

**Bachelor Project**



**Czech  
Technical  
University  
in Prague**

**F3**

**Faculty of Electrical Engineering  
Department of Cybernetics**

## **Map Import for Mobile Robot from CAD Drawing**

**Vojtěch Pánek**

**Supervisor: Ing. Vladimír Smutný, Ph.D.  
Field of study: Cybernetics and Robotics  
Subfield: Robotics  
May 2018**



## I. Personal and study details

Student's name: **Pánek Vojtěch** Personal ID number: **457211**  
Faculty / Institute: **Faculty of Electrical Engineering**  
Department / Institute: **Department of Cybernetics**  
Study program: **Cybernetics and Robotics**  
Branch of study: **Robotics**

## II. Bachelor's thesis details

Bachelor's thesis title in English:

**Map Import for Mobile Robot from CAD Drawing**

Bachelor's thesis title in Czech:

**Import mapy z výkresu pro mobilního robota**

Guidelines:

1. Get familiar with following topics:
  - a. ROS operating system,
  - b. CAD drawings,
  - c. Normal Distributions Transform (NDT) representation of the mobile robot map.
2. Choose publicly documented form of CAD 2D drawings format and implement converter from this format to the NDT map representation.
3. Test the implemented import filter on real data.
4. Make conclusion.

Bibliography / sources:

- [1] Biber Peter: The Normal Distributions Transform: A New Approach to Laser Scan Matching, Las Vegas 2003.
- [2] Todor Stoyanov, Jari Saarinen, Henrik Andreasson, Achim J. Lilienthal: Normal Distributions Transform Occupancy Map fusion: Simultaneous mapping and tracking in large scale dynamic environments, IROS 2013, Tokyo.

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: **06.12.2017** Deadline for bachelor thesis submission: **25.05.2018**  
Assignment valid until: **30.09.2019**

\_\_\_\_\_  
Ing. Vladimír Smutný, Ph.D.  
Supervisor's signature

\_\_\_\_\_  
doc. Ing. Tomáš Svoboda, Ph.D.  
Head of department's signature

\_\_\_\_\_  
prof. Ing. Pavel Ripka, CSc.  
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



## Acknowledgements

I would like to offer my special thanks to my supervisor Vladimír Smutný for his patient guidance and many useful suggestions during the whole work. I would also like to thank David Nováček for his advice on NDT SLAM implementation in Jackal robot. Finally, I wish to thank my family and girlfriend for their support.

## Declaration

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

Prague, date .....

signature .....

## Abstract

This thesis aims to implement a converter from 2D CAD drawing of a building to NDT (Normal Distribution Transform) representation of map used to indoor navigation a for mobile robot. The map should serve for initialization of lifelong localization.

DXF (Drawing Exchange Format) was selected as an input format for the converter. Other common CAD formats can be converted to DXF by CAD editor.

The thesis briefly examines few other topics related to the conversion and use of CAD drawing in mobile robotics. The first one is automatic adjustment of CAD drawing. The second is a visualization of CAD drawing in ROS RViz. Lastly, it examines possibilities of defining destinations for robot motion planning in CAD drawing.

**Keywords:** NDT map, DXF CAD drawing, ROS, RViz, mobile robotics

**Supervisor:** Ing. Vladimír Smutný, Ph.D.

## Abstrakt

Tato bakalářská práce je zaměřena na implementaci převodníku z 2D CAD výkresu budovy do NDT (Normal Distribution Transform) reprezentace mapy, kterou využívá mobilní robot pro navigaci ve vnitřních prostorách. Mapa by měla sloužit jako počáteční informace pro celoživotního určování polohy robotu.

Jako vstupní formát byl vybrán DXF (Drawing Exchange Format), do kterého lze pomocí CAD editoru převést ostatní běžně užívané formáty.

Práce se dotýká několika dalších témat spojených s převodem a využitím CAD výkresů v mobilní robotice. Prvním je strojová úprava CAD výkresů. Druhým je zobrazení výkresu v ROS RViz. Posledním zkoumaným tématem je možnost využití CAD výkresů pro zakreslení cílových destinací pro plánování trasy.

**Klíčová slova:** NDT mapa, DXF CAD výkres, ROS, RViz, mobilní robotika

**Překlad názvu:** Import mapy z výkresu pro mobilního robota

# Contents

<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
<b>2 State of the art</b>	<b>3</b>
2.1 CAD used in mobile robotics . . . .	3
2.2 NDT . . . . .	3
<b>3 Description of used technologies</b>	<b>5</b>
3.1 DXF . . . . .	5
3.1.1 The structure of ASCII DXF . .	5
3.2 NDT . . . . .	8
3.3 Alpha shape . . . . .	11
3.4 Axis-aligned bounding boxes . . .	13
3.5 RViz . . . . .	13
<b>4 Converter from DXF drawing to NDT map representation</b>	<b>15</b>
4.1 Program overview . . . . .	15
4.2 Implementation . . . . .	15
4.2.1 Implementation of converter from DXF CAD drawing to NDT map representation . . . . .	15
4.2.2 Implementation of station blocks export . . . . .	17
4.2.3 Implementation of RViz visualization . . . . .	17
4.3 User manual . . . . .	18
4.3.1 Installation . . . . .	19
4.3.2 Example of conversion . . . . .	19
4.3.3 Configuration file . . . . .	20
4.3.4 Output file . . . . .	22
4.3.5 Stations . . . . .	22
4.3.6 Visualization of CAD drawing in RViz . . . . .	23
<b>5 Practical observations</b>	<b>25</b>
5.1 Imperfections in CAD drawings .	25
5.1.1 Hatches . . . . .	25
5.2 Adjusting the drawing by Alpha shapes . . . . .	26
5.3 Division of entities to cells . . . . .	28
<b>6 Experimental results</b>	<b>29</b>
<b>7 Conclusion</b>	<b>35</b>
<b>Bibliography</b>	<b>37</b>
<b>A Example of configuration file</b>	<b>39</b>
<b>B CD content description</b>	<b>43</b>

## Figures

3.1 Gaussian curve with 2D domain .	9
3.2 Example of 2D map - CAD drawing and NDT . . . . .	9
3.3 Example of the Alpha shape for different $\alpha$ . . . . .	11
3.4 Alpha shapes for $\alpha \rightarrow 0$ and $\alpha \rightarrow \infty$ . . . . .	12
3.5 Relation between Delaunay triangulation and Alpha shape . . . . .	12
3.6 Axis-aligned bounding boxes . . . . .	13
4.1 Converter program structure . . . . .	16
4.2 Structure of RViz visualization process . . . . .	18
4.3 Visualization of loaded CAD drawing and NDT grid in RViz . . . . .	23
5.1 Alpha shape of wall with diagonal hatch . . . . .	26
5.2 Calculation of sampling density for Alpha shape . . . . .	27
5.3 Alpha shape of wall with parallel lines . . . . .	27
5.4 Alpha shape - corner distortion . . . . .	28
5.5 Bounding box of entity and cells . . . . .	28
6.4 Verification - RViz . . . . .	30
6.1 Verification - lab . . . . .	31
6.2 Verification - stairs . . . . .	32
6.3 Verification - lecture hall . . . . .	33

## Tables

3.1 List of sections in DXF . . . . .	6
3.2 List of important DXF entities . . . . .	7
3.3 List of DXF block general variables . . . . .	8
4.1 NDT cell parameters . . . . .	22



# Chapter 1

## Introduction

Majority of mobile robots with a certain level of autonomy needs to have information of its relative location against some points in the real world for navigation. It also needs a position of surrounding obstacles for collision avoidance. These challenges are often solved using SLAM (Simultaneous Localization And Mapping) algorithms. SLAM algorithms use a representation of a map, which is being built and used for navigating the environment at the same time [4]. Mobile robotics use various implementations of SLAM algorithms and map representations.

We have two possible solutions for creating the map of a new environment, where we want to use a mobile robot. The first option is to deploy the robot and let it create the map itself. This solution has the disadvantage of possible distortion of the map shape caused by measurement errors. The second option is to take a precise map of the environment and convert it to map representation used in the robot. Architectonic building plans for indoor environments are usually available and can be used for this purpose. Assuming that technical drawings are more accurate than robot range scans, bigger weight can be assigned to data from drawing. This work deals with the conversion from architectonic CAD drawing to NDT (Normal Distribution Transform) map representation.

NDT is a relatively new method of range scans matching introduced in [1]. Range scans matching is a part of many SLAM implementations. NDT introduces maps based on a grid, where each grid cell contains multivariate normal distribution, giving probabilistic information of obstacle presence and weight, dependent on the number of measurements in the cell. This work describes mainly theory concerned with NDT map representation, and other parts of NDT algorithm will be explained only briefly.

### 1.1 Motivation

The problem of map import was encountered during development of mobile robot (AGV - Automated Guided Vehicle) [3] used for moving material in an indoor environment. This AGV does not use any markers or reflectors placed in environment for navigation, and it relies only on a scanning of surrounding obstacles and on an odometry. The scanning is done using one

lidar with horizontal scanning plane. The NDT implementation is used as part of SLAM.

The testing of implemented SLAM has revealed a few drawbacks. The first one is deformation of generated maps as a result of odometry measurement errors. The second one is false loop-closing when the robot goes through a monotonous corridor, where consecutive lidar scans look very similar. The CAD drawing could provide initial NDT map free of global map deformations, which could suppress influence of measurement errors.

A potential user of AGV will most likely want to set up locations of various points of interest (e.g. charging stations, loading and unloading stations) and areas (restricted zones, pedestrian crossings). This information may be inserted into the drawing and exported for path planning purposes.

Another useful feature for user could be some visualization of robot position in environment. The used robot runs on ROS (Robot Operating System), so the ROS RViz package can be used for the visualization.

## Chapter 2

### State of the art

Two essential items with which the thesis deals are CAD architectural drawings and NDT maps. This chapter briefly introduces the related literature.

#### 2.1 CAD used in mobile robotics

There are a few works related to use of CAD (Computer Aided Design) architectural drawings for the purpose of SLAM and navigation. Murarka and Kuipers in [9] divide building drawing to closed rooms separated by doors and create a topological map, where rooms are nodes and doors are edges. They describe an algorithm for room extraction based on casting an infinite ray from the room number text position to the first wall it crosses and subsequently following the enclosing walls until the area is watertight. Also, the rooms are stored in the form of medial axis transform.

