Simultaneous Localisation and Mapping Using Autonomous Target Detection and Recognition

Examensarbete utfört i reglerteknik
vid Tekniska högskolan vid Linköpings universitet
av

Kristian Sinivaara

LiTH-ISY-EX--14/4793--SE

Handledare: Hanna Nyqvist
ISY, Linköpings universitet
Christina Grönwall
Totalförsvarets forskningsinstitut FOI

Examinator: Gustaf Hendeby
ISY, Linköpings universitet

Linköping, 1 september 2014
Simultaneous localisation and mapping (SLAM) is an often used positioning approach in GPS denied indoor environments. This thesis presents a novel method of combining SLAM with autonomous/aided target detection and recognition (ATD/R), which is beneficial for both methods. The method uses physical objects that are recognisable by ATR as unambiguous features in SLAM, while SLAM provides the ATR with better position estimates. The intended application is to improve the positioning of a first responder moving through an indoor environment, where the map offers localisation and simultaneously helps locate people, furniture and potentially dangerous objects like gas cannisters.

The developed algorithm, dubbed ATR-SLAM, uses existing methods from different fields such as EKF-SLAM and ATR based on rectangle estimation. Landmarks in the form of 3D point features based on NARF are used in conjunction with identified objects and 3D object models are used to replace landmarks when the same information is used. This leads to a more compact map representation with fewer landmarks, which partly compensates for the introduced cost of the ATR. Experiments performed using point clouds generated from a time-of-flight laser scanner show that ATR-SLAM produces more consistent maps and more robust loop closures than EKF-SLAM using only NARF landmarks.
Abstract

Simultaneous localisation and mapping (SLAM) is an often used positioning approach in GPS denied indoor environments. This thesis presents a novel method of combining SLAM with autonomous/aided target detection and recognition (ATD/R), which is beneficial for both methods. The method uses physical objects that are recognisable by ATR as unambiguous features in SLAM, while SLAM provides the ATR with better position estimates. The intended application is to improve the positioning of a first responder moving through an indoor environment, where the map offers localisation and simultaneously helps locate people, furniture and potentially dangerous objects like gas cannisters.

The developed algorithm, dubbed ATR-SLAM, uses existing methods from different fields such as EKF-SLAM and ATR based on rectangle estimation. Landmarks in the form of 3D point features based on NARF are used in conjunction with identified objects and 3D object models are used to replace landmarks when the same information is used. This leads to a more compact map representation with fewer landmarks, which partly compensates for the introduced cost of the ATR. Experiments performed using point clouds generated from a time-of-flight laser scanner show that ATR-SLAM produces more consistent maps and more robust loop closures than EKF-SLAM using only NARF landmarks.
## Contents

1 Introduction .......................... 1
   1.1 Background .......................... 1
   1.2 Problem Definition .................. 3
   1.3 Limitations ........................ 3
   1.4 Related Work ....................... 4
   1.5 Available Resources ................ 5
      1.5.1 Sensor: Riegl VZ-400 .......... 5
      1.5.2 Data Set: The Fire Facility .... 6
   1.6 Outline of Thesis ................... 8

2 Theoretical Foundation .......... 9
   2.1 Simultaneous localisation and mapping .... 9
      2.1.1 Model .......................... 9
      2.1.2 Algorithm ....................... 10
   2.2 Feature Extraction ................ 13
      2.2.1 Normal Aligned Radial Feature .... 14
   2.3 Autonomous Target Detection and Recognition .. 17
   2.4 Data Association ................... 19
      2.4.1 Nearest Neighbour ............. 19
      2.4.2 Mahalanobis Distance .......... 20

3 Methodology ........ 21
   3.1 Overview ......................... 21
   3.2 Software ......................... 23
   3.3 Coordinate Systems ................. 23
   3.4 Pre-Processing .................... 25
      3.4.1 Point Cloud Measurements ... 25
      3.4.2 Inertial Input .................. 26
   3.5 EKF-SLAM ......................... 27
      3.5.1 SLAM Model .................... 27
      3.5.2 EKF .......................... 29
      3.5.3 NARF Extraction ............ 30
      3.5.4 Landmark Association ......... 31
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5.5</td>
<td>Map Augmentation</td>
<td>34</td>
</tr>
<tr>
<td>3.6</td>
<td>ATR-SLAM</td>
<td>35</td>
</tr>
<tr>
<td>3.6.1</td>
<td>SLAM Model</td>
<td>35</td>
</tr>
<tr>
<td>3.6.2</td>
<td>EKF</td>
<td>38</td>
</tr>
<tr>
<td>3.6.3</td>
<td>ATR</td>
<td>38</td>
</tr>
<tr>
<td>3.6.4</td>
<td>Object Association</td>
<td>44</td>
</tr>
<tr>
<td>3.6.5</td>
<td>Map Augmentation</td>
<td>45</td>
</tr>
<tr>
<td>3.6.6</td>
<td>Landmark Reduction</td>
<td>47</td>
</tr>
<tr>
<td>3.7</td>
<td>Evaluation</td>
<td>47</td>
</tr>
<tr>
<td>3.7.1</td>
<td>Test One: Single Path</td>
<td>49</td>
</tr>
<tr>
<td>3.7.2</td>
<td>Test Two: Back and Forth</td>
<td>49</td>
</tr>
<tr>
<td>3.7.3</td>
<td>Test Three: Large Scale SLAM</td>
<td>49</td>
</tr>
<tr>
<td>4</td>
<td>Results</td>
<td>53</td>
</tr>
<tr>
<td>4.1</td>
<td>Test One: Single Path</td>
<td>53</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Consistency</td>
<td>53</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Computational Complexity</td>
<td>57</td>
</tr>
<tr>
<td>4.2</td>
<td>Test Two: Back and Forth</td>
<td>60</td>
</tr>
<tr>
<td>4.3</td>
<td>Test Three: Large Scale SLAM</td>
<td>66</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Map Consistency</td>
<td>66</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Computational Complexity</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>Discussion</td>
<td>75</td>
</tr>
<tr>
<td>5.1</td>
<td>Results</td>
<td>75</td>
</tr>
<tr>
<td>5.2</td>
<td>EKF-SLAM Implementation</td>
<td>76</td>
</tr>
<tr>
<td>5.3</td>
<td>ATR-SLAM Implementation</td>
<td>77</td>
</tr>
<tr>
<td>5.4</td>
<td>Future Work</td>
<td>78</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions</td>
<td>79</td>
</tr>
<tr>
<td>A</td>
<td>Parameter Settings</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>83</td>
</tr>
</tbody>
</table>
This master's thesis explores the problem of simultaneous localisation and mapping (SLAM) and the possibility of combining it with autonomous target recognition (ATR). This introduction provides a motivational background to the problem and a definition of the problem itself.

1.1 Background

Positioning systems can provide the ability to locate yourself and build a model of your surroundings, which can be of great importance in many applications. It can help you avoid getting lost and give an improved perception of the environment around you. Simultaneous localisation and mapping (SLAM) [9] is a positioning approach commonly used indoors or underground, where satellite navigation (GPS) is not viable. SLAM is a way to keep track of a sensor carrier while simultaneously building a model of the environment. A common application for such systems is in robotics, where mobile autonomous robots navigate in an unknown environment and possibly interact with certain objects. Another area of application is augmented reality, where this technology is used to enhance human performance in certain situations by quickly providing important information of their surroundings.

Automatic target detection and recognition (ATD/R, or simply ATR) [24] is a general term used for algorithms that can detect and recognise specified targets using sensor data. These targets correspond to objects such as people, vehicles or furnitures. In SLAM, data might also come from different kinds of objects, but no distinction is usually made. The data is instead stored in the form of individual points, called landmarks.
The Swedish Defence Research Agency (FOI) are interested in combining SLAM with ATR for indoor navigation. Merging SLAM and ATR could be beneficial for both methods. The SLAM algorithm would for example benefit from well defined features in the form of physical objects, giving a more robust loop closure. The ATR algorithm could use the SLAM model’s position estimates to improve the chance of making a correct identification. From a user perspective, a map with distinct object models might be easier to interpret than clouds of landmarks.

The case studied in this thesis is where a team of firefighters move into a burning building. This is a hostile environment with poor visibility due to the fire and smoke. Figure 1.1 shows an example of such an environment. Navigation under such dangerous conditions is hard and getting lost can cost lives. This is also indoors, making GPS an unsuitable positioning method for navigation. Instead, SLAM can build a map of the scene by using sensors that are attachable to the firefighters. This thesis will use sets of 3D data from a laser scanner that has been collected by FOI from a fire facility used for training firefighters. Laser scanners are active imaging sensors and unlike cameras, they provide accurate 3D information in the poor lighting conditions present in the fire facility.

A combined SLAM and ATR algorithm could be used to create a rich 3D map of the environment, where different kinds of objects would be explicitly depicted. The map created by SLAM would offer localisation and the objects detected with ATR would provide an increased perception of the surroundings. Such a system could be used not only to help the firefighter navigate through rooms and corridors, but also to help him/her detect people, furnitures and potentially dan-
gerous objects like gas cannisters. Furthermore, this intelligence could also be stored and communicated to other team members.

1.2 Problem Definition

The purpose of this thesis is to explore the potential synergy between SLAM and ATR. The main objective is to develop and evaluate an algorithm that successfully combines elements of SLAM and ATR in order to produce better localisation, mapping and target detection than each method by itself. During this development, the following issues will be investigated:

- Can ATR make the loop closure of the SLAM algorithm more robust? The question partly concerns if objects are less ambiguous features than conventional landmarks, reducing the risk of false loop closures. If the position and orientation of the objects in 3D-space could be determined, then this information could be integrated into the SLAM model.

- Is it possible to replace landmarks in the SLAM model by incorporating object models from the ATR and in doing so, will it reduce the number of landmarks enough to have an impact on the computational complexity introduced by the ATR? This could be performed by a removal of landmarks that occupy the same physical space as the object models.

