= k\frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}|}} \\ &= s(\mathbf{p}))_{|\boldsymbol{\mu}}. \end{aligned} \tag{3.9}$$

When we divide 3.8 by 3.9, we get the formula for the inverse of the covariance matrix

$$\boldsymbol{\Sigma}^{-1} = \frac{\mathcal{H}(s(\mathbf{p}))|_{\boldsymbol{\mu}})}{s(\mathbf{p}))|_{\boldsymbol{\mu}}}, \tag{3.10}$$

leading to the unknown value k being

$$k = s(\mathbf{p}))|_{\boldsymbol{\mu}}\sqrt{|2\pi\boldsymbol{\Sigma}|}. \tag{3.11}$$

Our desired function $M(\mathbf{x}, \boldsymbol{\Sigma})$, further referred as score approximation, has therefore the following formula

$$\begin{aligned} M(\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= k\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ &= s(\mathbf{p})|)_{\boldsymbol{\mu}}\sqrt{|2\pi\boldsymbol{\Sigma}|})\frac{e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}}{\sqrt{|2\pi\boldsymbol{\Sigma}|}} \\ &= s(\mathbf{p}))|_{\boldsymbol{\mu}}e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}. \end{aligned} \tag{3.12}$$

During this computation, we abandoned the constraint forcing the covariance matrix to be positive-definitive. This creates further problems, as the Newton's algorithm converges to maximums. With the approximation $M(\mathbf{x}, \boldsymbol{\Sigma})|_{\boldsymbol{\mu}}$ being a saddle point or a minimum, the algorithm would fail. Because of this, the determinant of the criterion is checked beforehand. If $|\mathcal{H}(M(\boldsymbol{\mu}, \boldsymbol{\Sigma}) + s(\boldsymbol{\mu}))| \geqq 0$, the estimation of the robot's pose from odometry is returned directly and the covariance $\boldsymbol{\Sigma}_{t-1}$ from the previous iteration is kept. To include the decrease of the accuracy of odometry over time, the covariance $\boldsymbol{\Sigma}_{t-1}$ is multiplied by a matrix $\mathbf{C}$, so that $\boldsymbol{\Sigma}_t = \mathbf{C} \cdot \boldsymbol{\Sigma}_{t-1}$. The fact that the hessian $\mathcal{H}(s(\boldsymbol{\mu}))$ is positive-semidefinitive corresponds to the situation where there is not sufficient match between the laser scan and the map.

The adjusted localisation algorithm in time t follows these steps:

1. Initialise the transformation $\mathbf{p} = [t_x, t_y, t_\theta]^T$ with the odometry position estimation $\mathbf{e}_t = [e_x, e_y, e_\theta]^T$.
2. Initialise the Score approximation $M(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\mu} = \mathbf{p}$ and $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{t-1}$ from the previous iteration.
3. Transform every point from the laser scan with the transformation $T(\mathbf{p_i})$ as described in 2.2.
4. Compute the value of the Score function $s(\mathbf{p_i})|_t$ as described in 2.2.
5. Optimise $\mathbf{p}$ with Newton's method described in Chapter 3 in [8] using the sum $M(\mathbf{p_i}, \boldsymbol{\Sigma})|_t + s(\mathbf{p_i})|_t$ as criterion.
6. Find a new transformation $T(\mathbf{p_{i+1}})$ and update the mean $\boldsymbol{\mu}$ of score approximation $M(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, such that $\boldsymbol{\mu} = \mathbf{p_{i+1}}$.
7. Go to step 3. The algorithm ends when the maximum number of iterations is reached, or when the absolute value of the difference between the value of the sum $M(\mathbf{p_i}, \boldsymbol{\Sigma})|_t + s(\mathbf{p_i})|_t$ in the iteration i and $i+1$ is smaller than the threshold.
8. Store the covariance matrix $\boldsymbol{\Sigma}$ for score approximation in the next cycle. If the determinant $|\mathcal{H}(M(\mathbf{p_i}, \boldsymbol{\Sigma})|_t + s(\mathbf{p_i})|_t)| \geqq 0$, set $\boldsymbol{\Sigma}_t = \mathbf{C} \cdot \boldsymbol{\Sigma}_{t-1}$. Otherwise, set $\boldsymbol{\Sigma}_t = -\mathcal{H}(M(\mathbf{p_i}, \boldsymbol{\Sigma})|_t + s(\mathbf{p_i})|_t)$.

Chapter 4

Implementation

Sections of code, dealing with both visibility and odometry, were added to the ROS node `ndt_ciirc_slam`, implemented by Nováček, who built upon the work of Jelínek. Further changes were made to the code, mainly to respect the concepts of Object oriented programming (OOP). In this chapter, we firstly briefly introduce the Robot Operating System (ROS) and other software used in the project. Then the implemented classes and methods are described. Finally, the third section details Python scripts developed for visualisation of the score function and its approximation.

4.1 Used software

The Robot Operating System (ROS) is a framework designed for robotics software development. A ROS project usually contains several **Nodes**, that are independent processes performing computation and communicating between each other using data-structured **Messages**. Messages are published to **Topics**, following the publish-subscribe pattern. Synchronous communication between nodes is supported with **Services**. All connectivity in ROS is peer-to-peer. Larger systems of nodes can be organised into **Packages**. A visualisation part of ROS is called RViz, exploiting the possibility to subscribe to any topic and therefore visualise any message sent on the system [10].

The ROS node `ndt_ciirc_slam` performs NDT SLAM on odometry and laser scan data. It uses Point Cloud Library (PCL), a ROS library providing interface for point clouds, such as registration, filtering, segmentation and other manipulation with point clouds. The second used library is Eigen, C++ library for linear algebra. In the project, it is used for manipulation of vectors and matrices and for numerical solving of registration algorithms.

NDT maps can be imported to `ndt_ciirc_slam` in a form of `.txt` files. Each line contains the mean, covariance matrix, occupancy value and number of points, separated by commas. The conversion from `.dxf` CAD drawing to NDT map was implemented by Pánek [9].

4.2 Program overview

The *NdtSlamNode* subscribes to pairs of odometry and point cloud messages. Once a message arrives, the odometry is used as a guessed position for visibility computation in class *GridVisibility*. The returned *NdtGrid* is then used in PCL-to-NDT registration class *P2DRegistration*. It is in this class where the actual odometry information, combined with the approximation of the score function from the previous iteration held in class *ScoreApproximation*, is used in the optimisation cycle for scan matching.

The output of scan matching defines the transformation between the point cloud and the NDT map. With this information, the scan can be registered into the map via ray tracing. Before this step, one more run of Visibility algorithm is needed in order to prevent registration to grid cells that cannot be visible from the robot's position, e.g. are on the opposite side of a wall.

Outputs of the algorithm are position and orientation of the robot and map of the environment. Map data are carried in *nav_msgs::OccupancyGrid* messages for occupancy grid representation and *visualization_msgs::MarkerArray* for grid comprised of normal distributions, published for visualisation purposes in RViz. The pose is in the standard form of *geometry_msgs::PoseStamped*. Messages are published to topics that are further specified in ROS launch files that are part of the project. Program overview diagram is shown on Fig. 4.1.

4.2.1 AlgorithmSlam

The *AlgorithmSlam* class is responsible for computation of SLAM. It is described in detail in [8], here we emphasise only changes caused by object oriented programming approach. Two classes inhered from *AlgorithmSlam* are *AlgorithmNdtSlam*, which is the original class, and *AlgorithmVisibilityNdtSlam*, which includes support for visibility computation. Newly implemented method *save_pose_to_file*, conditioned by the parameter *save_trajectory_* (boolean), enables export of pose estimation from odometry and the output of localisation algorithm for their subsequent comparison in the form of text file, where each line contains one pose.

4.2.2 GridVisibility

GridVisibility is an abstract class with two descendants, *GridVisibilityBresenham* and *GridVisibilityRayTracing*. These classes implement public methods *getVisibility*, *transformCircle* and *visualise*. The visibility circle, with a given radius and angle of the sector of a circle, is created in the method *initVisibilityCircle* with a supercover Midpoint circle algorithm described in 2.3.3, called by the constructor. The generated circle is then transformed between individual visibility computation in the method *transformCircle*.