Borkowski et al. in [10] examine utilization of BIM (Building Information Modeling) for mobile robots. BIM framework describes the building itself and many built-in systems (e.g. air ventilation, security). The paper also deals with a transfer of rooms geometry from Autodesk Revit BIM software to occupancy maps and creation of topological maps similar to [9], but with an addition of accessibility information. A concept of robot integration to building BIM model is proposed.

#### 2.2 NDT

NDT (Normal Distribution Transform) is one of the methods of map representation and scan matching in SLAM. It divides environment map to a grid. Each grid cell contains an approximation of the respective part of the environment by a normal distribution.

NDT map was introduced by Biber and Straßer in [1]. They use 2D grids with cells of uniform size. Four mutually shifted grids are used to overcome effects of space discretization. Their approach uses computation of the cell parameters from points acquired by one laser scan and falling into the cell. The collinearity of all points in the cell and resulting singularity of the cell covariance matrix are solved by setting a minimal ratio between eigenvalues of



## Chapter 3

### Description of used technologies

#### 3.1 DXF

DXF (Drawing Interchange Format) is CAD drawing file format developed by Autodesk company for the purpose of exchange CAD drawings between AutoCAD and other CAD editors. The format was chosen for this work due to its publicly available specifications. Data in DXF can be represented in binary form or in ASCII text form which is used in this work.

This thesis draws from Autodesk 2012 DXF reference [16], which should correspond with R18.2 AutoCAD version.

##### 3.1.1 The structure of ASCII DXF

The basic structure of any ASCII DXF is formed by group codes on odd lines of file and values on even lines. The group code specifies the type of the value on the following line. The file is further divided into sections. A section always starts with group code 0 and value SECTION and ends with group code 0 and value ENDSEC. Group code 0 means that the following value is a string indicating the entity type, which could be section, entity, block, etc. Type of section is marked by group code 2 (name of entity). The boundary of HEADER section should look like this:

```
0
SECTION
2
HEADER
...
...
...
0
ENDSEC
```

There is a couple of section types, but only a few will be further described due to their importance to the thesis.

Section name	Description
HEADER	General information about drawing and CAD editor
CLASSES	Definitions of application-defined classes
TABLES	Various data tables (line styles, text styles, views definitions, quotation settings etc.)
BLOCKS	Definitions of blocks (entities merged under one identifier)
ENTITIES	List of entities
OBJECTS	Definitions of non-graphical entities (used e.g. for scripting)
THUMBNAILIMAGE	Optional thumbnail of drawing

**Table 3.1:** List of sections in DXF

### ■ ENTITIES section

ENTITIES section contains the list of all entities placed in the drawing. Single entity record starts with group code 0 and type of entity and continues to start of another entity or end of ENTITIES section.

```
0
LINE
...
...
...
0
CIRCLE
```

The list of variables in entity description varies between different types of entities.

Table 3.2 contains only entities and their variables which are being used in this work. DXF format supports much more entities, e.g. texts, light sources, 3D meshes and other complex geometric objects. Listed entities have more variables, of which we only mention two. The first is the unique hexadecimal identifier of each entity in drawing called handle (group code 5). The second is the name of the layer in which the entity is placed (group code 8).

All coordinates mentioned in table 3.2 are in WCS (World Coordinate System) of the drawing.

Entity type	Group code	Description
LINE	10	Start point x-coordinate
	20	Start point y-coordinate
	30	Start point z-coordinate
	11	Endpoint x-coordinate
	21	Endpoint y-coordinate
	31	Endpoint z-coordinate
CIRCLE	10	Center x-coordinate
	20	Center y-coordinate
	30	Center z-coordinate
	40	Radius
ARC	10	Center x-coordinate
	20	Center y-coordinate
	30	Center z-coordinate
	40	Radius
	50	Start angle
	51	End angle
LWPOLYLINE	90	Number of vertices
	10	Vertex x-coordinate (for each vertex)
	20	Vertex y-coordinate (for each vertex)
	42	Bulge of the line starting in the vertex (for each vertex)
INSERT	2	Name of inserted block
	10	Insertion point x-coordinate
	20	Insertion point y-coordinate
	30	Insertion point z-coordinate
	50	Rotation angle

**Table 3.2:** List of important DXF entities and the selection of their variables

## ■ BLOCKS section

Blocks are groups of entities which can be inserted into the drawing by INSERT entity. The nesting of blocks is possible by including the INSERT entity of an inner block inside an outer block.

Blocks start with 0 and BLOCK and end with next block or end of the section. Block definition contains some general info and definitions of block entities. The definition also contains the ENDBLK value, which marks the end of entities definitions, not the end of the block definition. Entities definitions have the same form as in ENTITIES section.

Name variable is used for block specification in INSERT entity. Base point corresponds to the insertion point of INSERT entity and creates origin of the coordinate system of block entities.

Group code	Variable
2	Block name
4	Block description
8	Layer
10	Base point x-coordinate
20	Base point y-coordinate
30	Base point z-coordinate

**Table 3.3:** List of DXF block general variables

Each entity inside the block definition has its layer variable defined, which could differ from the layer of the block. Entities of the block created in layer 0 inherit settings from the layer of the INSERT entity which placed their block to the drawing.

Turning the layer off makes disappear all entities created on the layer and inserted entities created on the layer 0. The inserted entities created on other layers than 0 remain visible. Turning the layer 0 off affects only entities currently present on this layer. Freezing and locking the layer also affects the inserted entities created on other layers than 0. Layer 0 is present in every drawing and is used to block creating.

Properties like color or linetype of entity can be inherited from the layer (ByLayer), from the block (ByBlock) or overridden in entity settings.

Block description is a simple character string. We use description for storage of information concerning control blocks. Data are stored in format similar to YAML maps (key: value). Newline symbol in AutoCAD is in DXF coded like " $\wedge$ M $\wedge$ J" sequence.

## ■ TABLE section

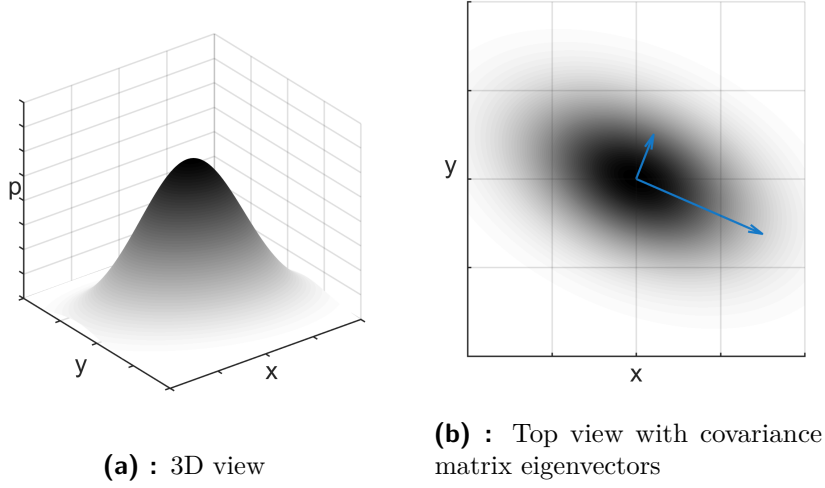
The TABLE section contains, among other tables, the list of all layers present in the drawing. Layers records contain their name, ID or bit coded standard flags. Flags describe settings like freezing, locking or referencing of the layer.

## ■ 3.2 NDT

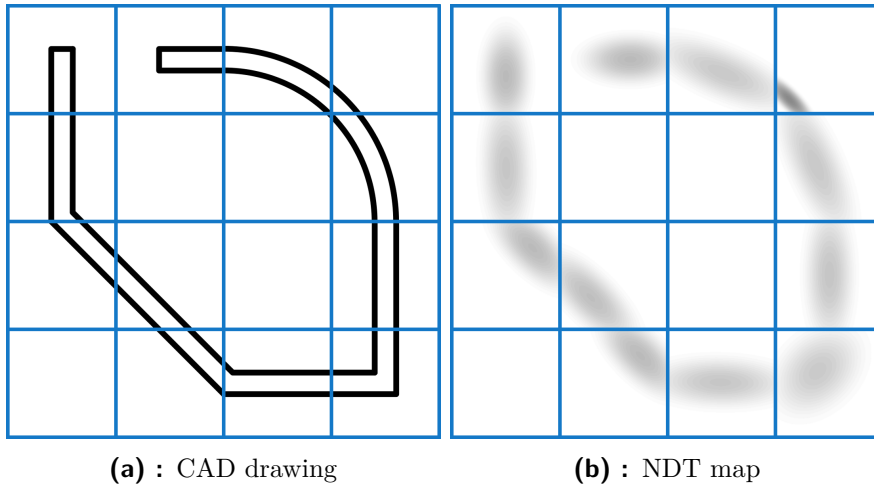
NDT stands for Normal Distribution Transform and is fundamental for NDT map representation. This map is very similar to occupancy grid. The difference is that probability distribution inside each cell is normal instead of uniform. The distribution describes the probability of measuring a point (by some rangefinder) at certain position in the cell, so we have much more precise information of occupancy inside the cells compared to the occupancy grid. The dimension of normal distribution domain is the same as of the used map, that is 2D for planar or 3D for spatial maps. The maps are being visualized by multivariate Gaussian curves. The example of 2D Gaussian



curve is in Figure 3.1. The examples of CAD map and an NDT map are in Figure 3.2.



**Figure 3.1:** Gaussian curve with 2D domain



**Figure 3.2:** Example of 2D map with visualized NDT cells

All the maps are considered to be 2D further through this thesis. 2D normal distribution is described by a vector of centroid

$$\mu = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} \quad (3.1)$$

and by a covariance matrix

$$\Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}. \quad (3.2)$$

The calculation of normal distribution parameters proposed in [2] gives bigger influence to the newest data, which makes an advantage in changing environment. We have the positions of static walls available a priori. We want to use the initial NDT map for correction of robot measurement errors. The computation better suited for our application is the standardly used unbiased parameters estimation

$$\mu = \frac{1}{n} \sum_{i=1}^n \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \quad (3.3)$$

$$\Sigma = \frac{1}{n-1} \sum_{i=1}^n \begin{bmatrix} x_i - \bar{x} \\ y_i - \bar{y} \end{bmatrix} \begin{bmatrix} x_i - \bar{x} \\ y_i - \bar{y} \end{bmatrix}^T. \quad (3.4)$$

The covariance matrix must be positive-semidefinite. This condition is not met if one of the eigenvalues of the matrix is zero, which happens when all measured points in the cell are collinear. Biber and Straßer in [1] solved these situations by setting too small eigenvalue to 0.001 times the bigger eigenvalue. We set the smallest ratio of the eigenvalues as the variable in a configuration file and to 0.001 by default.