- Can the information stored in the SLAM model in turn be used to increase the probability of correct object detection in the ATR algorithm? The SLAM model’s trajectory and map could be used to give the ATR initial target position estimates. Previously seen objects could reduce the risk of false positives from new observations that would conflict with the map in the SLAM model.

This requires the development of two separate SLAM algorithms, one using only conventional landmarks and one using both landmarks and objects extracted by the ATR.

1.3 Limitations

The scope of what is covered in this thesis is limited in several ways in order to reduce the complexity of the problem. Because time is a limiting factor, comparisons of different approaches to SLAM and ATR are excluded, even though both are flexible in a variety of ways and a comparison of different methods would be interesting. The focus of this thesis lies instead on the unification of the two. Solving the SLAM problem coincides with the intended purpose of this thesis and is the backbone of the developed algorithm. However, the focus in this work is not on the SLAM algorithm by itself and the implementation is therefore of simple design.

There are a few assumptions and approximations made in this thesis. Unlike
Introduction

reality, the world is assumed to be static, i.e. everything picked up by the sensors is stationary. The ATR algorithm will not handle cluttered environments, a problem deemed to be out of scope for this thesis. Also, the ATR will be partly adapted to information in the data set and not a completely general ATR. This includes information about what kinds of objects there are in the environment, as well as their shape and sizes.

1.4 Related Work

Indoor navigation using SLAM is a well studied problem. Previous work on the subject of first responders moving indoors have been done by Rantakokko et al. [23] using foot-mounted sensors for inertial input. When the first responder entered the building, the system provided meter-level accuracies for a few minutes before drift from the sensors became too high. The article mentions a need for other supporting sensor, including imaging sensors, in order to provide a sufficient level of accuracy. Rydell and Emilsson [26], used EKF-SLAM with both a stereo camera and inertial sensors for indoor navigation. Their experiments showed that the imaging sensor provided good performance under sufficient light conditions, but the system had to otherwise rely on the inertial sensors.

SLAM can be used to register data from separate measurements in order to form a coherent map that can be expanded upon by recognition algorithms. Nüchter et al.[20] use a 3D laser scanner and a variant of ICP-SLAM [21] to create a 3D map from individual scans. Coarse scene features in the map, like walls and floors, are distinguished and semantically labelled. A trained Ada-boost classification algorithm is also used to let the robot recognise and locate objects that are part of the interior. The purpose is to create a semantic map with objects that autonomous robots can reason about, e.g. for symbolic planning. The map can also help disambiguate sensor data and the information becomes reviewable. This is an inspiring approach, although this thesis focuses less on the semantics and more on how the distinct properties of the extracted objects can be used to improve SLAM.

Related work on the subject of semantic mapping and object recognition is proposed by Rusu [25] and Blodow et al. [5], where the goal is to support a robot performing simple tasks in an indoor kitchen environments. Spatial and visual information in the form of point clouds are first collected using calculations of next best view (NBV) poses. They then have 2 levels of feature extraction to separate furniture objects from drawers and kitchen appliances. Kanezaki et al. [12] have a proposed solution for detecting objects in cluttered environments using time of flight (TOF) and a visual sensor. Their robot is able to detect and recognise various objects in an office environment using a sliding box approach.

The ATR algorithm’s introduction of objects to the SLAM model in this work is similar to that of text based features, where parts of the map are annotated with text labels. Case et al. [7] used text detection and recognition modules to extend
a map with semantic labels. Although the map is SLAM generated, it already exists when the autonomous sign reading system starts to operate and how the labels could affect the SLAM is not discussed. Similarly, Posner et al. [22] use boosting and Markov modelling for text spotting in man-made environments. The potential for topic-based navigation is discussed, but the relation to SLAM is not mentioned.

1.5 Available Resources

FOI has a long history of working with 3D mapping using high-resolution laser scanners and other sensors. For this thesis, FOI provides not only 3D data in the form of point clouds collected from a fire facility, but also a point cloud library in Matlab with useful functions for manipulating 3D data.

FOI owns a Riegl VZ-400 laser scanner [14], which was used in the data collecting of the fire facility data set. A laser scanner is used, as opposed to a camera, because it has the advantage of providing accurate range data from the environment, even in the poor light conditions of the fire facility.

1.5.1 Sensor: Riegl VZ-400

The Riegl VZ-400 is a time-of-flight laser scanner, which is depicted in Figure 1.2. The VZ-400 is a scanner with a high target rate that produces high resolution point clouds (3D data). The mechanism consists of a fast rotating polygon mirror and a slower rotating head, which covers a 360° field of view (-40° to +60° in elevation). A Nikon D700 is mounted on the VZ-400 in order to acquire photos for texturing each point in the point cloud with a RGB-value.

The VZ-400 also comes with a software, RiSCAN PRO, which is used to process point clouds into a more manageable format for the kind of methods used in this work. The amount of information contained in the raw point clouds produced by the scanner is more than enough to describe the environment. In fact, they are too dense and need to be down sampled in order to be effectively used by the SLAM and ATR algorithms. The down sampling is done by creating an Octree of the point cloud, which is a data structure that allows the data points to be sorted to a desired resolution equally across the entire data set. The resolution of the raw point cloud depends on the maximum range of the scanner and points that are too far away cannot be down sampled. This is however not a problem in the indoor environment where the scans where taken, as the furthest points from a scan are far below the maximum range of the scanner.

The software can also perform registration of the scans taken from different positions, giving accurate information about how all scans are connected in the form of transformation matrices for each scan. The registration process allows the point clouds from all scans to be described in the same coordinate system, creating an accurate map of the environment. In this work, the information already produced by the RiSCAN PRO software is used as ground truth data. This
information is used to simulate the movement of the sensor carrier, and it is also useful when evaluating the developed algorithms.

1.5.2 Data Set: The Fire Facility

The primary data set used in this work was collected by FOI at a firefighter training facility in Linköping. The facility consists of a complex of cargo containers with the inside modified to mimic a residence. There were four $12 \times 2$ m containers used in the data collection, composing most of the ground floor of the facility. The layout of the facility is illustrated in Figure 1.3, including the scan locations. There are 23 scans spread across the facility and it is important to note that many consecutive scanning locations are separated by several meters. A big difference in viewpoint might make it hard for the algorithms to associate features between scans, which has to be taken into consideration. From the scans, two down sampled point cloud data sets of the fire facility are produced by the Riegl scanner software. The point clouds in the first set have a resolution of $5 \text{ cm}^3$ between all neighbouring points, while the second is more dense with a resolution of $5 \text{ mm}^3$. In addition to the point clouds of both data sets, the scanners position and orientation of each scan, in relation to the first scan, are also available. This is later used in the creation of coordinate systems.

The visual scene inside the fire facility is that of a residence which has suffered
1.5 Available Resources

![Figure 1.3: A layout of the bottom floor of the facility, including the interior. Scanning locations are marked by encircled crossmarks.](image)

Figure 1.4 shows images taken by the VZ-400 during a scan in the fire facility. Dark soot from the fire covers more or less every surface in the complex. The scans were also taken in poor lighting conditions, which together with the soot creates a homogeneous dark texture of the environment. Because of this, the supplementary intensity produced by the scanner is of lesser use than in other situations. This also means that visual SLAM methods become unreliable in such an environment, because these rely on finding unique visual features from the texture.

The interior consists of furnitures, manikins and miscellaneous household equipment. Noticeable furnitures are sofas, tables and different kinds of chairs. There are also some objects that might seem misplaced from a normal residence such as various pipes and lattices as well as a few distinct fireplaces from which the
fires were created. The placement of the objects is mixed, where certain objects are sometimes placed solitarily and at other times placed on top of each other or in groups. Examples of the latter are dummies sitting on a chair or another chair on top of a sofa. Closely grouped objects create a cluttered environment and which makes it considerably harder for an ATR algorithm to distinguish between objects.

A distinct difference from an indoor environment in an actual house or apartment complex are that most of the walls in the fire facility are made of corrugated steel, as is common for cargo containers. Unlike a regular wall, these walls are not flat, but are instead wave shaped, which might negatively affect the performance of the feature extraction. The walls’ properties manifest themselves as straight vertical edges in the point cloud depending on the perspective.

1.6 Outline of Thesis

Following this introduction, Chapter 2 explains some basic concepts and provides a theoretical basis to the selected approach. Chapter 3 gives a detailed description of the steps performed during the implementation of this work, including the development of the algorithms and the tests used for evaluation. After that, the results of all tests are shown in Chapter 4. A discussion of the result and the used methods follows in Chapter 5. Finally, conclusions of the work are given in Chapter 6.
This chapter explains basic concepts explored in this thesis, such as SLAM and ATR. The purpose of this chapter is to provide a theoretical basis for the methodology presented in Chapter 3.

### 2.1 Simultaneous localisation and mapping

Simultaneous localisation and mapping (SLAM) [9, 32] concerns the problem of localising a sensor carrier while it is moving through an unknown environment and at the same time producing a map of it. The problem is derived from two sub-parts. One is *mapping*, the problem of building a model of the environment from known sensor positions. The other is *localisation*, the problem of using a model to determine the position of the sensor carrier. In SLAM, these things are done simultaneously, causing the mapping to be based on estimated positions, which in turn are based on the mapping. These dependencies make the simultaneous localisation and mapping a difficult problem to solve. Figure 2.1 shows an illustration of the SLAM problem.