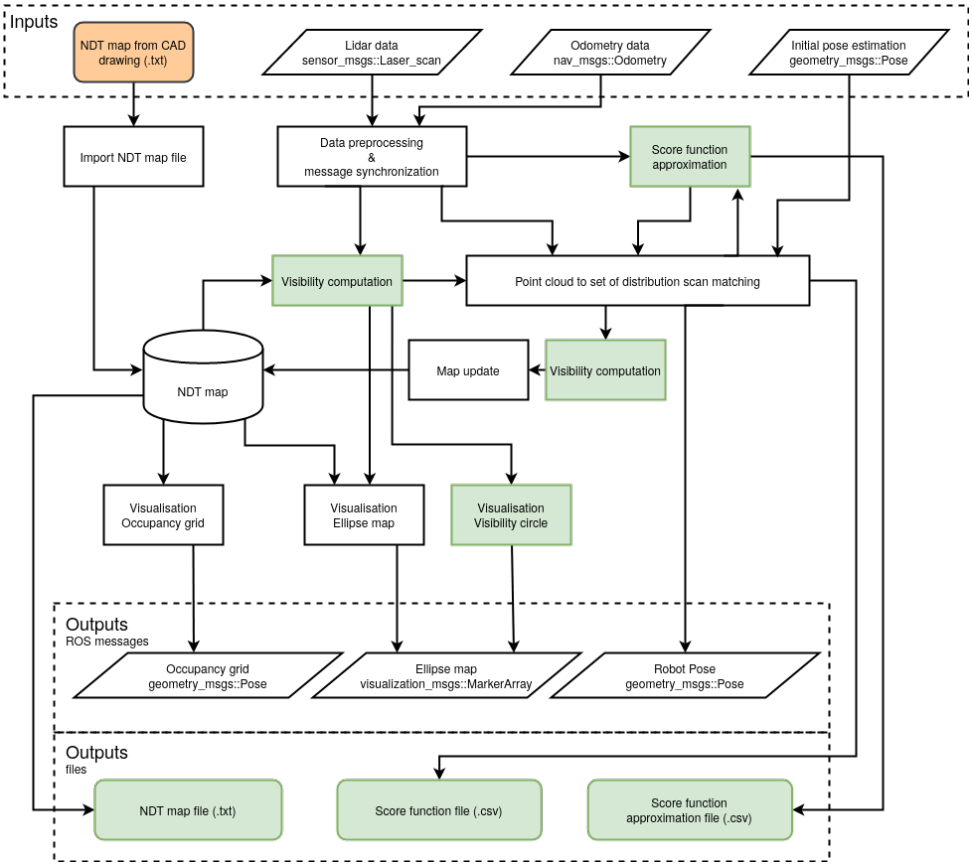


Figure 4.1: Overview of `ndt_ciirc_slam` ROS node. Conversion of CAD drawing to NDT map was implemented by Pánek (orange). Represention of the NDT map, together with scan matching and map update (white) is work of Nováček, who built upon work of Jelínek. The computation of visibility, score function approximation and exports of data files (green) were implemented by the author of this work.

The most important method of *GridVisibility* is *getVisibility*. This method takes three arguments, the complete NDT map, the transformation that the robot travelled since the last visibility computation and finally the guessed position, acquired from odometry. Firstly, the circle of sight is transformed by the robot motion. Visibility is than computed between the guessed position and cells containing individual parts of the discrete circle. Three options for visibility computation are provided. Ray casting, which utilises existing methods, Bresenham's line algorithm and Supercover Bresenham's line algorithm. New NDT map, established from the obstacles obtained by the visibility algorithm, is returned from this method.

Both the new NDT map and the visibility circle can be visualised independently in RViz. We decided to maintain the approach established by Nováček, therefore the frequency of visualisation is specified in an independent parameter.

The parameters of GridVisibility class are

- **cell_size_** - The size of individual NDT cells in the grid, defines the visibility circle (double).
- **visibility_angle_** - The aperture angle of the laser scanner (double).
- **visibility_sight_** - The maximum working range of the laser scanner (double).
- **visualize_circle_** - This value enables visualisation of NDT map from the visibility algorithm (boolean).
- **visualize_visibility_** - This value enables visualisation of the visibility circle (boolean).
- **visualize_every_x_sec_** - The frequency of visualisation of the visibility circle and NDT map (double).

4.2.3 P2DRegistration

The class P2DRegistration performs scan matching between a laser scan and an NDT map. This computation is performed by calling the method *computeTransformation*. The original class is described in [8], thus we outline only changes made with respect to the usage of prior odometry information and data file export.

The method *computeSingleGrid* performs one layer scan matching. Its arguments are the point cloud, NDT grid obtained from visibility, transformation estimate from odometry and fitting parameters. The prior odometry transformation is needed for the creation of approximation of the last score function $s(\mathbf{p_f})|_{t-1}$ wrapped in *ScoreApproximation* class, which is described in more detail in Chapter 3.

When working with highly abstract objects like multidimensional score functions or scan matching, it is often useful to visualise important data. For this reason, the option to export the score function computed on certain neighbourhood of the final pose, together with its approximation, was added. The method *exportScoreFunction* numerically computes values of the score function on the surroundings given by the following parameters

- **export_step_position** - The step length in metres in x and y directions (double).
- **export_step_orientation** - The step length in radians in θ direction (double).
- **export_num_of_steps** - The number of steps in x, y and θ direction (int).

- **export_score_data_** - This value enables the export of the score data (boolean).
- **score_data_file_name_** - The name of the export file (string).

Numerical computation of the score function on a neighbourhood of a reasonable size takes large amount of time and cannot be performed online on the robot. It is nevertheless possible to export the values while playing previously recorded data from a rosbag. The playing speed needs to be slowed down accordingly, at least to 1/10. An alternative would be to store only meta information, namely odometry and laser scan messages timestamps, last known robot pose and the NDT map, to analyse the data offline.

Since approximation of the score function is known analytically, it can be saved directly as the mean and covariance matrix. The two parameters are

- **export_score_approximation_** - This values enables the export of the score approximation (boolean).
- **score_approx_file_name_** - The name of the export file (string).

4.3 Visualisation tools

Python scripts `plot_func_comparison.py` and `plot_func_sum.py` provide way for visualisation and analysing of score functions and their approximations. They use the Python package for scientific computation `numpy` for vector and matrices manipulation and other computation. The visualisation depends on `matplotlib`, a Python 2D plotting library. The scripts take two arguments, specified in file `config.py`.

- **FILENAME_FUNC** - Path to a file containing score function data exported from *P2DRegistration*.
- **FILENAME_APPROX** - Path to a file containing score approximation data exported from *P2DRegistration*.

The first tool, `plot_func_comparison.py`, displays the score function and its approximation side by side. Its user interface can be seen on Fig. 4.2. The second tool, `plot_func_sum.py`, enables analysis of the contribution of the approximation function from the previous sample M_{t-1} to the current score function s_t, as it displays the sum $M_{t-1} + s_t$. This permits easy examination of flattening of the approximation based on the robot's transformation from the previous to the current sample. Its user interface is shown on Fig. 4.3.

Essential elements and widgets are labelled with numbers from 1 to 8. The same elements are marked with the same numbers on both Fig. 4.2 and Fig. 4.3. The elements are:

1. Selector - Three sectional views are available. For the pose $p = [x,\ y,\ \theta]^T$, selection of the first option plots values of the score function for constant θ. The second options plots the values for constant y and the third for constant x.

2. Buttons - Sample change, for moving back and forth on the timeline. The use of keyboard arrow buttons is also possible.

3. Plot - Graph of maximal values of the score function in time. The green point highlights value of the current sample.

4. Slider - Timeline slider, providing similar function as the Sample change buttons.

5. Plot - Graph of the score function. Red arrow represents the pose obtained from odometry (the guess). The green arrow is the final position, acquired from the optimising cycle. The arrows in sample t point to the guessed position in sample $t + 1$. White text in bottom left corner states the euclidean distance between the guess and final pose.

6. Plot - Graph of the approximation. The meaning of arrows and the white text is the same as on 5.

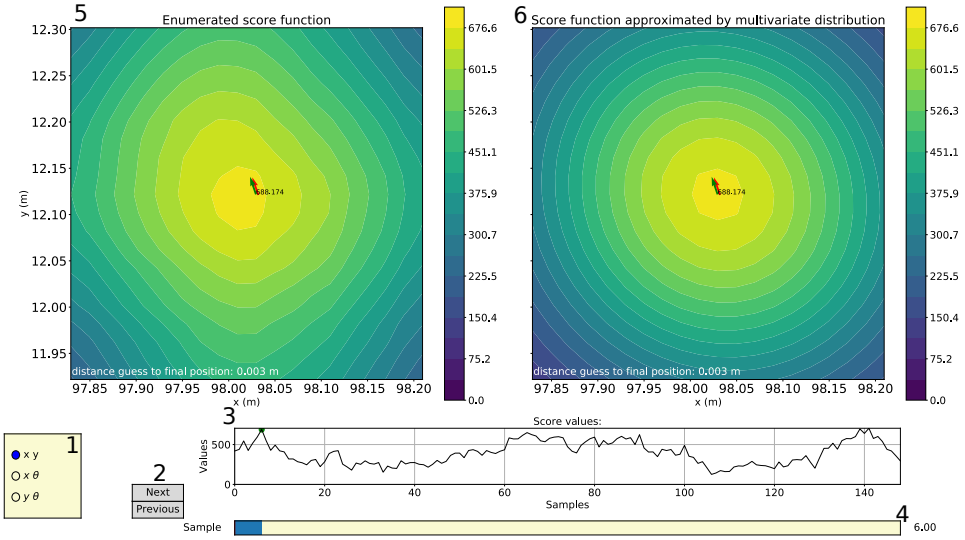


Figure 4.2: User interface of Python script `plot_func_comparison.py`

7. Slider - A user can use this slider to modify the coefficient k controlling maximum value of the approximation function. The maximum value of the contribution of the approximation M_{t-1} to score function s_t in time t is equal to $k \max(s_t)$.

8. Plot - Graph of the score function s_t summed with the approximation of the score function from the previous iteration M_{t-1}.