NDT-OM (Normal Distribution Transform Occupancy Map) is same as NDT map with the addition of occupancy information in each cell. While creating the map with a robot, the occupancy values are being updated by the ray-tracing as in [7] and [5] or by the beam-sensor model like in [2]. We set occupancy value to -1 when the cell doesn't contain any geometric object from CAD drawing and to 100 otherwise. The value -1 denotes unknown occupancy value because free space in our map can be filled with objects in the real environment. The values for free and occupied cells were chosen to conform with the used SLAM implementation.

In [5] cells contain weight value in addition to above mentioned NDT cell parameters. It is incremented whenever new measurement falls into the cell and is included in NDT parameters. We set the constant weight to every occupied cell equivalent to the value set in configuration file.

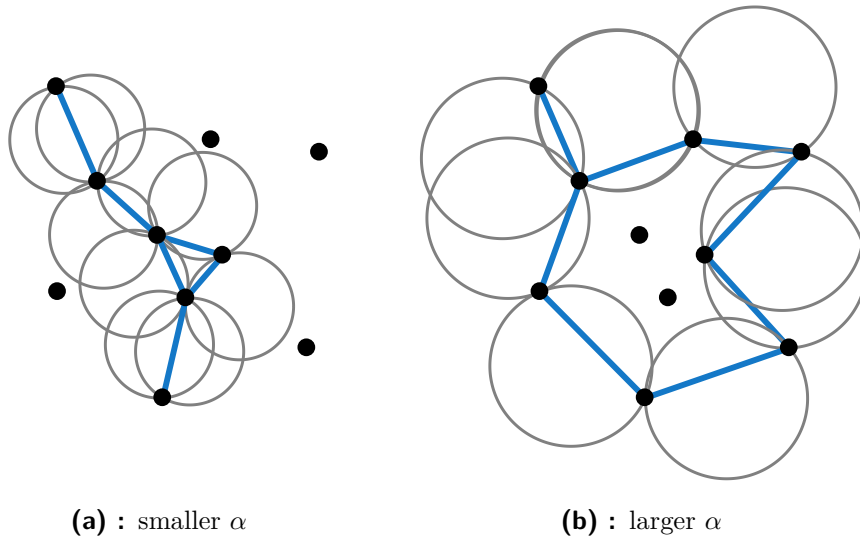
Cells can be stored in a tree structure with different resolutions on each layer [2]. This multi-resolution approach is not used in our robot.

Maps described above are usually stored in some type of pose graph structure. Multiple scans are merged based on the transformation from odometry data and create a frame, which is with its pose stored like a graph node. The new frame is created at the moment when the reliability of displacement estimation from odometry falls under a certain threshold. Nodes are connected by odometry edges containing transformations between frames. The loop closure algorithm creates new edges between the existing nodes.

Assuming, that the CAD drawing is precise, we create only the single frame containing the whole map. Consequently, the pose graph contains only single node. The correctness of this statement should be experimentally validated in future.

### 3.3 Alpha shape

Alpha shape is in this thesis used for obtaining outlines of walls from CAD drawing and assuring that the outlines are watertight. Alpha shape of a finite point set was introduced by Edelsbrunner et al. in [11]. It can be created by inserting discs in 2D or spheres in 3D of the radius defined by  $\alpha$  parameter between the points in the way when all discs are tangential to some point, and none of them contain any point. Connecting all pairs of points, that have a common tangential disc to straight lines creates an Alpha shape of the set. Nice explanation by the help of ice-cream and carving spoon is included in [12]. The examples of an Alpha shape of some point set for two general  $\alpha$  values are in Figure 3.3.

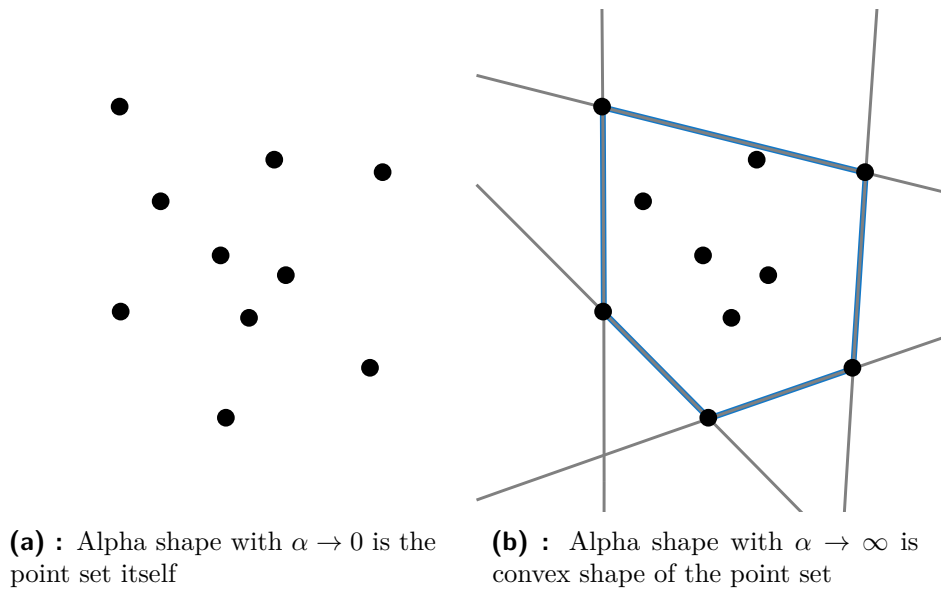


**Figure 3.3:** Alpha shapes of the same point set for two different  $\alpha$  parameters

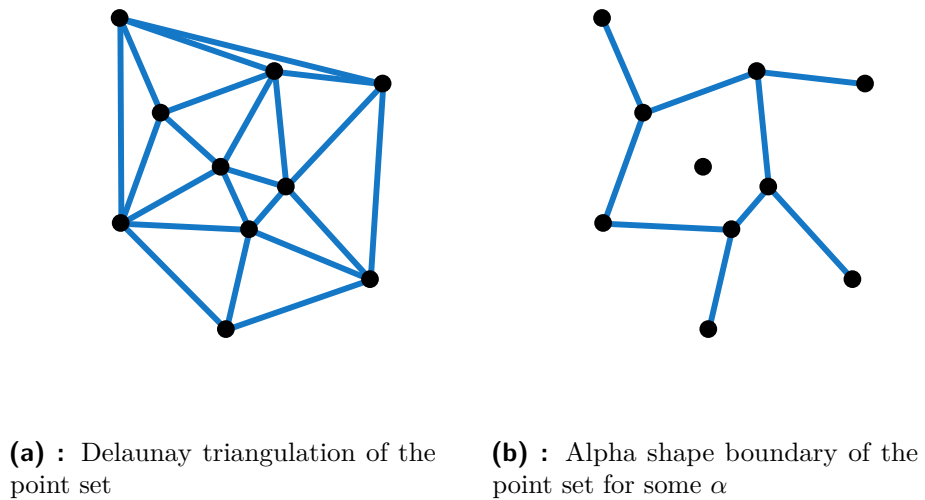
The disc size is being defined variously. The original work [11] defines radius of discs as  $\frac{1}{\alpha}$ , the Geometric algorithms class handout from Stanford university [12] as  $\alpha$  and CGAL (The Computational Geometry Algorithms Library) [14] as  $\sqrt{\alpha}$ . This thesis and the converter implementation uses the second mentioned practice and enters the second power of stored  $\alpha$  parameter as input into CGAL.

Interesting is the case with value  $\alpha \rightarrow 0$ , which leaves the point set as it is and does not connect any points and the case with value  $\alpha \rightarrow \infty$ , creating convex shape. Both situations can be seen in Figure 3.4.

Alpha shape boundary is always a subset of Delaunay triangulation of the same set of points. This property is used for effective Alpha shape construction. The algorithm is clarified in [15]. Similar method uses the CGAL library and its `Alpha_shape_2` class [14]. The relation between the Delaunay triangulation and an Alpha shape of a point set can be seen in Figure 3.5.



**Figure 3.4:** Alpha shapes of the same set of points for the corner  $\alpha$  values



**Figure 3.5:** Alpha shape boundary is always a subset of Delaunay triangulation

All the points from the set must be from the definition in general position (none four points cocircular, none three points collinear [11]). The collinearity of points stops us also from using the construction algorithm mentioned above, because some triangles in Delaunay triangulation may have zero area.

## 3.4 Axis-aligned bounding boxes

Original algorithm [18] creates virtual box around each object with its edges aligned with axis of a coordinate system. This box defines minimal and max-

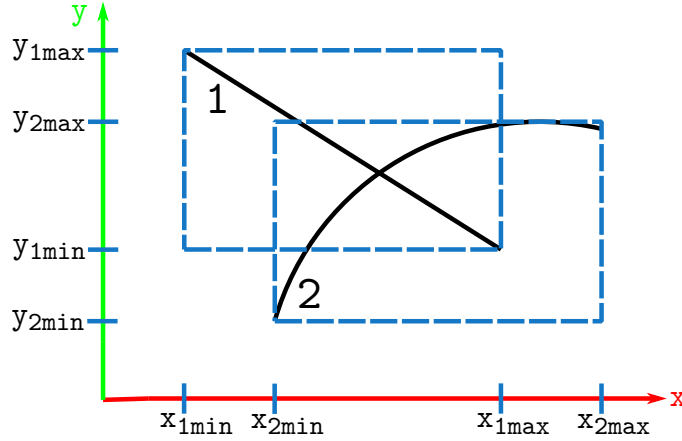


Figure 3.6: Axis-aligned bounding boxes collision detection

imal values of coordinates where we can find the bounded object. Searching for a collision between two objects is replaced by searching for a collision between their bounding boxes, and it can be done by checking four conditions (in 2D)

$$x_{1min} \leq x_{2max}, \quad (3.5)$$

$$x_{2min} \leq x_{1max}, \quad (3.6)$$

$$y_{1min} \leq y_{2max}, \quad (3.7)$$

$$y_{2min} \leq y_{1max}. \quad (3.8)$$

If all of these conditions are met, the objects bounding boxes collide and must have some intersection. The situation can be seen in Figure 3.6. The algorithm used in the converter for finding intersections between NDT grid and geometric entities using bounding boxes is described in section 5.3.