#### 2.1.1 Model

The SLAM problem is solved with the help of models that describe how the world should behave and sensors that provide new information of the actual world. The SLAM model can be described by

\[
p_{k+1} = f(p_k, u_k) + v_k, \quad (2.1a)
\]

\[
M_{k+1} = M_k, \quad (2.1b)
\]

\[
y_k = h(p_k, M_k, u_k) + e_k, \quad (2.1c)
\]
Figure 2.1: The SLAM problem. A sensor carrier (triangle) moves in an unknown environment with the help of landmarks (stars) that are measured by the sensors.

where \( f(\cdot) \) and \( h(\cdot) \) are functions. The model contains a motion model that describes how the sensor pose \( p_k \) changes between time instances \( k \), i.e. the movement of the sensor carrier. Sensors can give information of this movement and can be incorporated through an inertial input \( u_k \). Examples of such sensors include inertial measurement units (IMU) or odometers for wheeled vehicles. The map \( M_k \) contains the position of all landmarks, which are assumed to be static.

The measurement model describes how the measurements \( y_k \) relate to the map and the sensor pose, the current state of the system. The measurements can come from many different kinds of sensors. The sensor used in this thesis is a laser scanner that provides observations of the surroundings in the form of 3D points, some of which are stored as landmarks in the map. As the sensor carrier moves through the environment, the algorithm uses these models and compares new observations with the map in order to estimate the current position and vice versa.

Because every sensor is affected by noise, the true position can never be known for certain. Likewise, there is no model that can describe the world perfectly and the models used in SLAM must account for these errors. Therefore, the SLAM model contains the system error \( v_k \) and the measurement noise \( e_k \). These are modelled with uncertainties in the form of additive Gaussian noise in all estimated variables and the correlations between them represent their dependency on each other.

2.1.2 Algorithm

The accuracy of the sensors and how well the world is modelled obviously affect the performance, but it is also affected by the choice of SLAM algorithm. There are a few general issues that a SLAM algorithm need to handle.
• The issue of consistency concerns how well the resulting SLAM model reflects reality. Inconsistency causes inaccuracies in the map and false sensor position. If not corrected, inconsistency eventually leads to an useless map and failed localisation.

• A related issue is that of robust loop closure. Loop closure means that the SLAM algorithm drastically reduces its uncertainty when visiting the same position multiple times, which is a necessity to avoid inconsistencies caused by error accumulation over time. A robust SLAM algorithm should also avoid false loop closures that usually result in major inconsistencies. Situations with low SNR often increase the probability of incorrect associations, which in turn cause false loop closures.

• A third problem is how well the algorithm scales in computational complexity with an increasing number of landmarks. A high scaling in complexity might increase the computation time above an acceptable limit when the map grows large.

There are many different approaches to SLAM. A common type of these are based on nonlinear filtering [11], where the state is augmented with the map according to

\[ X_k = \begin{pmatrix} p_k \\ M_k \end{pmatrix}. \]  

(2.2)

These methods fall into the extended Kalman filter (EKF) approach, called EKF-SLAM [3, 8], or the particle filter (PF) approach, called FastSLAM [16]. In this thesis, the EKF-SLAM is selected as the most appropriate method to be used together with ATR. The reason behind this decision is that EKF-SLAM is straightforward to implement, its advantages fits the situation and its disadvantages can be mitigated by the ATR algorithm. One of its advantages is that, unlike the PF, EKF-SLAM can make full use of loop closure because the correlation of all landmarks and state variables are kept in a joint covariance matrix,

\[ P = \begin{pmatrix} p_{pp} & p_{pM} \\ p_{Mp} & p_{MM} \end{pmatrix}, \]  

(2.3)

where the diagonal of it represents the variance of each element and the rest shows the correlation between them. However, this is also why it is fragile against false associations between observations and landmarks. EKF-SLAM scales quadratically in computational complexity with the number of landmarks, which might become a problem when the map grows large.

**EKF-SLAM**

The EKF-SLAM algorithm is simply the application of the EKF to the SLAM model (2.1) with the state vector \( X \) in (2.2). The EKF-SLAM can handle non-linear motion and measurement models by linearising the models around the current estimated state. Once the models are linear, the standard Kalman filter is performed with its two update steps, the *time update* and the *measurement update*. These two steps make the core of the EKF-SLAM algorithm. It is important to
note that because the linearisation is an approximation, the EKF cannot guarantee convergence nor an optimal solution.

The linearised estimation is based on Jacobians from the nonlinear models, which are derived through Taylor expansion. These linear models are usable by the Kalman filter. The Jacobians of the model functions \( f(p_k, M_k, u_k) \) and \( h(p_k, M_k) \) from (2.1) are given by

\[
F = \frac{\partial f}{\partial X} \bigg|_{X=X_{k-1|k-1}}, \quad (2.4a)
\]

\[
H = \frac{\partial h}{\partial X} \bigg|_{X=X_{k|k-1}}, \quad (2.4b)
\]

where \( X_k \) from (2.2) is the current estimated state of the system.

The time update predicts the current state of the system based on previous estimations and the motion model of the system. The predicted system state is therefore given by

\[
p_{k|k-1} = f(p_{k-1|k-1}, u_{k-1}), \quad (2.5a)
\]

\[
M_{k|k-1} = M_{k-1|k-1}. \quad (2.5b)
\]

To account for the predicted model error \( v_k \) from (2.1), the time update includes an increase in uncertainty of the sensor pose with an addition of process noise \( v_k \sim \mathcal{N}(0, Q) \). Note that the map \( M_k \) is assumed to be stationary, causing no increase in the uncertainty of the landmarks. The predicted covariance matrix is therefore updated according to

\[
P_{k|k-1} = \begin{pmatrix}
FP_{k-1|k-1}^P F^T + Q & FP_{k-1|k-1}^M \\
F^T P_{k-1|k-1}^M & P_{k-1|k-1}^M
\end{pmatrix}. \quad (2.6)
\]

The measurement update corrects the prediction by using the measurements \( y_k \) associated with landmarks in the map \( M_k \) according to the measurement model in (2.1). The updated system state is calculated according to

\[
X_{k|k} = X_{k|k-1} + K\epsilon, \quad (2.7)
\]

where the innovation

\[
\epsilon = y_k - h(p_{k|k-1}, M_{k|k-1}) \quad (2.8)
\]

describes the difference, or error, between the model and observations. The Kalman gain \( K \), decides the influence of the innovation and is defined as

\[
K = P_{k|k-1} H^T S^{-1}, \quad (2.9)
\]

where

\[
S = HP_{k|k-1} H^T + R \quad (2.10)
\]

is the covariance of the innovation. \( R \) is the modelled variance of the measurement noise \( e_k \sim \mathcal{N}(0, R) \) and accounts for the uncertainty of the measurements.
The system covariance $P$ is updated by

$$P_{k|k} = P_{k|k-1} - KS K^T,$$

(2.11)

where the new information (innovation) reduces the uncertainty of the system.

In addition to the EKF, the EKF-SLAM algorithm also has a few steps common in most approaches to SLAM. In order to compare new observations with models in the measurement update step, it requires a function that can match each observation to the corresponding landmark in the map. This is done by the data association, explained in more detail in Section 2.4. Feature extraction is a method that filters important information from the sensors, reducing dimensionality of both the map and the search space for the matching in the data association step. The method of feature extraction used in this thesis is explained in Section 2.2. The map augmentation is a step of the SLAM algorithm which transforms observations that does not have matches in the map into new landmarks. This initialisation of new landmarks is based on including the uncertainty of the sensor pose and the observations [27].

### 2.2 Feature Extraction

When processing large amounts of 3D data, such as point clouds produced by laser scanners, dimensionality becomes a problem. In such cases, there is too much superfluous information for most applications. An average scan used in this thesis contains hundreds of thousands of points. It is ineffective and time consuming to compare and analyse every point in a scan, when only a small subset of these points are enough. Another problem occurs when comparing point cloud data from two different measurements. SLAM relies on finding corresponding points in the two data sets in order to estimate how the sensor has moved. Because of uncertainties in the positions of both the sensor and 3D points, it becomes hard to identify the same point twice using only its individual 3D position.

The use of point features can solve both of these problems by describing sections of data in a compact representation. This is done by extracting a small number of points from the data that are uniquely identifiable in other ways than their coordinates. For example, a point feature can describe a large number of 3D points in a volume around a single point by attaching a description of the data around it. All surrounding points are replaced by this description, reducing dimensionality. The uniqueness of the selected point, or rather the area around it, determines the likelihood of it being re-identified between multiple scans. Some points are more unique that others, e.g. a corner of a room is more easily identifiable than a point in the middle of a flat wall. There are many variations of features and feature extraction methods and the choice depends on purpose and what kind of sensor is used.

Scale-invariant feature transform (SIFT), developed by David G. Lowe [15], is a well known feature extraction method that is commonly used with visual sensors. It uses differences in texture to extract interest points that are re-identifiable,
even from different viewpoints. Although the data set contains RGB-values for each 3D point, the measurements are taken under poor light conditions with most of the environment covered in soot, as explained in Section 1.5.2. An appropriate method for the data set used in this thesis would rather make use of the range information produced by the laser scanner.

2.2.1 Normal Aligned Radial Feature

Developed by Steder et al. [30], normal aligned radial feature (NARF) is a method that selects rotation invariant point features from 3D-data. The selection is solely based on range data, not taking visual properties of point clouds like intensity into account. This is not a problem in our case because of the homogeneity of the texture in the fire facility data set. However, by producing a range image, or depth image, of the 3D-data, NARF uses ideas from computer vision. The features are developed to be recognised independently of the viewing angle, using data from single range scans. These are the reasons why the NARF is suitable for this work. In addition to that, NARF has also been used earlier in similar work for place recognition using SLAM [29]. Finally, all required functions and classes to NARF are under an open source license and available in Point Cloud Library, PCL, [1], which should save time in the implementation.