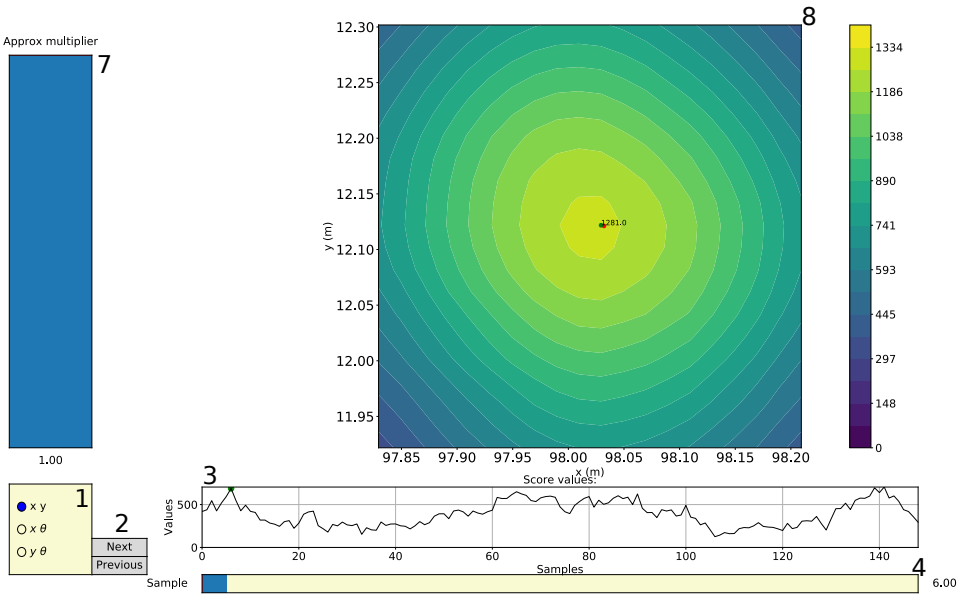


Figure 4.3: User interface of Python script `plot_func_sum.py`

Chapter 5

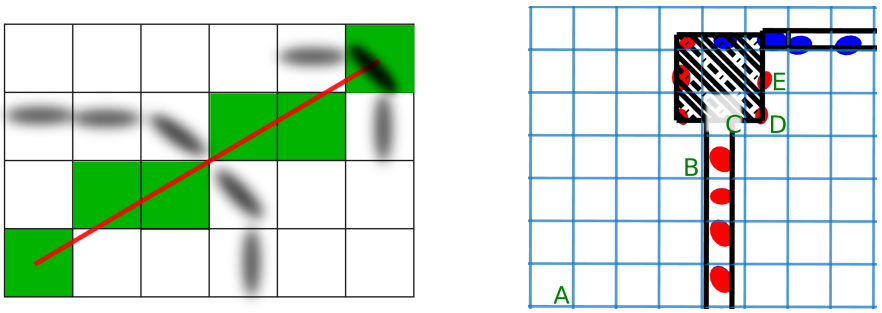
Practical observations

Several observations emerged during early experiments with the visibility algorithms described in the previous chapter. These observations were borne in mind during the experiments detailed in Chapter 6, which were performed in the same environment as the early ones. The setting of the robot was in all cases an office building, with the experimental platform passing though labs, corridors, kitchens and lecture halls. The NDT grid created from the original CAD drawing was used as the initial map.

5.1 Limitations of visibility algorithms in robotics

The algorithms introduced in Chapter 2 suffer several limitations when it comes to obstacle detection in complicated and unknown environments. The original CAD drawing may contain defects, whereas another may occur during the transform of the drawing into an NDT map. The location of individual obstacles is also influenced by the resolution of the NDT grid. Due to this, the cells containing obstacles may not necessarily border one another and the map therefore contains gaps. An example of unsuitable use of Bresenham's line algorithm, together with the effects in our environment, is shown on Fig. 5.1.

These problems are partially solved using supercover DDA algorithms, although in certain situations it does not help. An example of such a situation is a robot located close to a wall, pointing in the direction along the wall. Certain part of the rays cast from the robot's position to the section of the circle may end up in the same cell containing an obstacle, thus ignoring other sections of the wall and creating gaps. One way to overcome these limitations is using 4 overlapping grids and exploring visibility on all of them. This improvement was nevertheless abandoned for performance reasons.



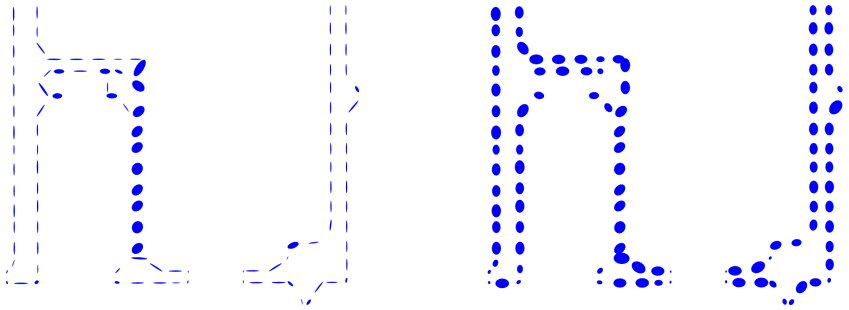
(a) : The ray is cast between the bottom left and upper right corners. Green colour shows cells checked by the algorithm. Due to the gap between two NDT cells, the ray goes through a wall.

(b) : The ray cast from cell A through B should be stopped in C. Instead, it continues and cells D and E are returned as visible (red ellipses). Original CAD drawing (black) was transformed to NDT (red and blue ellipses).

Figure 5.1: Unsuitable use of Bresenham's line algorithm

5.2 Eigen ration parameter in DXF to NDT conversion

The smallest possible ratio between the NDT covariance matrix eigen values, `eig_ratio` is used in the Panek's implementation of converter from CAD drawing to NDT maps. This ration ensures that the covariance matrix is always positive-semidefinitive, even if all of the measured points are collinear and therefore one of the eigen values of the matrix is zero. Biber and Straßer [2] set the zero eigen value to be equal to 0.001 of the largest eigenvalues and this value was kept as default for Panek's parameter `eig_ratio`. Nevertheless, this artificially forces the ellipsoids to be narrow, which negatively influences the ability of the NDT SLAM algorithm to fit LIDAR points to them. We decided to increase this value to 0.5. The difference between the original value and the new one can be seen on Fig. 5.2.



(a) : `eig_ratio = 0.001`

(b) : `eig_ratio = 0.5`

Figure 5.2: Comparison of NDT maps converted from CAD drawing using different eigen ratios parameters.

5.3 Visibility computation in our environment

As the robot drives through the environment, there are two possible approaches towards visibility computation. We can either compute the visibility on the whole map, or include only the cells that are static and which represent walls and other structures imported from CAD drawings. A whole map consists of both static and dynamic cells. Dynamic cells can stand for humans, another robots, furniture and other moving objects. Furniture can be specially useful when matching laser scan, as it often accounts for a great section of the recorded laser scan measurement.

On the other hand, the algorithm does not distinguish between movable objects and immobile furniture. The presence of dynamic, moving objects in the robot's map can block direct visibility to static building structures, such as walls. Although the algorithm implements NDT-OM and formerly occupied, now intersected cells can be updated to empty again, a passing human could still block a large portion of the static structures, especially if it is close to the robot. For these reasons, the implemented visibility algorithms take into account only static cells.

5.4 Inaccuracies in CAD drawings

During a construction of a building, the architects and civil engineers use CAD software to design and construct the building's structure, equip the building's floor with fire alarms and other security systems, as well as the air-conditioning and furniture. Different parts of CAD drawings arise in the course of the building's construction and may differ. It is therefore vital to generate the NDT map from the most recent and accurate CAD drawing.

Although the drawings are designed with millimetre precision, when it comes to the actual construction of the building, it is obviously impossible to maintain such a precise proportion. This leads to inconsistencies between the drawing and the real building and therefore between the NDT map and the environment where the robot moves. The localisation and mapping algorithm can be strongly influenced by this, as it heavily relies on a correct placement of the obstacles in correct cells. In the case of the office building where the experiments described in the next chapter were recorded, the difference between the width of one lecture hall defined in CAD drawings and a manually measured width was 5 cm. This lead to a duplication of one of the walls during SLAM. After a manual edition of the drawing, the building's walls align well with the laser scan and the duplication was reduced significantly.

5.5 Uncertainty of odometry

The pose gained from odometry and used for the approximation of the score function is influenced by certain error. This uncertainty, with respect to the translation and rotation performed by the robot between two iterations, should be taken into account and the approximation should be adjusted accordingly. Early experimental results showed that by tuning the parameter multiplying the diameter of the wheels of the robot, an error of less than 0.1 m can be achieved on a travelled distance of 35 m. During this experiment, the robot was driving only straightforward and on a flat surface of an office building's floor. However, if we set the minimal distance between two computations of NDT SLAM to 0.15 m, the error on this distance is less than a 0.001 m.

For this reason, we decided to set the coefficient k of the approximation $M(\boldsymbol{\mu}, \boldsymbol{\Sigma})|_t$ of the score function $s(\mathbf{p_f})|_t$ in time t constant

$$\max M_t = k \max s(\mathbf{p})|_t = 0.2 \max s(\mathbf{p})|_t = 0.2 s(\mathbf{p_f})|_t. \tag{5.1}$$

Even though experimental results seem reasonable for $k = 0.2$, more formal approach to dealing with the uncertainty of odometry remains a subject of future work.

Chapter 6

Experiments

In this chapter, we introduce the experimental platform and its components and describe two experiments, where we evaluate the improvements and problems brought by the use of visibility and odometry to the existing NDT SLAM algorithm. We also examine the change of values of the score function $s(\mathbf{p_f})$ during one round driven by the robot in a lecture room.

6.1 Experimental platform

The experimental platform (Fig. 6.1) is built upon robot Jackal from Clearpath Robotics^{TM}. It is equipped with Sick TiM561 LIDAR and Inertial Measurement Unit (IMU), which are crucial for this work. Another present accessories include a router, HTC Vive virtual reality system, information LED interface, safety foam elements and ceiling pointing digital camera. The camera itself is not used in this work, although a wider concept of localisation may rely on it. This is being addressed in a simultaneous theses.

Two computers are present on-board. The inner one is reserved for data collection and control, while the Intel NUC unit placed on top of the robot performs SLAM and other more complex computations. It has 8-core Intel Core i7-8650U, 4.20 GHz processor, 16 GB of RAM and 250 GB Hard Drive with Ubuntu 16.04 and ROS Kinetic. The Intel NUC unit is connected to the inner computer via a router through Ethernet interface. It is also possible to use WiFi in order to connect to the Intel NUC.