## 3.5 RViz

RViz is ROS package, used for visualization of robot state, surrounding environment and various operational data [19]. All viewable objects can be added to 3D view like "displays." Each display can read data from ROS topics and view them in real time. The visualization can also be used for robot control.



## Chapter 4

# Converter from DXF drawing to NDT map representation

This chapter describes the implementation of the converter from DXF CAD drawing file to NDT map used in the SLAM algorithm. Source code of the implemented converter is present on the attached CD.

### 4.1 Program overview

The main purpose of the converter is to create an NDT map representation from given CAD architectural drawing to DXF format. The NDT map is exported like a simple list of NDT cell parameters in a text file and has to be further converted for usage inside some particular robot.

The converter is dependent on the initial setup by a user. The setup is done by filling a configuration file. The user has to choose parts of the drawing usable for conversion and give them in the form of a list of desired layers to the converter. The user has to know the proper NDT grid settings and set them up in the configuration file. The configuration file also contains multiple parameters dedicated to the elimination of redundant lines inside the walls.

The converter can generate multiple visualization files. The "control views" are CAD drawings which contain different states of conversion and serve to inspection if the conversion goes well. Another type of visualization file is the file with drawing geometry definitions. This file can be loaded by implemented ROS package, and the loaded drawing can be visualized in RViz.

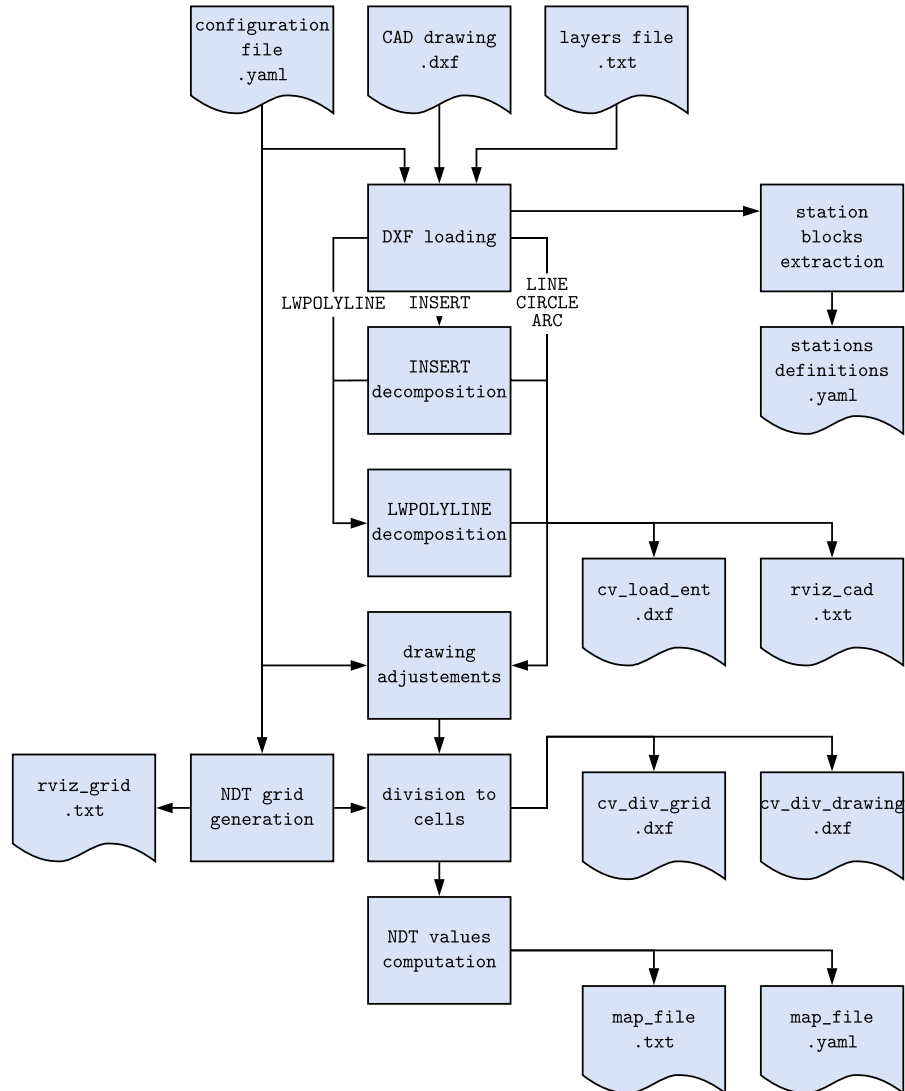
The destinations of robot movement can be drawn in the map as station blocks. The converter is able to extract the station blocks from a defined layer and export YAML format file with station definitions.

### 4.2 Implementation

#### 4.2.1 Implementation of converter from DXF CAD drawing to NDT map representation

This section deals with the implementation of the converter from DXF CAD drawing to NDT map representation. The structure of the implementation is

shown in Figure 4.1. The entire converter and RViz viewer were implemented in C++.



**Figure 4.1:** Converter program structure

The whole process starts with the loading of a configuration file according to the path given by a user. `yaml-cpp` library [17] was used for all manipulation with YAML format files. The configuration values are stored and further used within the program.

Simple DXF reader loads selected entities and all blocks from the CAD drawing specified in configuration file. The station blocks are separated and exported as described in section 4.2.2. It eliminates all other entities except `INSERT` entity type. Next step is decomposition of complex entities. `INSERT`s are linked together with their blocks and decomposed to other entities. `LWPOLYLINE`s are decomposed to `LINE`s and `ARC`s. At this



moment, we have only "primitive" entities (`LINE`, `CIRCLE`, `ARC`). Those are being exported to `cv_load_ent.dxf` and `rviz_cad.txt` for control and visualization purposes.

Now the adjustments are applied. We can delete all diagonal lines, or apply Alpha shape with use of CGAL [14] library.

Previously stored parameters of NDT grid like grid size, position, orientation and cell size are used to initialize the grid variable. Grid lines definitions are exported to `rviz_grid.txt`. Each entity is divided into grid cells. `LINE` entities use adjusted axis-aligned bounding box collision detection algorithm to improve time complexity of the division. Divided entities can be exported to `cv_div_drawing.dxf` and `cv_div_grid.dxf` files.

Divided entities are sampled to receive equivalent to points measured by range scanner. Each cell contains sampled points, from which the NDT parameters can be computed according to 3.3 and 3.4. Eigendecomposition of covariance matrix reveals ratio between eigenvalues, which is adjusted to defined minimal value if necessary. Eigenvectors are modified to be orthogonal. The covariance matrix is assembled again, and NDT parameters are saved into the cell. Computed NDT values can be exported to map files.

#### ■ 4.2.2 Implementation of station blocks export

The stations can be inserted into drawing by a user as described in section 4.3.5. The station blocks are inserted in a drawing on a special layer defined in a configuration file by `ctrl_stations_lay` key.

All `INSERT` entities on defined stations layer are loaded during DXF loading, and their corresponding blocks are found. The placed `INSERT` entity defines position and orientation of the station, and its block contains a visual representation of the station. Additional information like station type or name is stored in block description.

The loaded stations are exported in YAML format. The exact form of the file is described in section 4.3.5.

#### ■ 4.2.3 Implementation of RViz visualization

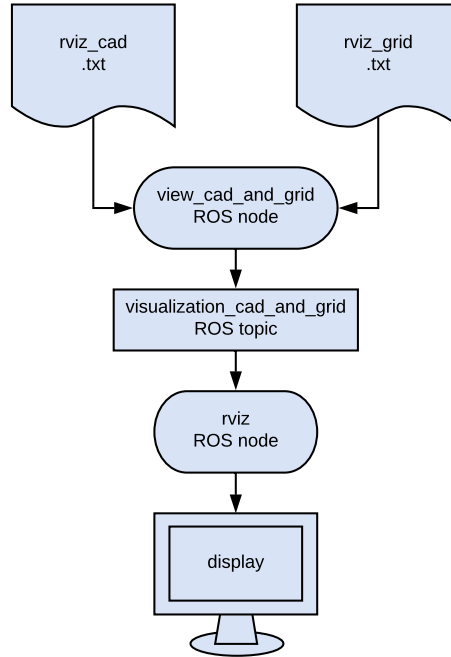
The converter creates two files dedicated to visualization in RViz. The first one's name is `rviz_cad.txt` and contains definitions of geometry objects from CAD drawing loaded by the converter. The second one is `rviz_grid.txt` and contains definitions of NDT grid lines used in the converter. A visualization of NDT map is not a part of this work, because it is being developed by the students of A3M99PTO class.

All geometry objects defined in `rviz_cad.txt` are lines. The arcs and circles from loaded CAD drawing have to be divided into line segments because the `rviz/DisplayTypes/Marker` message used for visualization in RViz does not support any object type for arc or circle. The division to line segments is done in the converter.

Package `view_cad` containing executable `view_cad_and_grid` was implemented. The executable loads both generated files mentioned above and

transfers the defined lines to `Marker::LINE_LIST` rviz display type. The `Grid` rviz display type was not used for the grid, because it can be only square and we need a grid with generally different number of rows and columns.

The executable creates `visualization_cad_and_grid` topic. RViz can subscribe from the created topic and visualize the `Marker` messages. The whole process of visualization is shown in Figure 4.2.



**Figure 4.2:** Structure of RViz visualization process

The coordinate frame of the visualized drawing is called `cad_frame` and corresponds to coordinate frame of the loaded DXF drawing.

## 4.3 User manual

We have to use several programs from the start of the conversion process to its end. The first one is a CAD editor. It is used for viewing the original CAD drawing, choosing the desired layers, conversion to DXF format and insertion of station blocks. It is also used for inspection of exported "control view" files.

A simple text editor can be used to create a configuration file or write a list of desired layers. Advanced text editors like Notepad++ can highlight YAML format syntax and can prevent possible mistakes in formatting.

The converter itself is the program without any graphical interface. It runs from a terminal and is controlled mainly by parameters in the configuration file. It loads the defined DXF file, configuration file, and file with desired layers. Then it creates inspection DXF files, geometry definitions files, the NDT map files and stations definitions file.

ROS RViz can be used for visualization of CAD drawing loaded to converter and NDT grid defined by a user. The data viewed in RViz are loaded from topic published by `view_cad` executable. The `view_cad` loads geometry definitions files (`rviz_grid.txt` and `rviz_cad.txt`) generated by the converter.