NARF selects 3D points, called interest points, in positions where the surface is stable enough to ensure robust estimations of normals, but also where there is enough change in its vicinity to make it somewhat unique. Because the algorithm was developed to robustly handle partial views, it explicitly extracts borders from the range image to represent the shapes of objects seen from a certain perspective. The knowledge of where borders are located is used in the selection of interest points. Descriptors are estimated by looking at a small radial patch of a range image centred at the location of each interest points. These contain information of how much change there is in an area around the point in order to describe it. This description of appearance rather than geometric data makes it possible to re-identify the same points regardless of the uncertainty of the sensor position. The extraction of NARF points consists of a few steps that are summarised below. These are run sequentially as each step is dependent on the previous result.

Range Image Creation

NARF makes use of a range image in the extraction of borders, interest points and descriptors. A range image is a 2D representation of a 3D space, where every image point receives a corresponding 2D coordinate and range value. The range value describes a points distance from the sensor. In this work, the measurements are spherical point clouds with the laser scanner in the origin, so the range image is obtained through a spherical projection. Every image point’s 2D coordinate is given by an azimuth and altitude angle. The pixels in the resulting range image are placed according to the image points and the resolution is determined by an angular resolution (degrees per pixel). The point cloud can rarely cover every pixel in an image and pixels not representing a point from the point cloud are set to unknown or optionally to the maximum range.
Note that the spherical projection causes shapes far away from the centre to become distorted when viewed in 2D, as illustrated in Figure 2.2. If the estimation of interest points where done using only the range images, the distortions would affect the calculations depending on the viewpoint. This is however handled by the interest point extraction by using both the range image and 3D point cloud data.

**Figure 2.2:** Range image created from a point cloud scan taken in the fire facility. Close objects are shown in violet and far away objects in yellow.

**Border Extraction**

The NARF interest point extraction finds borders in the range image. There are many reasons why it is important to clearly point out the borders in the range image. These borders represent the shape of the objects seen from a certain perspective. There is also an advantage to separate between different kinds of borders. The border extraction method traverses from foreground to background in the range image, separating object borders, shadow borders and veil points. The object borders are the outmost visible points of an object and shadow borders are points that are next to what is occluded. Veil points are phenomena typically found in 3D data from lidar sensors and should not be taken into account during interest point extraction. These are therefore distinguished from the others border points. An image of extracted border points from a scan taken in the fire facility data set can be seen in Figure 2.3a.
The border extraction is performed on every image point by first creating a heuristic to find the typical 3D distance to neighbouring points. This information is then used to calculate a score based on the likelihood that the point is part of a border. The type of border point (object border, shadow border or veil point) is also identified. Finally, points with a score under a certain threshold are removed.

![Image](image.png)

Figure 2.3: (a) The extracted border points in a section of a point cloud. The green points represent object border, red: shadow borders and teal: veil points

(b) The extracted interest points using a support size of 0.6 m.

Interest Point Extraction

The NARF interest point extraction uses the range image and border points to sort out points that has a higher chance to have an unique NARF descriptor. A flat surface is stable for estimation of normals, but is not by itself a good place for unique points. Therefore, the extraction process first looks at a local neighbourhood of every image point and sets a score based on the level of surface change. It also determines a dominant direction of the surface change. The size of this neighbourhood is determined by a support size parameter. This is also used later to describe the size of the range value patch in the extraction of NARF descriptors.

Note that the calculations are not performed solely in the range image space, but around every pixel’s corresponding point in the 3D point cloud. Local normals from each point directed toward the sensor are used to switch between the two representations. These normals are calculated using principal component analysis (PCA) on a local neighbourhood around the point.

After the first step, the method looks at the dominant direction of other points in the area, and sets an interest score based on how much these directions differ from each other and how much the surface in the point itself changes. This is what makes the extraction select points that are not in the middle of a flat wall nor at the edges, but rather points on surfaces close to edges. It also performs smoothing of the interest values and a non-maximum suppression to sort out the final interest points, i.e. only interest-values above a certain threshold are kept.
The results of the extraction can be seen in Figure 2.3b.

**The NARF descriptor**

The information in an area around an interest point is defined by a descriptor. The goal of the NARF descriptor is to be able to recognise the same points in a 3D-scene, independent of perspective. In the previous step, the interest point extraction reduced the computational space by bringing forward the best candidates for this goal. The NARF descriptor describes the surrounding of a points in a compact format by listing the differences in a *normal aligned range value patch*. This is a small range image with the point in the centre, seen from the perspective of the normal.

The calculation of a NARF descriptor starts by producing a normal aligned range value patch. A star pattern is then overlayed on the patch, which can be described as a number of radial beams originating from the centre point. The size of the radial patch and the star is determined by the same support size used for interest point extraction. From each beam, a value that describes the surface change is calculated and stored in the NARF descriptor. Finally, a orientation is extracted from the descriptor, that shifts the values in the descriptor to make it invariant to the rotation.

### 2.3 Autonomous Target Detection and Recognition

Autonomous target recognition (ATR) [24] is a general term used for algorithms that can detect and recognise specified physical objects using data from sensors. The objective of the ATR algorithm is to have as many correct target detections (true positives) and as few incorrect ones (false positives) as possible. In this work, ATR is used to expand the SLAM map with explicit objects, much like an advanced form of feature extraction. The method uses 3D data to extract and recognise targets. **Targets** are small sections of the point cloud that correspond to the surface of specific objects such as furnitures and people. Once a target is identified as a certain object, all those points can be replaced by an object model. These models can be described by a single 3D point, but unlike point features, objects have additional attributes attached to them that describe their physical size, shape and orientation in the map.

As explained in Section 1.4, there are a variety of approaches to ATR. The ATR used in this thesis is primarily based on a method using rectangle estimation by Grönwall et al. [10]. This method was developed for the recognition of military ground vehicles in outdoor environments. Those targets are somewhat different from the furniture targets found in the indoor environment of the fire facility, but there are also similarities. Much like a tank, an armchair has a recognisable size and shape and it can be assumed to be on the floor/ground. The ground in this case can also be assumed to be flat and horizontal, because it is indoors. However, this also means that the target has to be separated from the walls and the other parts of the interior.
The ATR process can be divided into two steps, target extraction and model matching. The model matching process requires that the 3D points of the target clouds represent the surface of the individual objects. The targets therefore need to be separated from the rest of the scene in each point cloud scan before entering the model matching. This process is called target extraction.

The target extraction process starts with a removal of coarse scene features with large flat surfaces, such as floor, roof, walls, and doors. The candidate for this process is a form of plane extraction based on a RANSAC algorithm derived from [31]. When these planes are removed from the data set, the remaining interior of the point cloud scan is clustered into smaller potential targets using $k$-means clustering [28].

The model matching is based on rectangle estimation, which provides an estimated size and orientation of the target in two dimensions. This is then matched and evaluated with specified object models. The rectangle estimation is described as an optimisation problem

$$\min (c_3 - c_1)(c_4 - c_2)$$

subject to:

$$n_1 x_i + n_2 y_i - c_1 \geq 0, \ i = 1, \ldots, N$$

$$-n_2 x_i + n_1 y_i - c_2 \geq 0, \ i = 1, \ldots, N$$

$$n_1 x_i + n_2 y_i - c_3 \leq 0, \ i = 1, \ldots, N$$

$$-n_2 x_i + n_1 y_i - c_4 \leq 0, \ i = 1, \ldots, N$$

$$n^T n = 1$$

where each $n_1 x + n_2 y - c = 0$ describes a line in two dimensions with a slope $n = (n_1, n_2)^T$ and distance $c$ from the origin. The points $(x_i, y_i)$, $i = 1, \ldots, N$ are either on or inside the four lines that make the minimised rectangle. The size and orientation of the target from a perspective (top or side view) can then be determined from the rectangle’s size and its orientation relative to the axis of the sensor’s local coordinate system.

The quality of the match is evaluated by the relative mean square error (RE) [6]. Name the distance between each point in the target point cloud $l = (x, y, z)^T_i$ and its projection on the closest surface of the object model $l’ = (x’, y’, z’)^T_i$. The RE is then calculated according to

$$RE = \frac{H(l, l’)}{S(l)},$$

where $H$ is the Hausdorff distance [18]

$$H(l) = \frac{1}{2N} \sum_{i=1}^{N} ||l_i - l_i'||^2_2$$
2.4 Data Association

The ability to know when a point in the environment is seen for the second time is essential for solving the SLAM problem. Data association [2] compares new observations with landmarks in the map and decides which points correspond to each other. Observed points with no matches in the map are initiated as new landmarks in the map augmentation step, while matched points are fused with the map through the EKF. The latter become the measurements $y_k$ of the measurement model. In fact, the data association is necessary for the measurement update to know which parts of the map should be compared to the new observations. The problem is simply to find out which observations and landmarks describe the same part of the environment.

There are a few approaches to the association problem. In this thesis, a mix of approaches will be used depending on which features are being used. Because the features used in this thesis, NARF points and objects, have different qualities that make them identifiable based on appearance rather than geometrics, some of the complexity can be removed from the data association without losing robustness. Therefore, the selected methods for data association are not as advanced as might otherwise be appropriate. The selected methods are based on individual compatibility between data points based on nearest neighbour search and Mahalanobis distances.

2.4.1 Nearest Neighbour

A simple way to associate a number of observed 3D points $y_i$ and stored landmarks $m_j$ is to match those with the shortest Euclidean distance $d_{ij}$ to each other,

$$d_{ij} = ||y_i - m_j||,$$

(2.18)

where $i \in [1, \ldots, N^y]$ and $j \in [1, \ldots, N^m]$ are the number of measured points and landmarks in the map. This can be implemented through a nearest neighbour search for each newly observed 3D point $y_i$. A threshold can be tuned to remove matched points that are too far away from each other. However, the problem with

and

$$S(l) = \frac{1}{N} \sum_{i=1}^{N} ||l_i - \mu||^2$$

(2.16)

is the spread of the data based on the estimated mean value $\mu = (\mu_x, \mu_y, \mu_z)$.