6.1.1 Robot Jackal

Robot Jackal is a small four wheel land robotics research platform, designed for development of autonomous robots. It is equipped with an onboard computer, GPS and IMU and offers integration with ROS with packages for localisation, navigation and visualisation.

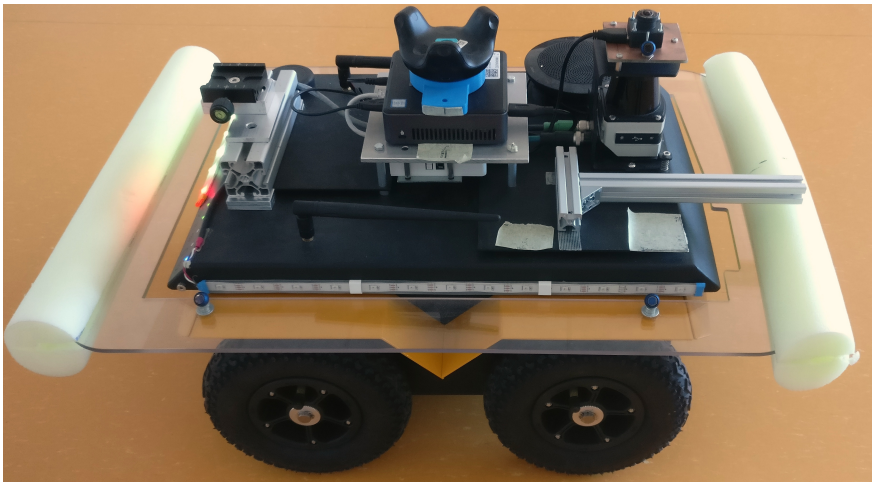


Figure 6.1: The experimental platform

Two 500 W motors enable speed of up to 2 m/s with payload of at most 20 kg. A 270 Watt hours battery is mounted, which allows of up to 8 hours of operating time. The onboard computer has CPU Celeron J1800, Dual core 2.4 GHz, 2 GB RAM, 32 GB Hard Drive together with Ubuntu 14.04 running ROS Indigo [13].

6.1.2 Laser scanner

The experimental platform is equipped with a 2D LIDAR Sick TiM561, mounted on the top front of the robot. It emits pulsed laser light, that is reflected by a rotating mirror. Reflections from obstacles are detected by a photodetector and transmitted to the robot's integrated computer via Ethernet interface [14]. It's parameters are listed in Table 6.1. The maximum working range value is used for visibility computation as the radius in the midpoint circle algorithm, while the aperture angle defines the effective section of this circle.

Sick TiM561	
Light source	850 nm
Laser class	1, eye-sage
Horizontal aperture angle	270 °
Scanning frequency	15 Hz
Angular resolution	0.33 °
Working range	0.05 - 10 m
Systematic error	± 60 mm
Statistical error	20 mm
Connection	Ethernet, USB 3
Ambient light immunity	80 000 lx

Table 6.1: Parametrs of Sick TiM561 LIDAR

6.2 Performed Experiments

The main focus of the following section is to evaluate improvements brought by the inclusion of visibility computation and odometry into the existing NDT SLAM algorithm. In the first experiment, we describe a long-term mapping of a lecture hall, with the robot passing 10 cycles. We evaluate the quality of the created map and compare it to the outcome of the previous version of the algorithm, witch did not use visibility. In the second experiment, the robot perform a passage through a corridor, an environment in which it is generally rather difficult to achieve accurate localisation. We emphasise the importance of the use of odometry. All the experiments share the following common properties and settings:

- The cell size was set to 0.25 m.
- All experiments used a map of the building's floor generated from a CAD drawing.
- The minimum distance and rotation between two scan matching computations were set to 0.15 m and 3° respectively.
- The robot was manually guided with a wireless Game controller.
- The maximum speed of the robot was set to 0.5 m/s.
- The frequency of odometry messages was 50 Hz and 15 Hz for the laser scan.
- The NDT map was visualised every 10 seconds.
- On the enclosed figures, blue ellipses represent static obstacles (walls and pillars) and green ones obstacles created by the mapping process.
- The used visibility algorithm was Bresenham's line algorithm.
- When using visibility, the sight of view was set to 10 m, and the visibility angle to 270° matching the parameters of the laser scanner.

6.2.1 First Experiment

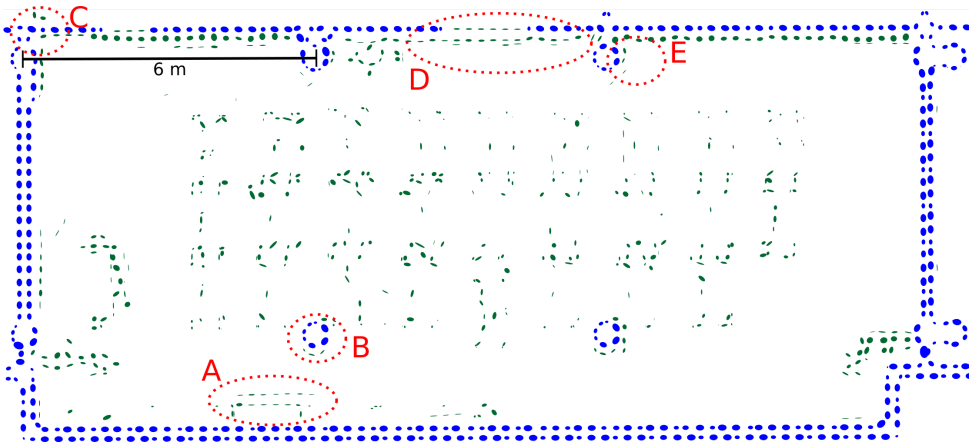
The first experiment was performed in a lecture hall displayed on Fig. 6.2 with furniture placed in its central section, surrounded by a narrow alley that the robot drove in. The original CAD drawing was manually edited to agree with the true dimensions of the hall. The difference in length of the room was 45 mm. The robot performed 5 rounds counterclockwise, followed by 5 rounds clockwise.

Figure 6.2: Environment of the first experiment

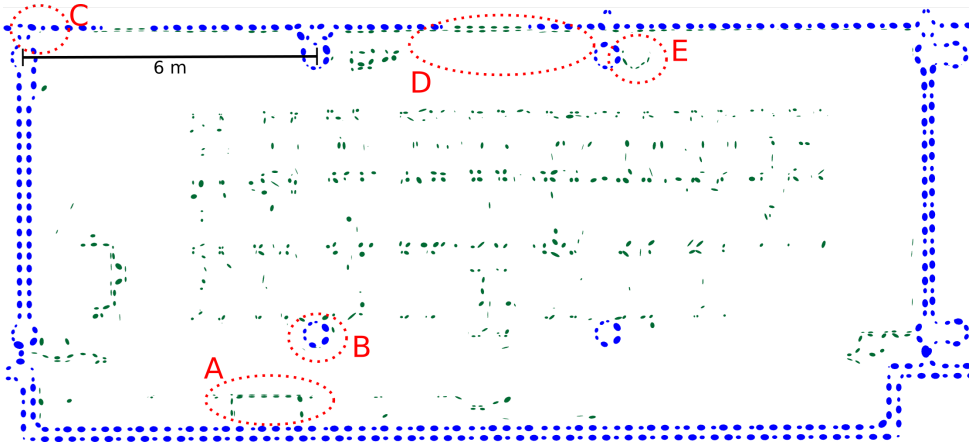
Details about duration of each algorithm are presented in Tab. 6.2. On this dataset, the use of visibility and odometry led to roughly 30 % decrease in average time needed for one SLAM cycle. The speedup was caused mainly by the reduction of iterations performed by the optimisation algorithm, that now operates with fewer cells. The use of prior odometry information and approximation of the score function decreased total number of optimising cycles per one SLAM round.

Algorithm	Minimal [ms]	Maximal [ms]	Average [ms]
NDT SLAM	82.113	186.273	121.683
NDT SLAM using visibility	50.172	123.759	77.862

Table 6.2: Duration of one SLAM cycle in the course of the first experiment



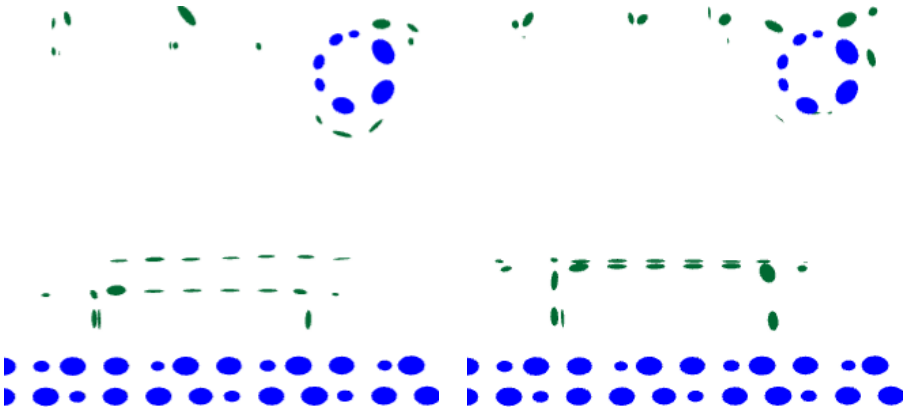
(a) : The final map of the first experiment produced **without** visibility after passing **ten** loops (five loops anti-clockwise and five loops clockwise).



(b) : The final map of the first experiment produced **with** visibility after passing **ten** loops (five loops anti-clockwise and five loops clockwise).

Figure 6.3: Comparison of the final map of the first experiment created **without** and **with** visibility. The areas highlighted by red circles are described in detail below.