### ■ 4.3.1 Installation

The converter is dependent on two libraries which are not present in the standard C++ installation. The first library is `yaml-cpp` which allows to load and export files in YAML format. It is available on GitHub [17]. The second library is `CGAL`, which is in the converter used for Alpha shape creation. It can be downloaded from the main page of the CGAL project [13]. Other libraries have to be installed before CGAL library, namely `CMake`, `Boost`, `Gmp` and `Mpfr`. These CGAL dependencies may or may not be present on a users computer. After installation of all needed libraries the converter can be built by `make` command.

The RViz visualization is independent of the converter. ROS Kinetic and RViz package have to be installed to use the `view_cad` package. The `view_cad` package should be moved to some initialized catkin workspace and built by `catkin build` command.

### ■ 4.3.2 Example of conversion

First of all, we need to prepare the CAD drawing. It has to be saved as a `.dxf` file, which can be done in AutoCAD by pressing "Save As," "Other Formats," and choosing one of the `.dxf` formats from the drop down menu. The converter was designed in accordance with AutoCAD 2012 DXF Reference [16]. AutoCAD 2018 DXF was tested and works well. After we have DXF file of our CAD drawing, we can set the `dxf_file` key in the configuration file. The configuration file is described in detail in section 4.3.3.

The second step is choosing the layers we want to convert. Approved practice is turning off all layers and then turning on one by one, looking for valuable parts of the map, especially walls. The desired layers can be given to the program by two different ways. The first one is writing the layers names in a separate `.txt` file, each on a new line and setting `layers_file` value. The second one is writing the names as YAML list directly to the configuration file under `layers_list` key.

Now we have to set the parameters of the NDT grid. It is good to turn on all desired layers and try to determine the best position, orientation and size of the grid, so it covers whole part of the map, we want to use. We set the `origin`, `rotation` and `grid_size` configuration parameters. The `cell_size` value depends on a requested level of NDT map detail. We can also set minimal ratio between covariance matrix eigenvalues `eig_ratio`, weight of occupied cell `weight` and sampling density for NDT values calculation `spm`.

If the desired layers contain some redundant entities inside the wall, we can delete them manually in CAD editor, or use one of the conversion settings. First is `rem45`, which removes all lines angled by 45 degrees. This option

is handy when the only 45 degrees angled lines are diagonal wall hatches. Otherwise, it can delete some desired parts of the map. The second option is `alpha_shapes` described in 3.3. The  $\alpha$  parameter can be controlled by `alpha_value`. Sampling density (samples per meter) can be set directly by `alpha_spm`, or it will be computed after setting the minimal parallel line distance `pl_distance`.

The NDT map output can be stored in `.txt` file and `.yaml` file. To enable the desired output, `mapfile_txt` or `mapfile_yaml` should be set to 1. The path of the output file can be changed in `map_path_txt` and `map_path_yaml`. The structure of output file is described in 4.3.4.

The correctness of drawing manipulations in the converter can be checked in inspectional `.dxf` files. The file `control_view_load_ent.dxf` containing the loaded drawing can be generated by `control_view`. The division of loaded entities into NDT cells can be checked in `control_view_div_drawing.dxf` (in `.dxf` drawing coordinate system) and in `control_view_div_grid.dxf` (in NDT grid coordinate system) by adjusting `cv_division_drawing` and `cv_division_grid`. If Alpha shape is used, `control_view_alpha.dxf` file is generated automatically.

After we set up the configuration file, we can run the converter.

```
./bin/dxf2ndt
```

It asks for the path to the prepared configuration file and starts the conversion. In the end, the output files defined by the configuration should be created.

### 4.3.3 Configuration file

The whole conversion is controlled by parameters in one YAML format configuration file. A path and a name of the configuration file is specified by a user at the beginning of the converter run. List of the configuration parameters follows.

- `dxf_file` — path and filename of the input `.dxf` file from a running directory ( `string` )
- `layers_file` — path and filename of the layers `.txt` file containing names of desired layers (each on new line) ( `string` )
- `layers_list` — YAML style list of desired layers names - overrides `layers_file` if are both present ( `string list` )
- `origin` — origin of NDT grid in CAD drawing coordinate system ( `[double, double, double]` )
- `rotation` — rotation of NDT grid against CAD drawing coordinate system ( `double` )
- `grid_size` — size of the NDT grid ( `[double, double, double]` )
- `cell_size` — length of side of square NDT cell ( `double` )

- `eig_ratio` — the smallest possible ratio between the NDT covariance matrix eigenvalues ( `double` )
- `weight` — the weight to be set for the occupied cells ( `integer` )
- `spm` — samples per meter value for sampling before NDT values calculation ( `double` )
- `cv_load_ent` — turns on generation of `.dxf` file with loaded part of the CAD drawing ( `1,0` )
- `cv_div_drawing` — turns on generation of `.dxf` file with drawing divided into NDT cells in original CAD coordinate system ( `1,0` )
- `cv_div_grid` — turns on generation of `.dxf` file with drawing divided into NDT cells in NDT grid coordinate system ( `1,0` )
- `mapfile_txt` — turns on creation of `.txt` output NDT map file ( `1,0` )
- `map_path_txt` — path to `.txt` output NDT map file ( `string` )
- `mapfile_yaml` — turns on creation of `.yaml` output NDT map file ( `1,0` )
- `map_path_yaml` — path to `.yaml` output NDT map file ( `string` )
- `rem45` — removes all lines angled by 45 degrees against original CAD drawing coordinate system ( `1,0` )
- `alpha_shapes` — apply the Alpha shape on the drawing ( `1,0` )
- `alpha_value` —  $\alpha$  value for the Alpha shape ( `double` )
- `pl_distance` — the smallest distance between any parallel line in the wall and wall line — serves to computation of Alpha shape sampling density 5.2 ( `double` )
- `alpha_spm` — directly sets the Alpha shape sampling density (instead of computation) ( `double` )
- `gen_rviz` — generates `.txt` files containing definition of loaded CAD drawing and defined NDT grid for visualization in RViz ( `1,0` )
- `ctrl_station_lay` — name of CAD layer containing stations for robot control ( `string` )

Both LF and CR+LF newline characters should work in the configuration file.

#### 4.3.4 Output file

The output file can be generated like simple .txt file or .yaml file. Files are nearly identical, only the cell values in the .yaml file are stored in YAML style array. Each cell is on one line, defined by 14 numbers. All generated files use LF newline character.

Parameter	Data type
mean vector	3× double
covariance matrix	9× double
occupancy	double
weight	integer

**Table 4.1:** Parameters describing each NDT cell in output file

#### 4.3.5 Stations

A user can place some destinations of robot motion into drawing in the form of stations. Each station must be defined by new drawing block to have a unique name. A user draws a visual representation of station in CAD editor and creates a new block from the drawing. A type and name of the station are defined during the block creation in the block description. The description form was chosen to be similar to YAML key-value format as shown in the following example.

```
type: charging station
name: lab 33
```

A user should pay attention to the position of the block base point in the drawn visual representation because the base point corresponds to the exported position of the station.

A separate layer for the stations has to be created in the drawing. Name of the station layer has to be specified in the converter configuration file as a value to the `ctrl_stations_lay` key. All the blocks placed in the layer are identified as stations. This approach can be used in the future for other objects used for planning purposes. Each type of objects like corridor, preferred path or various zones can have own layer.

When we have defined the station block and created a station layer, we can place the block into the drawing to the defined layer by `INSERT` entity. The `INSERT` defines position and orientation of the station.

After placing all the stations, we can run the converter and the `stations_export.yaml` file is generated. The file contains the definitions of all placed stations as shown below.

```
- control_entity: station
  station_number: 4
  type: charging station
  type_number: 1
```

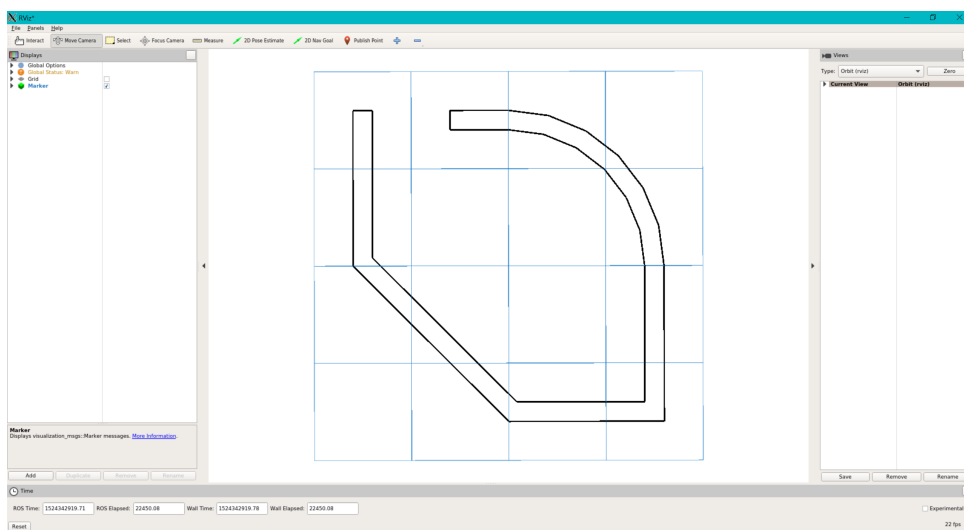
```
name: lab 33
position: [57.012, 12.704, 0]
rotation: [0, -1, 0, 1, 0, 0, 0, 0, 1]
```

- `control_entity` — a type of control entity (now only station)
- `station_number` — a serial number unique between all the stations
- `type` — a user-defined type of the station, e.g. charging station, materials loading, printer
- `type_number` — a serial number unique between all stations of the same type
- `name` — a user-defined name of the station
- `position` — the position of the station (position of the `INSERT` entity) in the coordinate system of the drawing
- `rotation` — the rotation matrix defining station orientation (orientation of the `INSERT` entity) in the coordinate system of the drawing

A user can parse the generated file and use it in a motion planning software.

### 4.3.6 Visualization of CAD drawing in RViz

By enabling generation of files for visualization in RViz, the converter makes one file with line definitions of loaded CAD and one file with created NDT grid. Both can be viewed using implemented ROS package. The executable asks for generated files and publishes `Marker::LINE_LIST` ROS messages, which can be added in RViz like new Marker display. An example of the visualization is in Figure 4.3.