The rectangle estimation does not always give a perfect fit with the object model and can be optionally improved with least-square fitting [4]. This minimises the distance between the target points $l_i$ and their projections $l'_i$ according to

$$\min \sum_{i=1}^{N} ||l_i - (l'_i r + t)||.$$  

(2.17)
matching using only the Euclidean distance is that the nearest point might not be the correct one. This method does not take into account any uncertainties of the landmarks or the sensor position.

The NARF descriptor on the other hand is not dependent on any uncertainties in the SLAM model and is suitable for nearest neighbour search based on the squared NARF descriptor distance. A matching method called Fast Library for Approximate Nearest Neighbour (FLANN) [17] is available in PCL. This method uses approximate nearest neighbour search, which is a lot faster than exact nearest neighbour search, at a minor loss in precision. What makes it unique is that it applies priority search on hierarchical $k$-means trees ($k$D-trees split by $k$-means clustering). Because each NARF descriptor used in this thesis is a high dimensional vector (36 descriptor values) and the high number of points in the map and each measurement, the use of a fast matching algorithm is suitable to speed up the association.

### 2.4.2 Mahalanobis Distance

The drawbacks of using Euclidian distance for association can be overcome by using the normalised innovation squared (NIS), also known as the Mahalanobis distance [19]. It squares the innovation $i_{ij}$ calculated from the difference between measurement $y_i$ and landmark $m_j$ and normalises it with its uncertainty $S$. The NIS can then be defined as

\[
M_{ij} = i_{ij}^T S^{-1} i_{ij},
\]

\[
i_{ij} = y_i - m_j,
\]

\[
S = HPH^T + R_y,
\]

where $M_{ij}$ takes into account the uncertainty of the sensor pose and landmark through $H$ and $P$ and the uncertainty of the measurement through the modelled variance of the measurement noise $R_y$. Because the expected errors in EKF-SLAM are Gaussian, the NIS forms a $\chi^2$-distribution with the same degree of freedom as the dimension of the innovation, $M_{ij} = \chi^2_{\text{dim}(i)}$. The integral of the $\chi^2$-distribution tells the probability that $y_i$ and $m_j$ will be matched. Association is made by gating the pairs through an arbitrary threshold $M_{ij} \leq g$. Values below the threshold are accepted and values above are discarded. Low values of $g$ indicate a harsh gate, where only very certain associations pass. Higher values indicate a more lenient gate, as more matches are accepted.
This chapter describes the selected methods and covers all the necessary steps of their implementation. The evaluation process and the development of the tests are also described.

### 3.1 Overview

The main objective of this thesis is to implement an algorithm that combines SLAM and ATR and investigate the performance. The following steps summarise the necessary steps to accomplish this task.

1. A preliminary study of the problem, available resources, as well as a review of the related literature.
2. Creation of proper measurements and input. (Pre-processing)
3. Implementation of the EKF-SLAM algorithm, including feature extraction and data association.
4. Implementation of the ATR algorithm.
5. Implementation of the Combined SLAM and ATR algorithm, named ATR-SLAM. This includes the introduction of objects to the SLAM model, a new data association and the reduction of landmarks.
6. Testing and evaluation of the algorithms, including an analysis of the ATR algorithm’s impact.

The first step is necessary in order to investigate what has already been done in the field and what options are available. This is presented in previous chapters.
The second step makes it possible to create scenarios that mimic the movement of the sensor carrier. The implementation is a process of gradual expansion in the order of step 3–5. The different parts, or subsystems, that make up SLAM are separate algorithms that can perform their task individually. Therefore, the algorithms are first developed separately and then sequentially combined. The final step evaluates the performance of the developed algorithms by creating scenarios that address the issues defined in Section 1.2. Comparisons between the two algorithms are made.

The EKF-SLAM algorithm composes the backbone of the developed algorithms. The SLAM loop will decide when different algorithms are to be run and all other algorithms, the EKF, feature extraction, ATR, data association and map augmentation are therefore sub-algorithms of SLAM. In order to investigate the issues presented in the problem formulation two different versions of SLAM are implemented. One is the EKF-SLAM, which is developed around the usage of only NARF points as features. The second is the algorithm that combines SLAM and ATR, dubbed ATR-SLAM, which in addition to NARF points also uses objects produced by the ATR as features. The ATR-SLAM is simply an extended version of the regular EKF-SLAM, modified for the use of objects. Figure 3.1 shows an overview flowchart, made to illustrate the relationship.

From a SLAM perspective, the ATR is simply another type of feature extraction, using objects. If objects are present frequently enough, then it would be possible to run SLAM using only ATR. However these kinds of objects are not always available. It is therefore necessary to include another feature type, making it possible to navigate through rooms with no identifiable objects present.

**Figure 3.1:** A high-level overview flowchart of the ATR-SLAM algorithm. If the ATR is removed, the regular EKF-SLAM remains. The ATR-SLAM is an expansion of the EKF-SLAM.
3.2 Software

The environment in which the developed algorithms are implemented are selected based on the available resources. All programs in this thesis are implemented in either Matlab or C++. Matlab is selected as the primary development environment due to several reasons. Previous work implemented in Matlab at FOI in the area of ATR, 3D modelling and point cloud manipulation are available in an internal library. This includes the methods used in the model matching and extraction of interior. K-means clustering, used in the target extraction, is also available in Matlab through Mathwork's statistical toolbox.

The Point Cloud Library (PCL) [1] is primarily used for the extraction of NARF points. It would not be possible to develop a sufficient feature extraction method without PCL. The library is also used for the association of NARF descriptors, where the PCL’s kd-tree and FLANN methods are used. The feature extraction and data association using PCL are written in C++ as executable programs.

The main loop of the program, the SLAM algorithm, runs in Matlab. The executable feature extraction and association programs are run through a system call in Matlab. The Matlab program waits for the executables to finish before continuing. One way communication from Matlab to the executables is performed by simple line commands. Two way communication is done through reading and writing of files. This is appropriate because the only information needed to be passed around are large matrices of 3D position and descriptor values.

3.3 Coordinate Systems

There are two types of coordinate systems, or frames of reference, to keep track of. The global coordinate system, or world frame, is the coordinate system that is used to unify measurements from multiple scans. This system has its origin in the first scan position. The local coordinate system, or sensor frame, is a coordinate system that has its origin in the current sensor position, which is oriented relative to the global as illustrated in Figure 3.2.

The coordinate systems are connected through linear transformation. The transformation of a 3D position expressed in the sensor frame $l^S$ to a position in the world frame $l^W$ can be described by

$$l^W = r^{W/S} l^S + t^{W/S},$$

where $r^{W/S}$ is a positive rotation $r$ around the coordinate system’s z-axis,

$$r(\phi) = \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

(3.2)
and $t^{W/S}$ is a translation

$$ t = \begin{pmatrix} t^x \\ t^y \\ t^z \end{pmatrix} $$

(3.3)
given by the sensor’s position relative to the origin of the global coordinate system. Given $r = r^{W/S}$ and $t = t^{W/S}$, the reverse transformation of a 3D position expressed in the world frame $l^W$ to a position in the sensor frame $l^S$ is given by

$$ l^S = r(\phi)^T (l^W - t), $$

(3.4)

where the transpose of the same rotation is used to transform the coordinates from a global to a local system. These transformations make it possible to connect all the scans as long as the sensor pose in each scan relative to the first scan position is supplied.

In order to correctly describe the position and orientation of a freely moving sensor in 3D, a pose with six degrees of freedom (6DoF) is generally required, i.e. a 3D position and a roll, pitch and yaw, expressed in Euler angles. In this work, the roll and pitch angles have been assumed to be known and set to zero. The bearing is therefore the only interesting orientation. This approximation is possible to make, because the laser scanner stays horizontal in all scan positions. However, the scanner changes height in a few scans, which means that the coordinate systems must account for the translation along the $z$-axis. The final system has
four degrees of freedom, with the sensor pose

\[ p = \begin{pmatrix} p^x \\ p^y \\ p^z \\ p^\phi \end{pmatrix}, \]

where \((xyz)\) represent the 3D-position and \(\phi\) is the yaw angle from the \(x\)-axis. This system is similar to that of SLAM in the horizontal plane, but with the addition height. This is a simplification to the full 6D SLAM as there are fewer states and less complex transformations between coordinate systems.

### 3.4 Pre-Processing

The data sets of the fire facility, Section 1.5.2, are collected by FOI for multiple purposes other than the focus of this thesis. The size and weight of the laser scanner used, Riegl VZ-400, and the scanning process would be too cumbersome for a first responder moving fast through the facility. As explained in Section 1.5.1, the collected point clouds are already pre-processed to a desired resolution and the measurement noise is therefore negligible. No inertial input from sensors are available, but the position and orientation of the laser scanner in each scanning location will instead be used to simulate such an input. It is therefore necessary to pre-process the available data in order to create scenarios where the SLAM algorithms can be tested properly.

#### 3.4.1 Point Cloud Measurements

The purpose of pre-processing the point clouds is to create proper measurements that can be used as input to the feature extraction and ATR target extraction. The point clouds available from the fire facility data set consist of 3D points as well as an RGB-value to each point. Each scan covers a 360 degree field of view and both resolutions are considered noise free. The measurements used in this thesis are distorted versions of these point clouds that imitate the scenario of a first responder moving through the facility.

The process starts by cutting the point clouds from the data set into an appropriate field of view of 120° in order to represent a scenario where a less all-encompassing sensor is used. The cut-outs also allows the same scan locations to be used twice (or even three times), while still not using any measurements twice. This becomes useful in the creation of test scenarios, as the sensor carrier can move in one direction and then back again. Overlapping of the field of view between scans is considered in the selection. Figure 3.3 shows the before and after result of this process.