Fig. 6.3 shows the final maps generated without and with the use of visibility. The central part of the map is occupied by furniture. Desks and chairs form a regular grid, which can be observed after closer look. The cell size of 0.25 m nevertheless prevents more detailed representation of such a small objects, as table and chair legs. The cluster of green ellipses near right-bottom corner represent a curtain, similar structure can be seen on the opposite side of the room. Above the left curtain is a teacher's desk. Two wide gaps in the north blue walls of the maps represent doors. The doors were closed during the experiment and the registration algorithm correctly filled previously empty cells. The green circular object left of **D** is not a duplication of a pillar, but a flowerpot with similar radius.

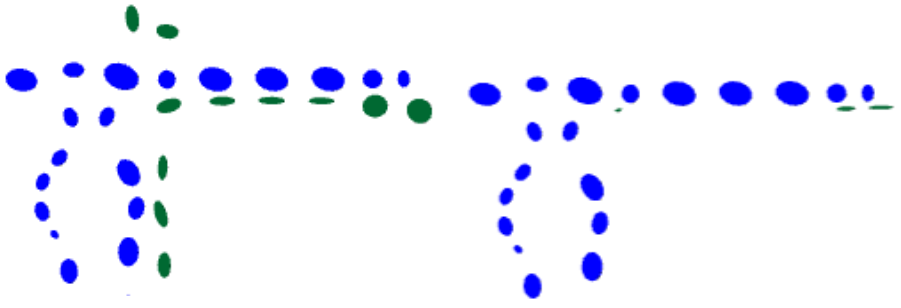


(a) : Details of areas **A** and **B** of the first experiment produced **without** visibility.

(b) : Details of areas **A** and **B** of the first experiment produced **with** visibility.

Figure 6.4: Comparison of areas **A** and **B** of the first experiment shows differences in duplication of a table and a pillar. The use of visibility prevented duplication of the pillar, while the duplication of the table was suppressed.

Area **A** is an NDT depiction of a table. On Fig. 6.4a the upper side of the table is noticeably duplicated, this does not repeat on 6.4b. The reason is that the original NDT SLAM does not distinguish between reverse and closer sides of the wall and the laser scan can be matched to the more distant side. On the other hand, since only the nearer side of the wall is provided to the matching algorithm by visibility computation, laser scan is matched correctly. The duplication of the pillar 6.4a (**B**) has the same cause.

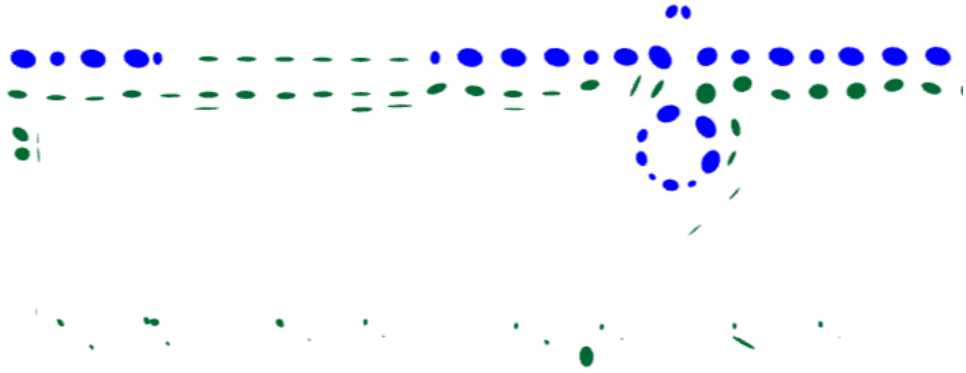


(a) : Detail of area **C** of the first experiment produced **without** visibility.

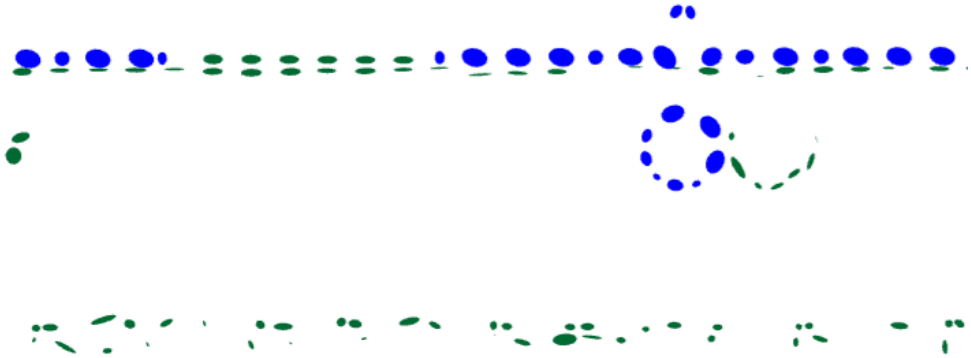
(b) : Detail of area **C** of the first experiment produced **with** visibility.

Figure 6.5: Comparison of area **C** of the first experiment details registration of new cells to the opposite side of the north wall.

Area **C** shown in detail on 6.5 provides example of laser data being nonsensically registered on the opposite side of the wall, which the robot evidently cannot observe. Similar issues can be also seen on the right side of Fig. 6.3a. This behaviour was eliminated by performing one extra iteration of the Visibility algorithm on the newly registered cells, mentioned earlier (Fig. 4.1).



(a) : Details of areas **D** and **E** of the first experiment produced **without** visibility.



(b) : Details of areas **D** and **E** of the first experiment produced **with** visibility.

Figure 6.6: Comparison of areas **D** and **E** of the first experiment shows differences of duplication of the north wall. While without visibility, the wall was on some places triplicated, the use of visibility prevented this. The error in registration of the pillar is detailed below.

Area **D** is a consequence of a similar phenomenon as **A** and **B**. Whereas on 6.6a the top wall is along its entire length duplicated and on some places triplicated, the duplicated cells on 6.6b are more sparse and the gaussians are visibly more narrow. These emerged while the robot was driving anticlockwise and had a good sightline on the lower wall and was conditioned to it by a large amount of points. Laser scan data that occasionally passed through the furniture located in the middle of the room were registered to wrong cells, but still very close to the correct one, as can be seen from the distance between blue and green ellipsoids on Fig. 6.6b. A future revision of the CAD drawing may further improve the algorithm's ability in similar situations.

The duplication of column on 6.6b (**E**) emerged during one of the final rounds of the experiment. As the robot was driving clockwise in the area between the two upper pillars (approximately **D**), not many points, nor cells were available for matching. The lower wall was almost invisible for the LIDAR, due to the furniture placed in the central section of the room. The upper wall lacked sufficient number of NDT cells, since it contains a wide door. Doors are not present in CAD drawings and aren't used in Visibility

computation. Therefore the majority of points occurred in front of the robot, whose location then converged slightly to the wall on the right side of the figure.

6.2.2 Second experiment

The goal of the second experiment was to study change of localisation and mapping in long corridors. The robot travelled distance of 36 m through a long corridor in both directions and explored three rooms along the way. In this experiment, the original CAD drawing was not manually edited. Closed doors, missing in the initial map, were correctly register as the robot was driving through the corridor.

Table 6.3 contains minimal, maximal and average times needed for one round of SLAM during this dataset. The differences are even more noticeable than in the first experiment, with the use of visibility and odometry being responsible for 40 % decrease in the average time.

Algorithm	Minimal [ms]	Maximal [ms]	Average [ms]
NDT SLAM	61.967	200.440	105.330
NDT SLAM using visibility	34.412	114.266	60.698

Table 6.3: Duration of one SLAM cycle in the course of the first experiment

Final maps obtained from NDT SLAM without and with visibility can be seen on Fig. 6.7 and 6.8. Once again, the areas highlighted by red circles are presented in more detail below. The two smaller rooms (below **A** and **B**) are kitchens. The robot explored at first the kitchen in the bottom of the picture and both algorithms produced similar results. Then, the robot drove towards **B**, with a stop exploring **A**. From the area **B** it returned back to **A**, where the experiment ended.

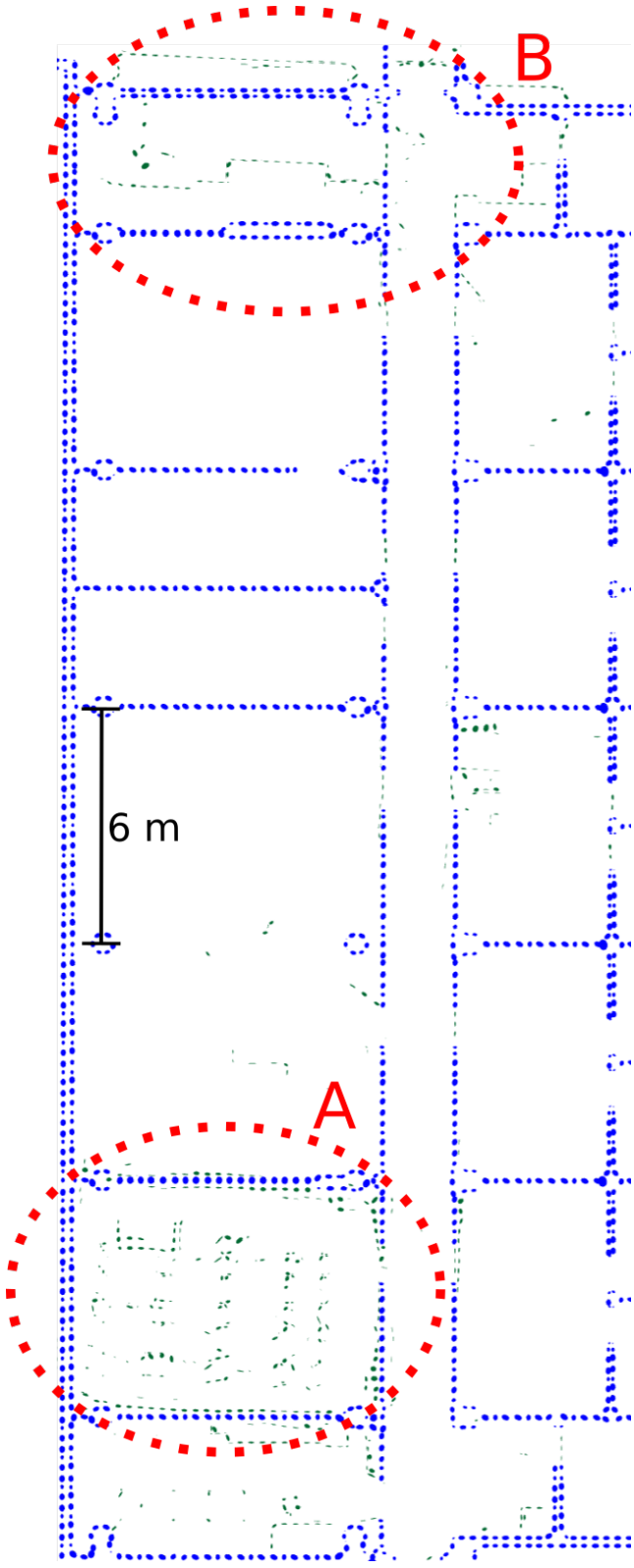


Figure 6.7: The final map of the second experiment produced **without** visibility.