**Figure 4.3:** Visualization of loaded CAD drawing and NDT grid in RViz

#### 4. Converter from DXF drawing to NDT map representation

To use the viewer, the package should be built in some catkin workspace and launched.

```
catkin build
roslaunch view_cad view_cad_and_grid
```



## Chapter 5

### Practical observations

#### 5.1 Imperfections in CAD drawings

The CAD drawing needs multiple adjustments before it can be processed by a parser. Mainly, we need to choose parts of the map important for SLAM. These parts have to be static and physically present in the environment so that the robot can scan them. This requirement leads to an elimination of all non-stationary objects like furniture and doors and all invisible or virtual objects like wirings, texts, and various architectonic signs. The major part of these objects can be simply removed at once thanks to their division to drawing layers or by selection of chosen drawing entities. But some undesired objects can be in the same layer as the needed and can be represented by one of the requested entity types.

The selection of desired layers is done by a user, who has to search the drawing for elements usable for SLAM and deliver their layers to the converter. This process is described in section 4.3.2. It is good to mention that we cannot select the layers just by their name, because the designer could make mistakes.

The extraction of certain drawing entities is done by the converter. It takes from the drawing only `LINE`, `CIRCLE`, `ARC` and `LWPOLYLINE` entities. It also extracts preceding entities from blocks, placed in the drawing by `INSERT` entity.

##### 5.1.1 Hatches

The objects, which are the most difficult to eliminate, are those on the desired layers and sketched by the several extracted entities. The encountered example is wall hatches composed of individual lines. The whole hatch inside a wall can be saved under one `HATCH` entity in DXF format, so the problem can be prevented in the stage of drawing. The decomposition of hatch objects into individual `LINE` entities can happen during conversions between different CAD editors. When we obtain the drawing with already "decomposed" hatches, we can delete them manually, or use some automation.

The simplest criteria common to all encountered hatch lines was their diagonality in the sense of rotation by  $\pi/4$  radians against the world coor-

dinate system (WCS) base. The elimination of all the diagonal lines was implemented and can be turned on by `rem45` key in the configuration file. But the usage is strictly limited to the situations when none `LINE` entity from desired layers is diagonal, and moreover, it does not register any of the non-diagonal hatches.

More universal is the algorithm mentioned in [9]. It finds the first wall by casting a ray from a position of room number text and then continues around the room and saves the walls. The similar algorithms are called boundary tracing. This algorithm wasn't implemented, because of a few disadvantages. Mainly, it can't handle the room with any objects inside and not connected to the walls, e.g. pillars. The initial ray can intersect a pillar earlier than a wall, which results in a failure. Furthermore, it should need some adaptation to non-watertight (can have gaps between lines) drawings.

The most promising treatment of hatches seems to be the use of Alpha shapes, further described in 3.3 and 5.2.

Defects like misplacement or absence of some walls can always appear in the drawing. These defects have to be corrected by a user in a CAD editor because there is no way of recognizing them as errors automatically without knowledge of the real environment.

## 5.2 Adjusting the drawing by Alpha shapes

Using Alpha shape, presented in 3.3, to extract only wall outlines is one of the solutions of the "decomposed" hatches elimination.

The constraint of the general position of points in a point set mentioned in 3.3 are critical for use on straight walls because all points on one side of a wall are collinear. We are adding negligible (e.g. units of micrometers) random noise to every sample point, so it does not influence the precision of a drawing too much, but moves all points to general positions.

The selection of  $\alpha$  value for use on the CAD drawing depends on several factors. Firstly, it should not be bigger than half of the robot width, or the shape will close the gaps usable for robot motion. Secondly, it should be so large that the shape eliminates the wall hatches. The second condition is very dependent on the density of lines sampling. An example of correct use of Alpha shape on diagonal hatches is shown in Figure 5.1.



**Figure 5.1:** Application of Alpha shape on a diagonal line hatch inside a wall - original CAD drawing is shown in black, remaining part of the drawing after application of Alpha shape is shown in blue

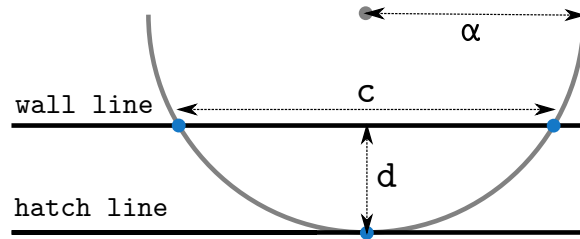
We can encounter a hatch or a line inside a wall and parallel to the wall line in the distance of  $d$ . The worst possible case is, when the samples of the hatch are shifted by half a sampling distance  $c$  from the samples of the wall as shown in Figure 5.2. To prevent integration of the hatch samples to the Alpha shape boundary, the sampling distance must meet the equation

$$c < \sqrt{8d \left( \alpha - \frac{d}{2} \right)}. \quad (5.1)$$

This requirement is based on the relation of parameters of a circular segment

$$\alpha = \frac{d}{2} + \frac{c^2}{8d}, \quad (5.2)$$

which can be seen in Figure 5.2.



**Figure 5.2:** Calculation of sampling density for Alpha shape use - sampling distance  $c$ , spacing of lines  $d$  and parameter of Alpha shape  $\alpha$

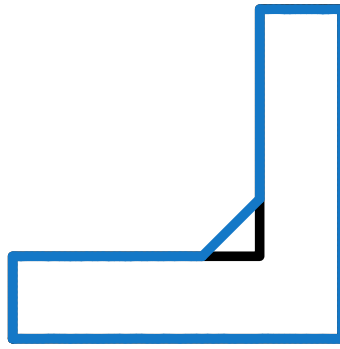
An example of Alpha shape usage on undersampled parallel lines is shown in Figure 5.3.



**Figure 5.3:** Application of Alpha shape on parallel lines inside a wall (undersampled) - original CAD drawing is shown in black, remaining part of the drawing after application of Alpha shape is shown in blue

To prevent cutting of the outer wall corners, the end points of each line are inserted to the point set.

Besides the elimination of redundant entities inside the walls, Alpha shape distorts the inner wall corners as shown in Figure 5.4. The distortion increases with  $\alpha$  value.

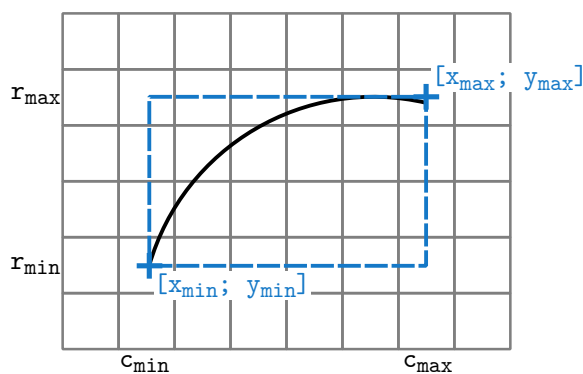


**Figure 5.4:** Corner distortion as result of Alpha shape application - original CAD drawing is shown in black, remaining part of the drawing after application of Alpha shape is shown in blue

### 5.3 Division of entities to cells

We need to divide entities into cells before the NDT values computation. The division is done by searching for intersections between NDT cells outlines and entities. The simplest approach is trying to find the intersection between each entity and each cell. This practice was machine time consuming, therefore, was replaced by another method inspired by axis-aligned bounding boxes algorithm 3.4.

We want to check the collision of entities with square cells. We have the advantage of easy calculation of the position of each cell and edges from the NDT grid definition parameters, and the cells can be iterated over rows and columns of the grid. We create the bounding box of an entity transformed to NDT grid coordinate system and simply compute the cells, which contain corner points of the bounding box  $[x_{min}; y_{min}]$ ,  $[x_{max}; y_{max}]$ . These cells are in rows  $r_{min}$  and  $r_{max}$  and columns  $c_{min}$  and  $c_{max}$  respectively as shown in Figure 5.5. Then we search for intersection only between the entity and cells with row number in the interval  $\langle r_{min}, r_{max} \rangle$  and column number in  $\langle c_{min}, c_{max} \rangle$ .



**Figure 5.5:** Axis-aligned bounding box of an entity and the grid - intersection is checked only between the entity and cells with rows in the interval  $\langle r_{min}, r_{max} \rangle$  and columns in the interval  $\langle c_{min}, c_{max} \rangle$

## Chapter 6

### Experimental results

All the implemented software was tested on architectonic drawings of CIIRC building in Prague. This chapter proceeds one of the conversions of the drawings and results of the conversion.

The Figures 6.1a 6.2a 6.3a contain original drawing with all layers turned on. Desired layers can be seen in Figures 6.1b 6.2b 6.3b. The list of desired layers was written into individual text file.

The configuration file was filled as shown below.

```
dxfile: user/CIIRC/6/ciirc_6.dxf

layers_file: user/CIIRC/6/layers_ciirc_6.txt

origin: [80000, -15000, 0]
rotation: 0
cell_size: 250
grid_size: [125000, 20000, 0]
eig_ratio: 0.001
weight: 20
spm: 10

alpha_shapes: 1
alpha_value: 150
pl_distance: 20

mapfile_txt: 1
map_path_txt: user/CIIRC/6/mapfile_ciirc_6.txt

gen_rviz: 1

ctrl_stations_lay: jackal_stations
```

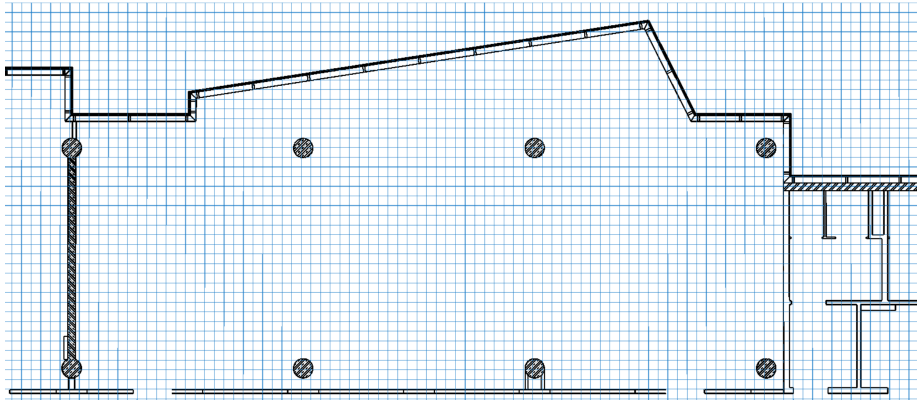
The drawing contains the hatches divided into individual lines as a consequence of some format conversions. The line hatches were eliminated using Alpha shape algorithm. All unwanted lines in walls were eliminated, but corner cutting effect appeared. It can be seen on corners in Figure 6.2c and on corners and right column in Figure 6.3c.