The intensity is of no use in either the NARF feature extraction or the ATR and is therefore discarded. White Gaussian noise \(e^y\) is then added to every point in all scans in order to represent a less accurate laser scanner. This is the noise later modelled in the SLAM measurement model with a standard deviation cor-
responding to the measurement noise. Each sample point $s$ in the pre-processed point cloud can then be described as

$$s = \begin{pmatrix} s^x \\ s^y \\ s^z \end{pmatrix} + e^y,$$

(3.6)

where the noise $e^y$ has the same dimension as $s$.

### 3.4.2 Inertial Input

The inertial input is created by using the positions and orientation of the scanner. The purpose is to generate data that can describe the movement of the sensor carrier throughout the fire facility. This is necessary from a SLAM perspective, because simple motion models such as constant velocity or acceleration models are insufficient in describing the movement of the first responder. The creation of the inertial input is a simple way to provide this information for each new scan.
position. It describes how the sensor pose changes between scans and is therefore created using the difference between each true sensor pose \( p_k \),

\[
u_k = (p_{k+1} - p_k) + e_k^u.
\] (3.7)

White Gaussian noise \( e_k^u \) is added to the three dimensional position and the yaw-angle. This noise corresponds to the model error of the SLAM motion model.

## 3.5 EKF-SLAM

The regular EKF-SLAM is implemented according to the methods described in Chapter 2. The algorithm solves the SLAM problem by continuously predicting and refining an estimated state of the system, i.e. the position and orientation of the sensor carrier as well as the position of the landmarks in the map. It does so by combining new observations with the SLAM model that describes how the world should behave. Figure 3.4 shows an flowchart of the steps of the algorithm.

The EKF-SLAM algorithm starts by predicting where the sensor should be positioned in the time update, based on previous information. New observations are produced by the feature extraction from the point clouds, which is based on NARF in the regular EKF-SLAM. The data association determines which of these observations correspond to previously seen landmarks in the map and which are to be added as new landmarks by the map augmentation. Those that have a correspondence are used to correct the prediction in the measurement update.

### 3.5.1 SLAM Model

As described in Section 2.1, it is necessary to have a few underlying models of the system in order for the SLAM algorithm to work. These models describes how the state of the system changes over time and the relationship between the sensor measurements and the landmarks in the map.

**System State**

EKF-SLAM is a nonlinear filtering approach, where the system state \( X \) is implemented as

\[
X_k = \begin{pmatrix} p_k \\ m_k^1 \\ \vdots \\ m_k^{N_m} \end{pmatrix}.
\] (3.8)

The system state consists of the target pose \( p_k \) and the position of every landmark \( m_j^k, j \in [1, \ldots, N_m] \), in the map \( M_k \), where \( N_m \) denotes the number of landmarks in the map. The pose \( p_k \) of the sensor carrier is implemented as in (3.5).

Landmarks \( m_k \), the NARF points selected by the feature extraction, are a small subset of the point cloud stored in the global coordinate system. Each landmark
Figure 3.4: The flowchart of the EKF-SLAM algorithm.

contains a 3D position as described by

$$m = \begin{pmatrix} m^x \\ m^y \\ m^z \end{pmatrix}. \quad (3.9)$$

The uncertainty and correlation of the state variables is the same as the covariance matrix (2.3).

**Motion Model**

The motion model, or system dynamics, describes how the system state changes at each time instance. This is implemented according to

$$p_{k+1} = f(p_k, u_k) + v_k = p_k + u_k + v_k, \quad (3.10a)$$

$$M_{k+1} = M_k. \quad (3.10b)$$

This is a motion model where the position $p_k$ of the sensor carrier does not change until new information from an inertial input $u_k$ arrives. This input is used to describe the movement between time instance $k$ to $k + 1$. The system error $v_k$ describes the models deviance from reality and is modelled as white Gaussian noise $v_k \sim \mathcal{N}(0, Q)$. As common in SLAM, the model error is implemented through its covariance $Q$.

The map $M_k$ is assumed to be time invariant, i.e. the world around the target
(walls, furniture etc.) does not move. This assumption becomes important when
the algorithm tries to recognise previously seen features in the environment with
loop closure.

The use of an inertial input is a design choice and could very well have been
implemented as measurements from a sensor and thus modelled through the
measurement model. Also, a motion model without inertial input could instead
be used to represent the complicated movement of the sensor carrier between
scans. However, this would require more knowledge about the sensor carrier’s
movement than is available and (3.10) is sufficient for the intended purpose.

Measurement Model

The measurement model describes the relationship between the measurements
from the 3D point cloud,

\[
Y^m_k = \begin{pmatrix}
  y^m_{k,1} \\
  \vdots \\
  y^m_{k,N_{ym}}
\end{pmatrix},
\]

and the model of the environment, given by the system state \(X\) in (3.8). \(N_{ym}\)
denotes the number of points in each observation. The connection between each
newly observed 3D-point \(y^{m,i}_k\) and map landmark \(m^j_k\) is given by

\[
y^{m,i}_k = h(p_k, m^j_k) + e^m_k,
\]

where \(j\) is the index to the corresponding landmark in the map and

\[
h(p, m) = r(p^\phi)^T \left( \begin{pmatrix} m^x \\ m^y \\ m^z \\ p^x \\ p^y \\ p^z \end{pmatrix} - \begin{pmatrix} p^x \\ p^y \\ p^z \end{pmatrix} \right).
\]

The implemented measurement model is simply a transformation of points in the
map’s global coordinate system to the sensor’s local coordinate system, which is
done using the sensor pose \(p_k\). All models are imperfect and the difference, or
innovation, between what is expected from the model and the new observations
is modelled with white Gaussian noise \(e \sim \mathcal{N}(0, R)\). As previously done with the
model error, the measurement error is implemented through its variance \(R\).

3.5.2 EKF

The EKF is implemented using the equations described in Section 2.1.2. The linearised
estimation is calculated with the help of a Taylor expansion and the EKF
uses the time update to make predictions in every new time instance and the measurement update to correct the predictions when new observations are available.

The time update is implemented according to (2.5) and (2.6). As seen in Section
3.5.1, the motion model (3.10) is implemented as a linear model and does not
need to be linearised.

The measurement update is implemented according to (2.7) through (2.11). The
measurement model (3.12) is nonlinear and requires linearisation. With \( h \) in (3.13), the Jacobian \( H \) from (2.4) is calculated according to

\[
H = \begin{pmatrix}
H^p & H^M
\end{pmatrix}
= \begin{pmatrix}
H^p & H^{m_1} & \ldots & H^{m_N}
\end{pmatrix},
\]

(3.14)

where

\[
H^p = \begin{pmatrix}
\frac{\partial h}{\partial p^x} & \frac{\partial h}{\partial p^y} & \frac{\partial h}{\partial p^z} & \frac{\partial h}{\partial p^\phi}
\end{pmatrix}.
\]

(3.15)

The first three partial derivatives form

\[
\begin{pmatrix}
\frac{\partial h}{\partial p^x} & \frac{\partial h}{\partial p^y} & \frac{\partial h}{\partial p^z}
\end{pmatrix} = -r(p^\phi)^T,
\]

(3.16)

where \( r \) is the transformation from (3.2). The last derivative is calculated according to

\[
\frac{\partial h}{\partial p^\phi} = \begin{pmatrix}
-sin(p^\phi) & cos(p^\phi) & 0 \\
-cos(p^\phi) & -sin(p^\phi) & 0 \\
0 & 0 & 0
\end{pmatrix} \begin{pmatrix}
m^x \\
m^y \\
m^z
\end{pmatrix} - \begin{pmatrix}
p^x \\
p^y \\
p^z
\end{pmatrix}.
\]

(3.17)

The derivatives with respect to the landmarks \( m^j \) are identical for each landmark,

\[
H^{m_j} = r(p^\phi)^T, \ j \in [1, \ldots, N_m].
\]

(3.18)

Each set of three rows of the \( H \) matrix describes a relationship between an observed NARF point and the model based on the sensor pose and a single corresponding landmark in the map. All columns that relate to other landmarks are set to zero.

### 3.5.3 NARF Extraction

The feature extraction for the EKF-SLAM is based on NARF as explained in Section 2.2.1. The process uses a point cloud as input data and returns a smaller number of 3D points and their NARF descriptors, which are later used in the EKF-SLAM algorithm. Note that all necessary functions needed for NARF extraction are provided in PCL. Therefore, the implementation of the feature extraction simply consists of properly arranging and adapting the functions to the problem and data set.

Algorithm 1 illustrates the process of the implemented feature extraction. The algorithm starts by creating a range image from the point cloud, which is later used in the rest of the extraction processes. The range image is set to have an equal field of view as the pre-processed point cloud. The border extraction is straightforwardly implemented with the range image as input. The interest point extraction finds the point features that are the most likely to be re-identified with the help of the borders and the range image. The default tuning of the extraction is adapted, depending on the used data set. A sufficient number of extracted points is required in order to acquire features that are re-identifiable by the association algorithm. The extraction of descriptors is performed in every extracted interest point, as described in Section 2.2.1.
Algorithm 1 Pseudo-code of the NARF Extraction

input: pointCloud
output: NARFpoint
for every point cloud measurement do
    rangeImage = RangeImageCreation(pointCloud)
    borders = BorderExtraction(rangeImage)
    interestPoints = InterestPointExtraction(rangeImage, borders)
    NARFpoint = DescriptorExtraction(rangeImage, interestPoints)
end for

3.5.4 Landmark Association

The data association in EKF-SLAM is entirely based on the association of NARF points, which is also encompassed by the ATR-SLAM. One important aspect behind the selection of the NARF is that its descriptor can be used to match individual points without any concern for the error in estimated sensor and landmark positions. The primary association of NARF points is therefore the NARF descriptor association. However, it does not guarantee correct associations and therefore a second gating, Mahalanobis gating, is implemented to remove false associations. The matches produced by the descriptor association are gated using the Mahalanobis distance, as described in Section 2.4. These two steps are made to ensure that the matched points are at least relatively close to each other. The main steps of the data association is shown in Algorithm 2. Concerning the implementation, it is important to note that the data association is partly implemented in PCL and partly implemented in Matlab. This means that the two association approaches have to work sequentially, instead of intertwined.