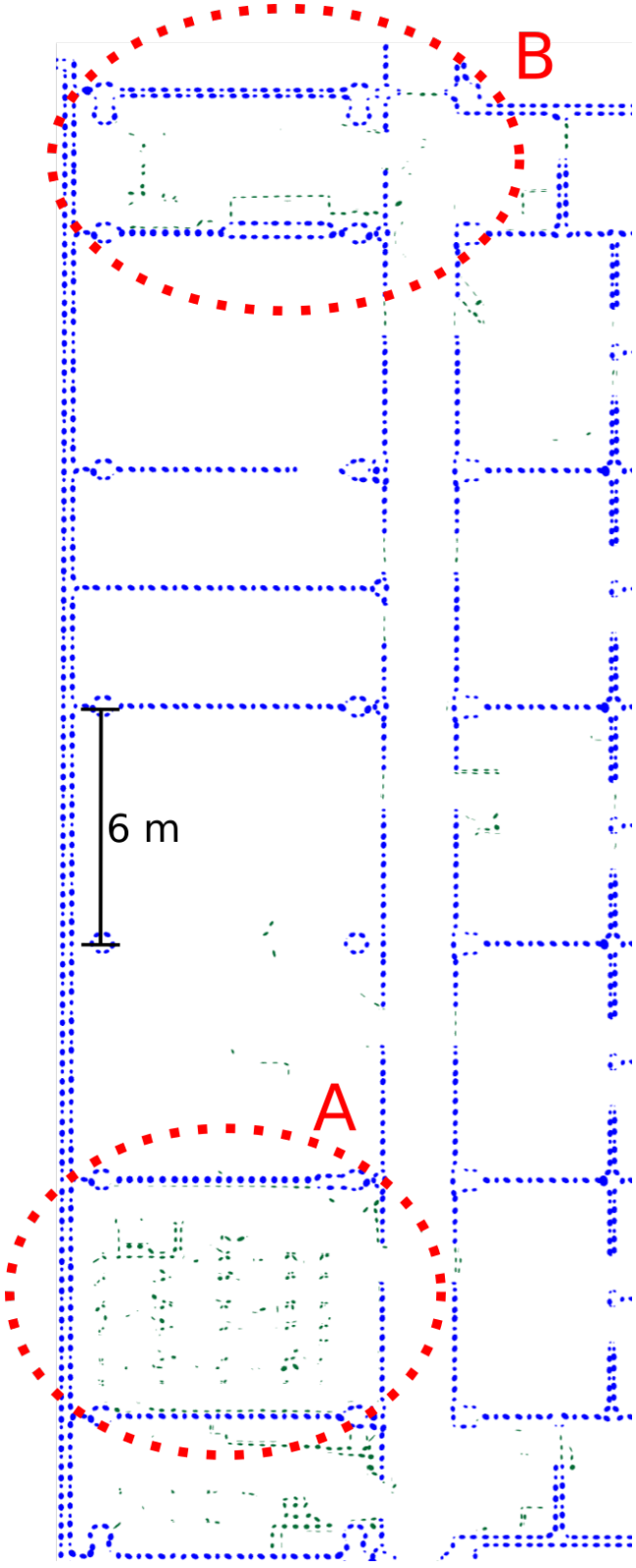
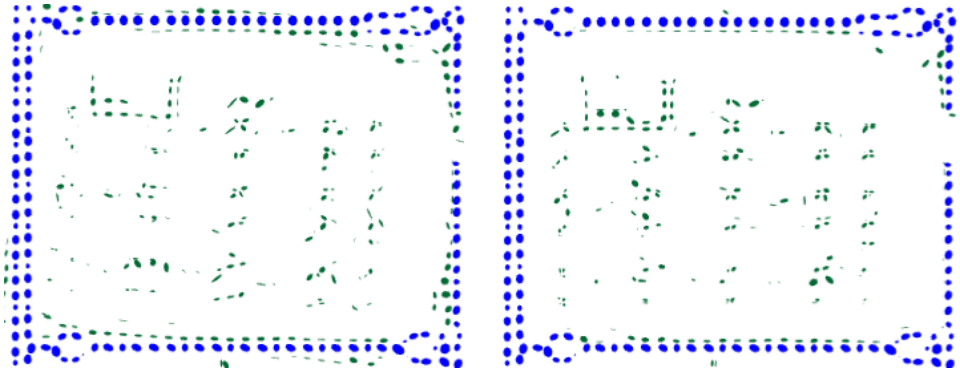


Figure 6.8: The final map of the second experiment produced **with** visibility.



(a) : Detail of area **A** of the second experiment produced **without** visibility.

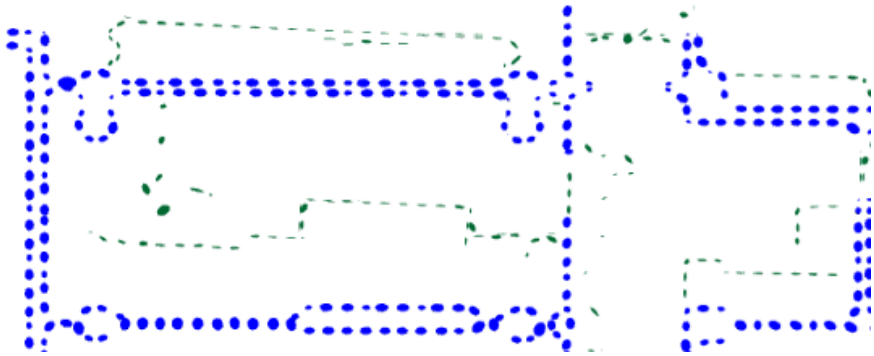
(b) : Detail of area **A** of the second experiment produced **with** visibility.

Figure 6.9: Comparison of area **A** of the second experiment shows detail of a small lecture room. The original CAD drawing was not manually edited according to the true dimensions of the room, which led to duplication of north and south walls in both scenarios. The use of visibility prevented registration of new cells to the opposite sides of west and north walls.

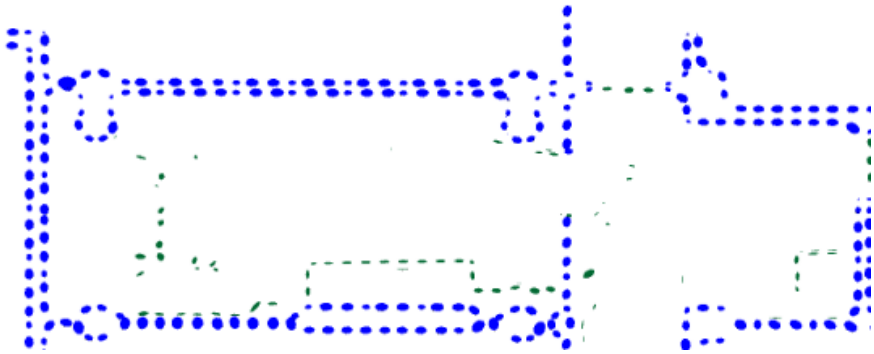
Fig. 6.9 details area **A**. The observed room was a small lecture hall, with open door on the right side and furniture in the central part. A duplication of the north and south walls can be noticed on both Fig. 6.9a and 6.9b. The reason is that the real room is 36 mm narrower and 50 mm longer than in the CAD drawing. Whereas the difference in width did not cause any significant changes to the map produced by NDT SLAM with visibility (Fig. 6.9b), this cannot be said about the difference in length. As the robot drove through the room counterclockwise, the laser scan was firstly matched to the northern wall, duplicating the south one. Then the situation repeated, but reversely. Nevertheless, the use of visibility prevented registration of LIDAR points to the opposite sides of the north, west and south wall, as can be seen on Fig. 6.9b. The rare cells which appear on the opposite side of south and north wall were not registered when the robot was located inside of the lecture room showed on Fig. 6.9, but when it moved through the corridor east of this hall, and the kitchen south of it, respectively.

The second selected area, **B**, represents a kitchen and its surrounding, explored after a long drive through the corridor. Fig. 6.10 compares detailed map of **B** without and with visibility. On Fig. 6.10a, newly registere

d obstacles are clearly misaligned. The absence of odometry data lead to displacement of the robot's position when it was driving through the corridor. On the trajectory longer than 30 m, the distance between the correct and the actual position of the robot was more than 1 m. Fig 6.10b shows that the use of score function approximation and odometry prevents this.