The Figures 6.1c 6.2c 6.3c contain generated NDT representation of the environment. We can see that some NDT cells contain both sides of one wall as middle wall and walls around the elevator in Figure 6.2c. This case can be eliminated by better placement of the grid and decrease of the cells size.

Some stations were created and placed on new `jackal_stations` layer. The stations can be seen in Figures 6.1a 6.1b. Their description was written in form shown in section 4.3.5. The converter successfully generated YAML file with stations definitions.

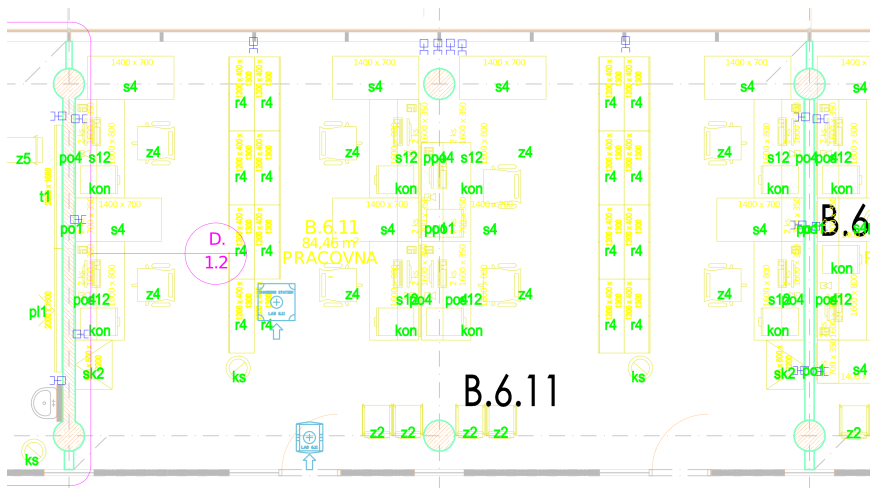
```
- control_entity: station
  station_number: 3
  type: stop
  type_number: 3
  name: lab
  position: [57.546, 10.509, 0]
  rotation: [0, -1, 0, 1, 0, 0, 0, 0, 1]
- control_entity: station
  station_number: 4
  type: charging station
  type_number: 1
  name: in lab
  position: [57.012, 12.704, 0]
  rotation: [0, -1, 0, 1, 0, 0, 0, 0, 1]
```

The export files for RViz were generated. The files were processed by `view_cad_and_grid` ROS node and viewed in RViz. The RViz display can be seen in Figure 6.4.

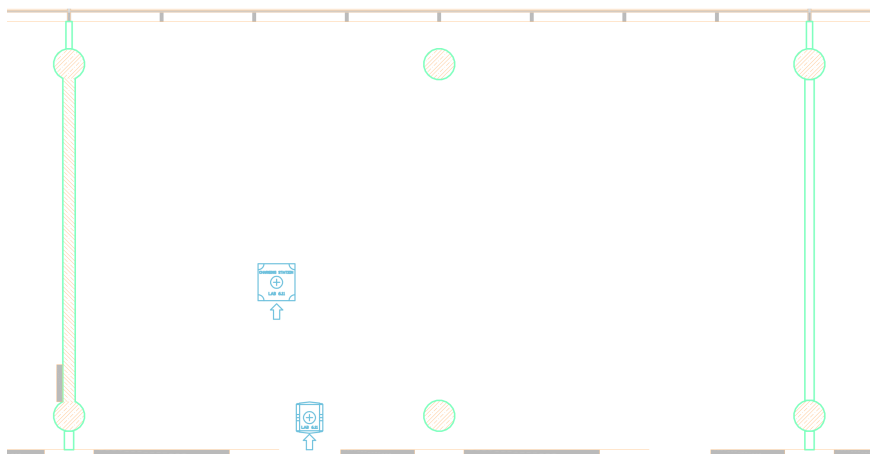


**Figure 6.4:** Verification of RViz drawing visualization

Experimental evaluation of the generated NDT maps using Jackal robot was not possible, because we are not able to import the NDT maps to the robot. Problems with positioning of the map in the robot coordinate system, importing data into the graph representation and starting the mapping algorithm have to be solved.