NARF Descriptor Association

The association matches extracted NARF points from a newly observed point cloud scan with points already stored in the map. It does so by using the NARF point’s descriptor and position together with knowledge of the sensor’s estimated position. The landmarks in the map, consisting of points from previous measurements, are stored in the global coordinate system. These are transformed to sensor frame in order for them to be compared with the observations.

The landmarks are placed in a kD-tree. A search method called Fast Library for Approximate Nearest Neighbour (FLANN), as described in Section 2.4.1, is used on every extracted NARF point from the measurements. The algorithm returns a list of the k-nearest neighbours in the map in ascending order based on the squared descriptor distance. The lower this squared distance metric is, the better the correspondence between the measurement point and the landmark. The result is therefore a list of matching points with the best matches first.

The list of candidate matches returned by the nearest neighbour search are gated using two metrics. The first gate is a rough Euclidean 3D distance threshold between the map and measurement feature point. The NARF descriptors are only
**Algorithm 2** Pseudo-code of the Data Association of NARF points. The association with regards to the NARF descriptor is followed by an gating using the Mahalanobis distance. The NARF points contain both an 3D point and an NARF descriptor.

```plaintext
input: measNARF, mapNARF
output: matchedLandmarks, newLandmarks
if empty(mapNARF) then
    matchedLandmarks = []
    newLandmarks = measNARF
else
    kdTree = kd-tree(mapNARF)
    for every measNARF do
        kNNList = FLANN(measNARF, kdTree)
        for every kNN in kNNList do
            if PassGating(kNN, measNARF) then
                if !alreadyMatched(measNARF) then
                    matchedLandmarks.pushback(kNN, newNARF)
                else
                    currentMatch = FindExistingMatch(matchedLandmarks)
                    if kNN.descDist < currentMatch.descDist then
                        ReplaceMatch()
                        FindNewMatch(currentMatch.measNARF)
                    end if
                end if
            end if
        end for
        if EndOf(kNNList) then
            newLandmarks.pushback(measNARF)
        end if
    end for
end if
matchedLandmarks = MahalanobisGating(matchedLandmarks)
```

concerned with the immediate vicinity of the interest point, so the rough distance threshold prevents the algorithm from associating features that are similar, but distant from each other, e.g. corners in separate rooms. The second gate is a threshold based on the squared descriptor distance that is chosen as a tuning parameter. A harsh threshold close to zero will result in few matches and a higher value will give more possible matches, but those are also less likely to be correct.

Starting with the nearest neighbour in the list, the best candidate match is chosen from the gating and the indices of the matches are stored. Every feature point that does not find a match, is stored as a potential new landmark. If a candidate interest point in the map is already matched with another measurement, the de-
3.5 EKF-SLAM

descriptor distance between the two matches are compared. If the earlier match has a descriptor distance value that is lower, the old match is replaced with the new one. The measurement point of the old match is then recursively associated again with every point in the map. This continues until all NARF points in the measurement has either received a match or is deemed to be a new landmark. Figure 3.5 shows a result of the data association after a sequence of two consecutive scans on the same target.

![Figure 3.5: Associated points from a measurement (blue) and landmarks in the map (red), highlighted in a section of the point cloud that shows a manikin sitting on a chair.](image)

**Mahalanobis Gating**

The quality of the matched pairs produced by the NARF descriptor association are evaluated by gating every match using the Mahalanobis distance. As explained in Section 2.4.2, the innovation of each pair is normalised with its covariance matrix according to (2.19), where

$$H = \nabla h(p, m^j)_{|p=p_{k|k-1}^k, m^j=m_{k|k-1}^j}$$ (3.19)

where $h$ is (3.13) and

$$P = \begin{pmatrix} p_{pp} & p_{pm^j} \\ p_{m^jp} & p_{m^jm^j} \end{pmatrix}. \quad (3.20)$$
The resulting values $M_{ij}$ for each pair of NARF measurement points $y^i$ and landmarks $m^j$ are gated with a threshold tuning parameter. Values that are below the threshold are kept, while those above are discarded.

### 3.5.5 Map Augmentation

The map augmentation handles the initialisation of new landmarks and their correlation to the system state. All measurement points that are not associated with existing landmarks are added to the map as new landmarks. The new state $X^+$ is updated according to

$$X^+ = \begin{pmatrix} p \\ M \\ M_{\text{new}} \end{pmatrix},$$

(3.21)

where $M_{\text{new}}$ consists of the new landmarks. Every non-associated newly observed point $y^{m,i}$ becomes a new landmark $m^i$ through

$$m^i = g(p, y^{m,i}),$$

(3.22)

where

$$g(p, y) = r(p^\Phi) \begin{pmatrix} y^x \\ y^y \\ y^z \end{pmatrix} + \begin{pmatrix} p^x \\ p^y \\ p^z \end{pmatrix},$$

(3.23)

is the transformation from a local to a global coordinate system.

As seen in (3.23), the addition of new landmarks is dependent on the sensor pose and measurements. These have uncertainties associated to them that need to be taken into account when updating the covariance matrix. The initialisation of the new uncertainties are based on the initialisation explained in Section 2.1.2. The function $g$ is nonlinear and in order to initialise the new uncertainty properly, a gradient $G$ of $g$ is calculated in a similar way previously done for $H$ in Section 3.5.2.

$$G = \begin{pmatrix} \frac{\partial g}{\partial p} & \frac{\partial g}{\partial y} \end{pmatrix} = \begin{pmatrix} G^p & G^y \end{pmatrix}.$$ 

(3.24)

The new extended covariance matrix $P^+$ can then be written as

$$P^+ = \begin{pmatrix} p_{pp} & p_{pM} & (G^p p_{pp})^T \\ p_{Mp} & p_{MM} & (G^p p_{pM})^T \\ G^p p_{pp} & G^p p_{pM} & G^p p_{pp} G^T + G^y R G^y T \end{pmatrix}.$$ 

(3.25)
3.6 ATR-SLAM

The implementation of the Combined SLAM and ATR algorithm, named ATR-SLAM, is the final and main method to be implemented. It builds on the previously implemented EKF-SLAM, using the NARF descriptors as landmarks and modified versions of the same EKF, association algorithm and map augmentation. There are also some changes caused by the introduction of ATR in the EKF-SLAM algorithm:

- An expanded SLAM model system state, using both NARF landmarks and objects as a part of the map. This introduces an additional measurement model to the SLAM model.

- ATR algorithms, such as target extraction and model matching. The creation of object models is really a pre-processing step, but is explained in the ATR for clarity.

- The addition of a new measurement update, map augmentation and association algorithm based on objects. The latter has been named object association in order to make a distinction from the data association based on NARF points, which is now called landmark association.

- A function named landmark reduction that replaces landmarks with objects in the map. This is where the physical object model is used to remove NARF points that are within the objects models occupied space.

An overview of the ATR-SLAM is shown in Figure 3.6. Note that in the algorithm, the correction of the predicted state are performed based on newly observed objects before doing any corrections based on NARF landmarks. This is due to the fact that objects are expected to be more robust features, and are more reliable even if the predicted position is significantly off mark. Also, in order to avoid using the same information twice, it is important that the reduction of NARF landmarks is performed before any association of them is performed.

3.6.1 SLAM Model

Objects are added to the SLAM model based on (2.1). This causes the system state to be extended with the representations of the objects. From a SLAM perspective, the objects are implemented similarly to the 3D NARF points, but with the addition of an orientation. The objects and the landmarks are ordered into separate subsections of the map in SLAM model, referred to as the landmark map and the object map.

System State

The state $X_k$ of the system is implemented according to (3.26). In addition to the target pose $p_k$ and the landmark map $M_k$ found in the EKF-SLAM model in Section 3.5.1, the system state of the ATR-SLAM also consists of the object map.
Figure 3.6: The flowchart of the ATR-SLAM algorithm. The processes involving ATR are introduced on the right.
The state of the ATR-SLAM is therefore defined by

$$X_k = \begin{pmatrix} p_k \\ M_k \\ o^1_k \\ \vdots \\ o^N_k \\ O_k \end{pmatrix},$$

(3.26)

where $N$ denotes the number of objects and every object is defined with a position and bearing, i.e.

$$o = \begin{pmatrix} o^x \\ o^y \\ o^z \\ o^\phi \end{pmatrix}.$$  

(3.27)

The uncertainties and correlations between the target pose, landmarks and objects are given by the covariance matrix

$$P = \begin{pmatrix} pp & p^pM & p^pO \\ p^M p & p^{MM} & p^MO \\ p^O p & p^{OM} & p^{OO} \end{pmatrix}.$$  

(3.28)

**System dynamics**

The system dynamics in ATR-SLAM is similar to the EKF-SLAM defined in Section 3.5.1, but has the addition of objects. Like the landmarks, the objects are also assumed to be static. Therefore, for each new time instance $k + 1$, the state dynamics are given by

$$p_{k+1} = f(p_k, u_k) + v_k = p_k + u_k + v_k$$

$$M_{k+1} = M_k$$

$$O_{k+1} = O_k$$

(3.29)

where $u_k$ is implemented as described in Section 3.4.2 and the system error $v_k$ is modelled as white Gaussian noise $v \sim N(0, Q)$. The variance $Q$ is implemented to model the uncertainty of the sensor position.