(a) : Detail of area **B** of the second experiment produced **without** visibility.



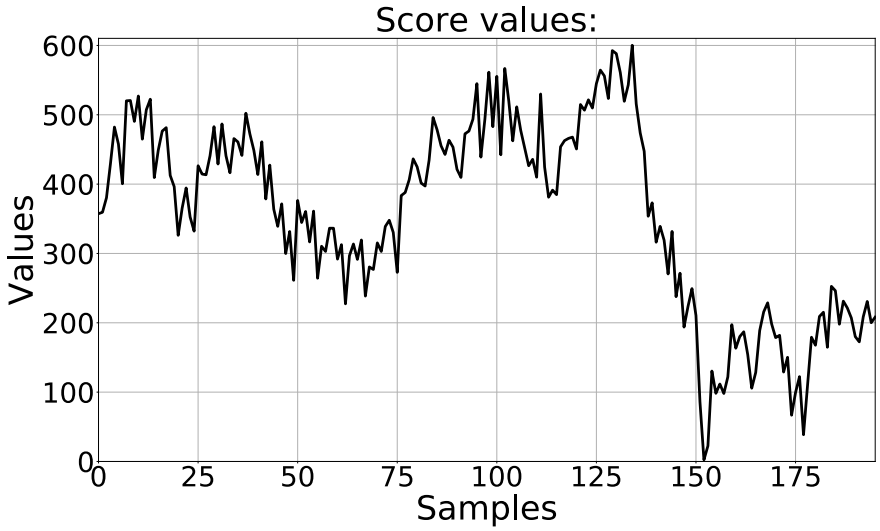
(b) : Detail of area **B** of the second experiment produced **with** visibility.

Figure 6.10: Comparison of area **B** of the second experiment shows map of a kitchen and its surrounding after a long drive through the corridor. The use of visibility and odometry stabilised the robot's position when it was driving through the corridor and prevented its incorrect displacement.

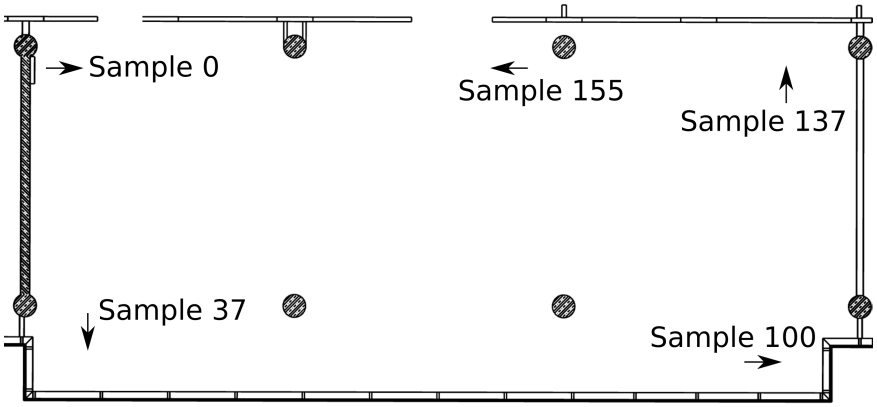
6.3 Course of the Score function

This section describes one round performed by the robot in a lecture room on the course of the value of the score function. Higher values of the function mean better estimate of robot's pose and are backed by a good alignment of the recorded point cloud and the cells obtained from visibility.

The values on Fig. 6.11a can be divided into 3 categories. Values greater than 500 are acquired in situations, when the robot stands in front of a wall and at the same time, it sees a wall on one of his sides. On Fig. 6.11a, these peaks appeared 4 times during the experiment. The robot started in the upper left corner of the hall. The first of the two peaks between samples 0 and 50 corresponds to the initial pose of the robot. The second one was gained after the robot's movement along the shorter side of the room. The third peek (around sample 100) emerged as the robot was driving along the longer side of the room to the bottom right corner. The last peak (sample 135) corresponds to the upper right corner of the hall. Fig. 6.11b gives context for the situation described earlier.



(a) : Values of the score function as the robot drives around a lecture hall.



(b) : Robot's poses giving context to the values of the score function.

Figure 6.11: Graph of values of the score function, together with robot's poses in a lecture room.

Second category contains values between 100 and 500. This corresponds to the robot driving alongside a wall, matching only a part of the scanned points.

Finally, the last category contains values less than 100. This result does not usually represent a sufficient match between LIDAR data and the map. In other words, the robot is lost. These are the situations when it is useful to prefer odometry estimate of the robot's pose. On Fig. 6.11b, this happens as the robot approaches the hole in the upper wall of the room. While there is a lack of NDT cells on robot's right, due to large distance and furniture placed in the center of the room, it does not observe the wall in front of it, nor on its left.

6.3.1 Visualisation of the Score function

In this section, we demonstrate several examples of visualisation of the score function $s(\mathbf{p})$. The function was plotted on a grid of 0.4 m by 0.4 m, 0.4 m by 0.4 rad respectively. Each example consists of 4 plots and 1 figure providing context of the situation. On these graphs, LIDAR sensor located on the robot (yellow arrow) measures a set of points (black squares). The blue ellipses represent prior information about static structures, imported from CAD drawing of the building, while the red ones are currently visible. Green ellipses are newly registered obstacles, that are not part of the original CAD drawing. Blue ellipses (non visible walls) and green ellipses are at this point ignored, as they are not taken into account during PCL-to-NDT registration.

Since the function $s(\mathbf{p})$ is a function of 3 variables $\mathbf{p} = [x, y, \theta]^T$, the plots represent planar cuts through score function in the final point $\mathbf{p_f}$. This is the point Newton's algorithm converges to and it can be found in the centre of each plot.

The first situation can be seen on Fig. 6.12. This corresponds to sample 135 on Fig. 6.11a, with one of the largest values of $s(\mathbf{p})$. The score function graph of this situation is on 6.13. The brighter the colour, the more points were matched with greater certainty. The conditions visible on Fig. 6.12 agree with the shape visible on graph 6.13a. Both x and y axis are well aligned with the walls (red ellipses), which forms a cross-liked shape.

The second situation is displayed on Fig. 6.14. In this case, the robot is placed in a corridor, with nearly all measured points collinear, forming 2 lines parallel to the y axis. The score function graph of this situation can be seen on 6.15. While the position in the direction of x axis is well constrained with uncertainty of the range of 0.05 metres, the position among y axis is unknown, since there are no visible obstacles matching visible NDT cells in this direction.

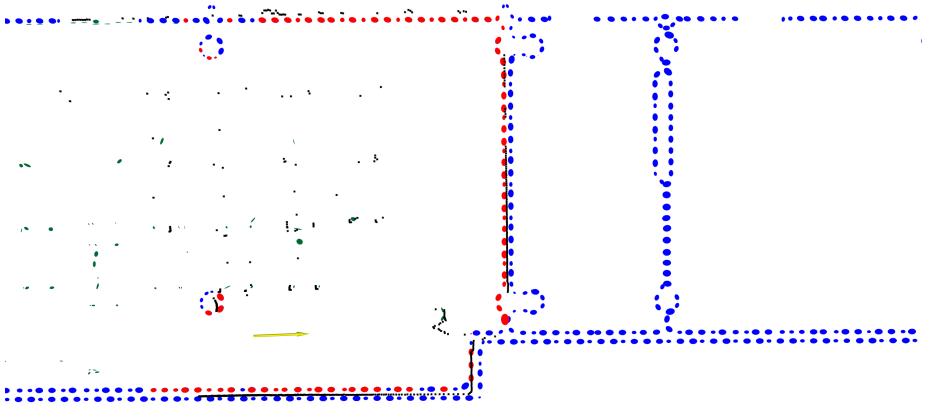
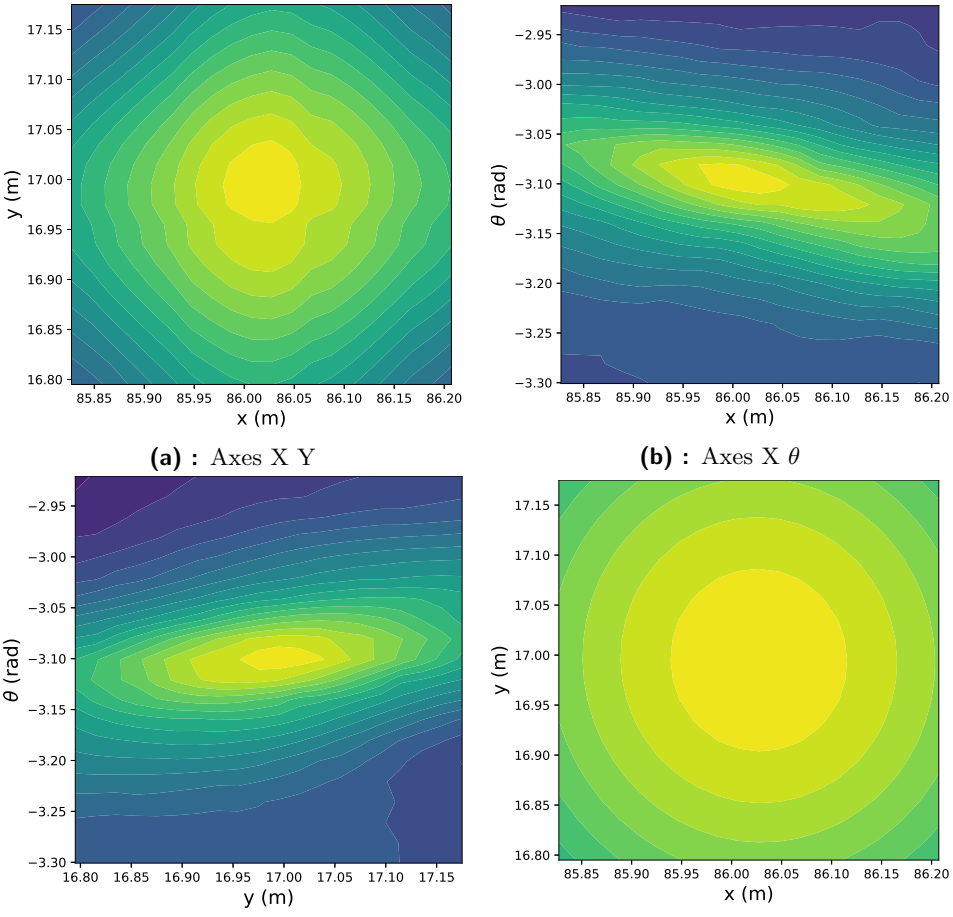


Figure 6.12: Situation 1. Both x and y axis are conditioned by sufficient number of points.