(a) : Original architectonic drawing



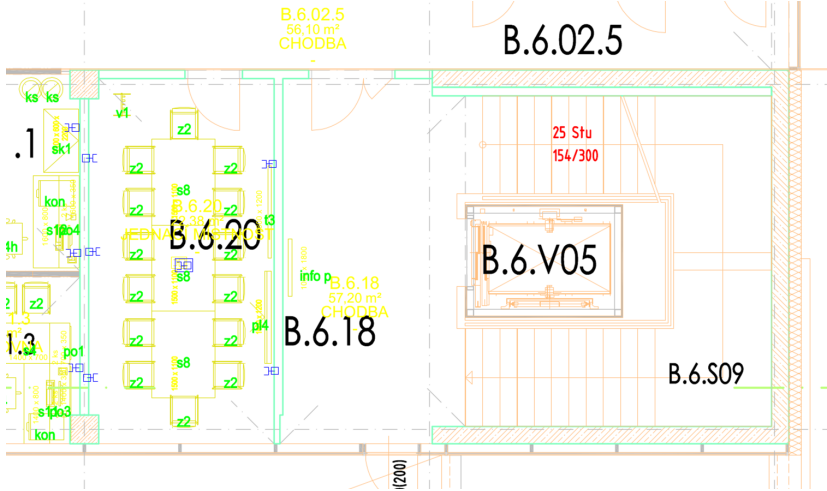
(b) : Drawing with desired layers



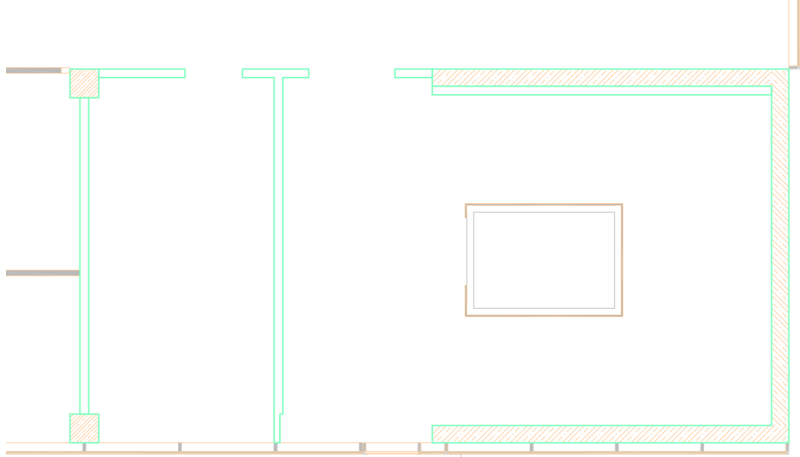
(c) : Generated NDT map

Figure 6.1: Cut of the map: laboratory

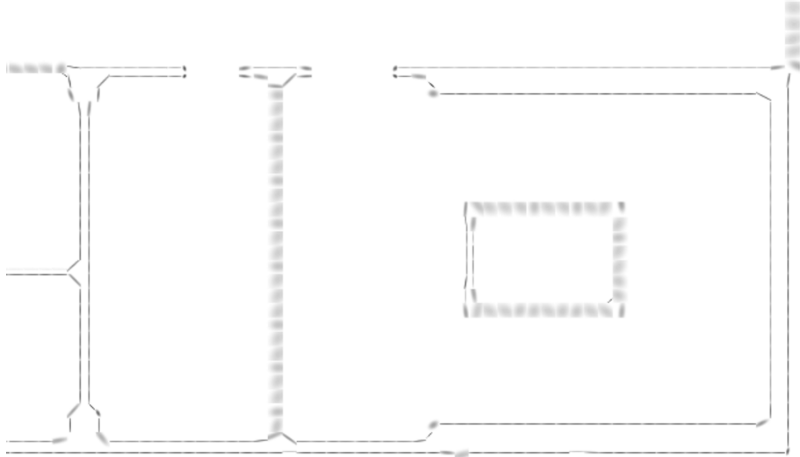
6. Experimental results



(a) : Original architectonic drawing



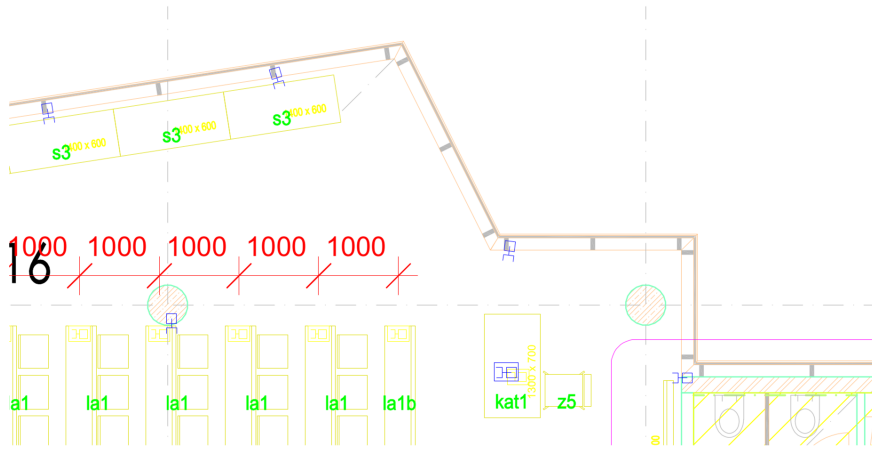
(b) : Drawing with desired layers



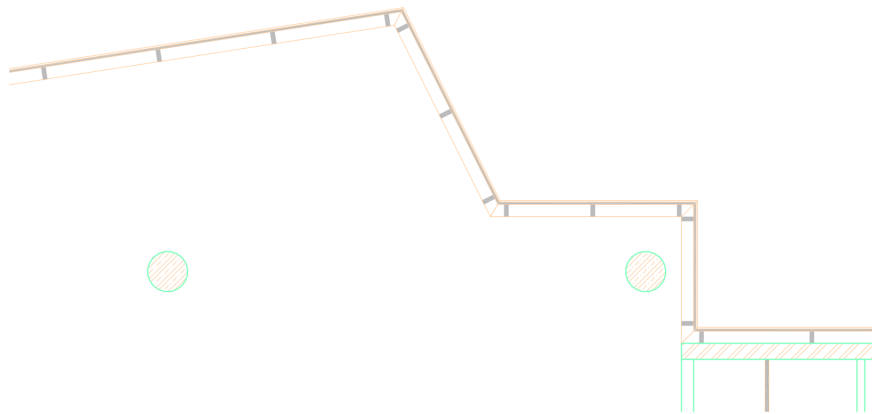
(c) : Generated NDT map

Figure 6.2: Cut of the map: stairs

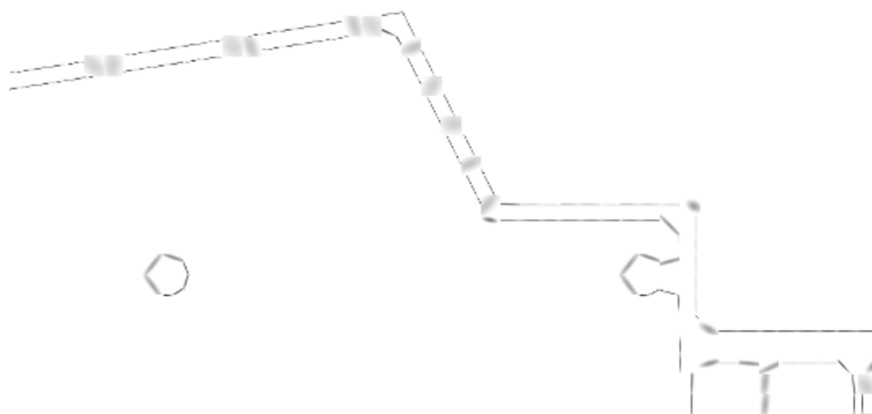




(a) : Original architectonic drawing




(b) : Drawing with desired layers



(c) : Generated NDT map

Figure 6.3: Cut of the map: lecture hall





## Chapter 7

### Conclusion

This thesis has dealt with a conversion of CAD architectonic drawings to NDT map representation. Publicly documented DXF format of CAD drawing has been chosen and the converter from this format to the NDT map representation has been implemented.

The CAD drawings can be visualized in RViz by the implemented ROS node. The converter contains the possibility of diagonal lines elimination or wall outlines extraction by Alpha shape algorithm. The converter can export definitions of user-defined movement destinations in simple YAML format file. The main output of the converter is a simple text file with definitions of NDT cells.

The conversion has been tested on the architectonic drawings of CTU CIIRC building. The results of conversion have been evaluated only visually because the present state of Jackal robot software does not provide a functional NDT map import. Once the map import to the Jackal robot is operational, the generated maps will need to be tested, and we will have to find proper parameters of NDT grid.

The elimination of undesired parts of the map could be improved in the future. The elimination of diagonal lines works only on special cases, and Alpha shape algorithm cuts corners and connects near objects. Using Alpha shape only for a selection of boundary lines or some boundary tracing algorithm should satisfy our demands. Elimination of outdoor wall lines, which indoor robot cannot scan, could also be implemented.





## Bibliography

- [1] Peter Biber, Wolfgang Straßer, *The Normal Distributions Transform: A New Approach to Laser Scan Matching*, IEEE IROS, Las Vegas, USA, 2003
- [2] Erik Einhorn, Horst-Michael Gross, *Generic NDT mapping in dynamic environments and its application for lifelong SLAM*, Robotics and Autonomous Systems, vol. 69, pp. 28–39, July 2015
- [3] David Nováček, *Celoživotní určování polohy mobilního robotu*, Bc. thesis, Dept. of Cybernetics, Fac. of Elect. Eng., Czech Tech. Univ. in Prague, 2017
- [4] Søren Riisgaard, Morten Blas, *Slam for Dummies (A Tutorial Approach to Simultaneous Localization and Mapping)*, 2005, available: [https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam\\_blas\\_repo.pdf](https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf)
- [5] Lukáš Jelínek, *Graph-based SLAM on Normal Distributions Transform Occupancy Map*, Bc. thesis, Dept. of Theor. Comp. Sci. and Math. Logic, Fac. of Math. and Phys., Charles Univ., Prague, 2016
- [6] Jari Saarinen, Henrik Andreasson, Todor Stoyanov, Juha Ala-Luhtala and Achim J. Lilienthal, *Normal Distributions Transform Occupancy Maps: Application to Large-Scale Online 3D Mapping*, IEEE ICRA, Karlsruhe, Germany, 2013
- [7] Jari Saarinen, Todor Stoyanov, Henrik Andreasson and Achim J. Lilienthal, *Fast 3D Mapping in Highly Dynamic Environments using Normal Distributions Transform Occupancy Maps*, IEEE/RSJ IROS, Tokyo, Japan, 2013
- [8] *Clearpath Robotics - Jackal data sheet*, 2014, available: <https://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/>
- [9] Aniket Murarka, Benjamin Kuipers, *Using CAD Drawings for Robot Navigation*, IEEE SMC, 2001

- [10] Adam Borkowski, Barbara Siemiatkowska, Jacek Szklarski, *Towards Semantic Navigation in Mobile Robotics*, Graph Transformations and Model-Driven Engineering - Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday, 2010
- [11] Herbert Edelsbrunner, David G. Kirkpatrick, Raimund Seidel, *On the Shape of a Set of Points in the Plane*, IEEE TIT, vol. 29, pp. 551-559, July 1983
- [12] Kaspar Fischer, *Introduction to Alpha Shapes*, Geometric algorithms class handout, Stanford Univ., 2000, available: [graphics.stanford.edu/courses/cs268-16-fall/Handouts/AlphaShapes/as\\_fisher.pdf](http://graphics.stanford.edu/courses/cs268-16-fall/Handouts/AlphaShapes/as_fisher.pdf)
- [13] *CGAL - main page*, available: <https://www.cgal.org/>
- [14] *CGAL - 2D Alpha Shapes*, available: [https://doc.cgal.org/latest/Alpha\\_shapes\\_2/index.html](https://doc.cgal.org/latest/Alpha_shapes_2/index.html)
- [15] Marjan Celikik, *Alpha shapes*, Computational Topology seminar presentation, Max-Planck-Institut für Informatik, available: <http://www.mpi-inf.mpg.de/~jgiesen/tch/sem06/Celikik.pdf>
- [16] *Autodesk AutoCAD 2012 DXF Reference*, available: [http://images.autodesk.com/adsk/files/autocad\\_2012\\_pdf\\_dxf-reference\\_enu.pdf](http://images.autodesk.com/adsk/files/autocad_2012_pdf_dxf-reference_enu.pdf)
- [17] *yaml-cpp YAML parser and emitter in C++*, available: <https://github.com/jbeder/yaml-cpp>
- [18] *3D collision detection - Axis-aligned bounding boxes*, available: [https://developer.mozilla.org/en-US/docs/Games/Techniques/3D\\_collision\\_detection](https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_collision_detection)
- [19] *ROS.org Wiki - rviz package*, available: <http://wiki.ros.org/rviz>
- [20] Marc Vigo, Núria Pla, Dolors Ayala, Jonàs Martínez, *Efficient algorithms for boundary extraction of 2D and 3D orthogonal pseudomanifolds*, Graphical Models, vol. 74, pp. 61-74, March 2012

## Appendix A

### Example of configuration file

This appendix contains an example of the configuration file in YAML format. Symbol '#' marks comment beginning. Not all keys have to be present in the configuration file. Absent keys are automatically set to some default values. Description of all the keys is present in section 4.3.3.

```
# DXF file path:
dxf_file: arcs_control.dxf

# List of layers:
# Path and name of layers file
layers_file: layers.txt

# YAML style list of desired layers - overrides layers
# from layers_file
layers_list:
- Layer01
- Layer02
- Layer03

# NDT configuration values:
# Origin of NDT grid in DXF coordinate system
origin: [-50, 670, 0]
# Rotation of NDT grid against DXF coordinate system in
# degrees
rotation: 45
# Length of cell side
cell_size: 500
# Grid size [x,y,z]
grid_size: [20000, 24000, 0]
# Minimal ratio between eigenvalues of covariance matrices
eig_ratio: 0.001
# Weight of cells occupied by some entity
weight: 20
```

A. Example of configuration file

```
# Samples per meter - for sampling for computation of normal
# distribution parameters
spm: 10

# Export of control DXF files:
# creates file with loaded and decomposed entities (lines,
# circles and arcs)
cv_load_ent: 0
# creates file with entities divided to cells - in original
# drawing coordinate system
cv_div_drawing: 1
# creates file with entities divided to cells - in NDT grid
# coordinate system
cv_div_grid: 0

# Removes all lines angled by pi/4 from DXF coordinate system
# axes
rem45: 1

# Alpha shapes:
# Turns on the usage of Alpha shape
alpha_shapes: 1
# Value of alpha parameter
alpha_value: 100
# Distance between wall line and parallel line inside
# the wall (the smallest in the drawing) - for calculation
# of sampling density for Alpha shape
pl_distance: 50
# Direct input of sampling density for Alpha shape - parameter
# active when not equal to 0 - overrides calculation from
# pl_distance parameter
alpha_spm: 0

# Turns on the generation of files for RViz visualization
gen_rviz: 1

# Creation of NDT map files:
# Turns the creation of .txt file on
mapfile_txt: 1
# Specification of path and name of .txt NDT map file
map_path_txt: mapfile01.txt
```



```
# Turns the creation of .yaml file on
mapfile_yaml: 0
# Specification of path and name of .yaml NDT map file
map_path_yaml: mapfile01.yaml

# Name of DXF file layer containing stations (motion
# destinations) for robot motion planning
ctrl_stations_lay: robot_stations
```



## Appendix B

### CD content description

- dxf2ndt - converter directory
  - bin - built binary
  - control\_views - exported inspection DXF files
    - └─...
  - include - header .h files
    - └─...
  - input\_files - example DXF and configuration files
    - └─...
  - rviz\_files - exported files for view\_cad
    - └─...
  - src - source .cpp files
    - └─...
  - example\_output\_mapfile.txt - generated example NDT map
  - makefile
  - stations\_export.yaml - generated example station definition file
  - tmpplt.dxf - template DXF used by converter
- view\_cad - directory of view\_cad ROS package
  - include - header .h files
    - └─...
  - src - source .cpp files
    - └─...
  - CMakeLists.txt
  - package.xml