(a) : Axes X Y

(b) : Axes X θ

(c) : Axes Y θ

(d) : Axes X Y - Approximation

Figure 6.13: Visualisation of the score function and its approximation where both x and y coordinates are supported by sufficient number of points.

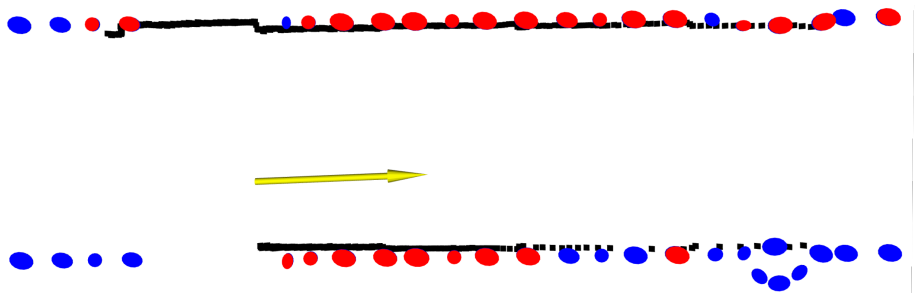
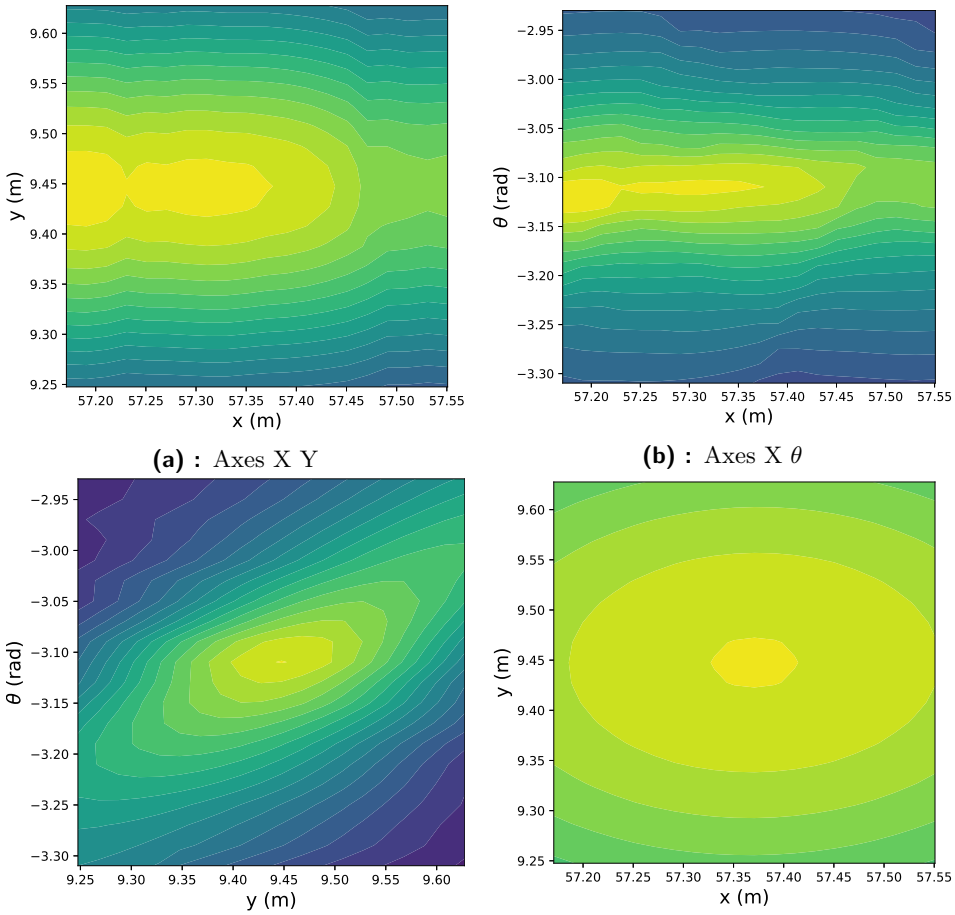


Figure 6.14: Situation 2. Only one axis is supported by sufficient number of measured points, position along corridor axis is not restricted by data.



(a) : Axes X Y

(b) : Axes X θ

(c) : Axes Y θ

(d) : Axes X Y - Approximation

Figure 6.15: Visualisation of the score function and its approximation where the robot is located in a corridor.

Chapter 7

Conclusions

The experimental results presented in Chapter 6 showed that the use of visibility and prior odometry information can improve both time performance and correctness of the existing NDT SLAM algorithm. While the computation of static objects visible from the robot's location greatly reduced duplication of walls, it also prevented registration of laser scan points to reversed sides of walls. Future improvements may deal with inaccuracies of CAD drawings, which would further improve robustness of the algorithm.

Approximation of the score function, described in Chapter 3, contributed to better results of the algorithm in corridors, which commonly represent a difficult environment for SLAM based on laser scans.

The implemented classes and methods extend the original NDT SLAM algorithm with an open interface for future changes of visibility computation, or the integration of the uncertainty of odometry. Two GUI offer easy option to further study the score function and the behaviour of the optimising cycle.

Bibliography

[1] Michael Abrash. *Michael Abrash's graphics programming black book*. Coriolis, Albany, NY, 1997.

[2] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. *IEEE International Conference on Intelligent Robots and Systems*, 3:2743 – 2748 vol.3, 11 2003.

[3] Jiri Bittner and Peter Wonka. Visibility in computer graphics. *Environment and Planning B: Planning and Design*, 30:729–755, 09 2003.

[4] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[5] Erik Einhorn and Horst michael Gross. H.m.: Generic 2d/3d slam with ndt maps for lifelong application. In *European Conference on Mobile Robots*, pages 240–247, 2013.

[6] L. Jelínek. Graph-based slam on normal distributions transform occupancy map. Bachelor thesis, Department of Theoretical Computer Science and Mathematical Logic, Faculty of mathematics and physics, Charles University, Prague, 2016.

[7] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems 18*, page 249–275, 1997.

[8] D. Nováček. Localization of mobile robot using multiple sensors. Master thesis, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, 2018.

[9] V. Pánek. Map import for mobile robot from CAD drawing. Bachelor thesis, Department of Cybernetics, Czech Technical University in Prague, Prague, 2018.

[10] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source

robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[11] J. Saarinen, T. Stoyanov, H. Andreasson, and A. J. Lilienthal. Fast 3d mapping in highly dynamic environments using normal distributions transform occupancy maps. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4694–4701, 2013.

[12] C. Schulz, R. Hanten, and A. Zell. Efficient map representations for multi-dimensional normal distributions transforms. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2679–2686, 2018.

[13] Jackal UGV - Small Weatherproof Robot - Clearpath. `https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/`. Accessed: 2020-05-10.

[14] TiM561-2050101 | Detection and ranging solutions | SICK. `https://www.sick.com/us/en/detection-and-ranging-solutions/2d-lidar-sensors/tim5xx/tim561-2050101/p/p369446`. Accessed: 2020-05-10.

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Boxan Matěj** Personal ID number: **469915**

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:

NDT SLAM Respecting Visibility

Bachelor's thesis title in Czech:

Lokalizace mobilního robotu metodou NDT za využití viditelnosti

Guidelines:

1. Get familiar with ROS and NDT SLAM.
2. Propose algorithms which improve stability and robustness of localization in NDT SLAM generally and also specifically using visibility of the obstacles.
3. Implement algorithms, test them in the real scenarios, and evaluate results.
4. Make conclusion.

Bibliography / sources:

[1] Lifelong localization in changing environments, Gian Diego Tipaldi, Daniel Meyer-Delius, Wolfram Burgard, IJRR 2013
[2] Einhorn, E.; Gross, H.-M., "Generic 2D/3D SLAM with NDT maps for lifelong application," in Mobile Robots (ECMR), 2013 European Conference on , vol., no., pp.240-247, 25-27 Sept. 2013
[3] John F. Hughes, Andries van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley: Computer Graphics: Principles and Practice, Addison-Wesley Professional; 3 edition (July 20, 2013)

Name and workplace of bachelor's thesis supervisor:

Ing. Vladimír Smutný, Ph.D., Robotic Perception, CIIRC

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

Date of bachelor's thesis assignment: **09.01.2020** Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

Ing. Vladimír Smutný, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	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

Appendix A

CD content description

- include - header .h files
  - ...
- launch - ros launch files
  - ...
- maps - example ndt maps
  - ...
- src - source .cpp files
  - ...
- visualisation - visualisation python scripts
  - data
    - score_approx
      - ...
    - trajectories
      - ...
  - plot_func_sum.py
  - plot_func_comparison.py
  - config.py
  - ...
- CMakeList.txt
- package.xml
- README.md