**Measurement Model**

The ATR-SLAM model has a measurement model $Y^o_k$ for observed objects in addition to the measurement model $Y^m_k$ from the EKF-SLAM that describes the connection between observed NARF points and landmarks in the map. When a new scan is read at time instance $k$, the object measurement model is described
by

\[ Y^o_k = \begin{pmatrix} y^o_1 \\ \vdots \\ y^o_{N^{yo}_k} \end{pmatrix}, \] (3.30)

where \( N^{yo}_k \) denotes the number of objects seen in the scan. Each

\[ y^o_{k,i} = w(p^o_{k,i}) + e^o_k \] (3.31)
describes the connection between a newly observed object \( y^o_{k,i} \) and a corresponding object in the map \( o^j \). The function

\[ w(p, o) = r(p^\phi)^T \left( \begin{pmatrix} o^x \\ o^y \\ o^z \end{pmatrix} - \begin{pmatrix} p^x \\ p^y \\ p^z \end{pmatrix} \right) \] (3.32)
is simply a transformation from the world frame to the sensor frame.

### 3.6.2 EKF

There are few changes made regarding the EKF from the EKF-SLAM to the ATR-SLAM. They concern the introduction of a new measurement model based on observed objects. The EKF of the ATR-SLAM is implemented using the equations described in Section 2.1.2. The implemented time update is unchanged from the regular EKF-SLAM as the objects, like landmarks, are assumed to be stationary.

The measurement update is implemented according to (2.7) through (2.11). The measurement model (3.31) is nonlinear and requires linearisation. With \( w \) in (3.32), the Jacobian \( W \) is calculated in the same way as \( H \) in Section 3.5.2, but with regards to objects instead of landmarks.

### 3.6.3 ATR

The detection and recognition of objects in ATR is implemented as a sequence of functions that when combined takes a point cloud measurement as input argument and returns a representation of the found objects. The used methods are explained in Section 2.3.

The sequence of functions shown in Algorithm 3 can be divided into two steps, target extraction and model matching. The first step reduces the search space by dividing the data into regions of interest where objects are more likely to be present. Point clouds of potential objects called targets are then extracted within the interest regions. In the second step, targets are compared with different object models. Every target is either associated with a single object model or discarded. The recognised objects are represented by a shape and size as well as an 3D position and yaw orientation in the local coordinate system of each scan.
Algorithm 3 The ATR algorithm for extracting objects from the point cloud measurements. The best object model for each target is selected if the quality of the match is sufficient.

**input**: PointCloud, ObjectModels  
**output**: DetectedObjects  

for every new point cloud measurement do  

    Interior = InteriorExtraction(PointCloud)  
    Targets = k-meansClustering(Interior)  

for every Target do  

    TargetPose = RectangleEstimation(Target)  
    for every ObjectModel do  

        MatchResultMatrix = ModelMatchQuality(TargetPose, ObjectModel)  

    end for  

    BestMatch = Min(MatchResultMatrix.CEM)  
    if BestMatch.CEM<ThreshHold then  

        DetectedObject = [BestMatch.ObjectModel.identifier, TargetPose]  

    end if  

end for  

end for  

Target Extraction

The extraction starts by removing all data in the point cloud not likely to be part of an object. This is done manually with the help of FOI’s point cloud functions described in Section 3.2. All major flat surfaces are removed from the point cloud such as walls, roofs and floors, leaving only the interior of the rooms. An example of the resulting data can be seen in Figure 3.7 together with the input data. Ideally, this should be performed automatically by a plane extraction algorithm based on RANSAC, but preliminary tests showed that the algorithm produced insufficient results with the data set used in this thesis. The remaining data is then clustered into small individual clouds of potential objects that become the targets by using the k-means clustering function kmeans in Matlab’s statistics toolbox. The functions is implemented to first randomly divide the data into a number of smaller sections, each with a central point, and then altering these through iteration based on the squared Euclidian distance. A typical result of the clustering function is seen in Figure 3.8. This function is partly a stochastic process, causing the performance to vary between iterations. However, the variation can be minimized by adapting two parameters, the number of clusters the data should be divided into and the number of iterations of the function. The latter is manually set appropriately.

The number of clusters is dependent on a metric returned by the k-means function, which is the sum of all point to centroid distances in each cluster. A high distance sum indicates a bad clustering and the number of clusters is therefore selected by iteratively running the k-means function until the relative change of the summarised distance is converging. Some parts of the data set have objects that
Methodology

Figure 3.7: Interior extraction from a point cloud. Left image shows the initial measurement without the roof and right image shows the extracted interior. The position of the sensor is marked in both scans for clarity.

are too close to other parts of the interior, which causes the clustering function to run with an incorrect number of clusters. In those cases, the clustering function is manipulated manually to the correct number, or discarded if no sufficient clustering could be performed.

Object Models

The object models are created manually in Matlab. Every model is made of a number of vertices and triangular faces with every face connected between three vertices. Together, they make a closed surface, representing the size and shape of the object. The complexity of the models can be varied by changing the number and position of the vertices. For example, a simple model of an armchair can be represented by eight vertices and twelve faces, making a cuboid with a certain width, length and height. In order to distinguish the front of an object from its back, two of the vertices are lowered. This is the typical level of complexity used in this thesis. Examples of constructed models are illustrated in Figure 3.9. An important factor for future processes is that the total surface of the object models remains convex.

The created object models are presented in Table 3.1. In the data sets, certain objects are very closely placed, making it hard to make separate clusters for each object. An example of this is the manikin sitting on a chair. This clutter is bypassed by making the models fit many close objects.

Model Matching

The process of matching targets and models consists of two steps. Firstly, an alignment of target and model and then a derivation of an error metric that decides the effectiveness of the match.
Figure 3.8: A typical result of the clustering function. The point cloud of the interior is clustered in three clusters, an armchair (red), an sitting person (blue) and a fraction of a sofa (green).

Figure 3.9: 3D models of some expected objects to be found in the data set. All dimensions are in meters.

The yaw angle relative to the $x$-axis in the sensor frame is the only interesting orientation of the object, since it is assumed that the objects are stationary on the horizontal floor. This results in constant zero-values for all roll and pitch angles. Therefore, the rectangle estimation (as explained in Section 2.3) is only used to fit a rectangle around the target from a top view, returning an estimated size and orientation of the target. The size is determined from the corner points of the
Table 3.1: The object models used in the model matching. Note that the dimensions are based on the longest sides of the models.

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Identifier (char)</th>
<th>Dimensions [m]</th>
<th>Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Armchair</td>
<td>a</td>
<td>$0.95 \times 0.85 \times 0.7$</td>
<td>8</td>
</tr>
<tr>
<td>Sofa</td>
<td>b</td>
<td>$1.6 \times 0.85 \times 0.9$</td>
<td>8</td>
</tr>
<tr>
<td>Sitting Person</td>
<td>c</td>
<td>$1.3 \times 0.6 \times 1.0$</td>
<td>8</td>
</tr>
<tr>
<td>Table</td>
<td>d</td>
<td>$0.95 \times 0.85 \times 0.7$</td>
<td>8</td>
</tr>
<tr>
<td>Fire Place</td>
<td>e</td>
<td>$1.2 \times 0.45 \times 0.5$</td>
<td>8</td>
</tr>
<tr>
<td>Tiny Table</td>
<td>f</td>
<td>$0.45 \times 0.4 \times 0.5$</td>
<td>8</td>
</tr>
</tbody>
</table>

rectangle and its longest side is aligned with the x-axis in order to estimate the orientation. Figure 3.10 shows two examples of how the rectangles are fitted to the targets.

![Rectangle estimation](image)

Figure 3.10: Rectangle estimation of (a) an armchair and (b) a person sitting on a chair. Both images are shown in top view.

Note that the rectangle estimation does not separate between what is front or back of the target. That is why the yaw angle receives two alternative values, one in the original position and one rotated $180^\circ$. All future functions in the model matching algorithm have to test both of these cases.

The estimated size and orientation is used to align the target point cloud to the object model. The yaw angle is used to create a rotation matrix and the position and size of the target is used for translation in three dimensions. All models are simply placed along the axis of sensor’s local coordinate system with one vertex in the origin. Figure 3.11 shows a target point cloud aligned with an armchair box model in both alternative orientations.

The quality of the match between target cloud and the object model is partly decided by the error function as explained in Section 2.3. The function calculates
The matching function matches every target with every object model available.

**Figure 3.11**: A target point cloud matched with an armchair model. The left figure shows the original orientation returned by the rectangle estimation, while the right shows the same target rotated $180^\circ$. In this case the original orientation is correct, as shown by the lower CEM value.

A relative mean square error (RE) based on how close the points are to the faces of the object model. Points far away from the surfaces results in a higher error score, indicating a poor match or even a non-match that is to be discarded.

With many simple and similar object models, the RE is not enough of an error metric to distinguish between objects of similar shape and size. A complementary function that adds weights to the RE is implemented to make the distinction more robust. The function checks what percentage of the target’s points are outside of an object model by calculating the scalar product between each point and all face normals. This requires the object models to be convex. The centre point of the model is first used to force all normals to point outward from the object. If the scalar product between the point and any normal is positive, then that point is marked as being outside the object. The weight $w$ is calculated as

$$w = 1 + \frac{N_{\text{outside}}}{N_{\text{inside}}},$$

where $N_{\text{outside}}$ and $N_{\text{inside}}$ are the number of points outside and inside the object models surface. Targets with a high percentage of the points outside the model receives a bigger weight, inferring a faulty match. The weight $w$ is then multiplied with the RE to form the combined error metric (CEM)

$$CEM = w \cdot RE$$

used for matching. Figure 3.11 shows a case where CEM decides the quality of the match.

The matching function matches every target with every object model available.
and selects the match with the lowest CEM. Target clouds that does not have any corresponding library model are discarded through a threshold on the CEM. This prevents unknown targets from receiving false matches and are instead ignored. A 3D centre point of the object model is calculated for each matched target using the orientation extracted by the rectangle estimation and is then transformed back to the original position of the object in the local coordinate system.

Every extracted object is returned as a structure containing an identifier to a single object model, as well as a 3D centre point and an orientation in the sensor frame, according to (3.27).

### 3.6.4 Object Association

Previously seen objects stored in the map are associated with newly seen objects extracted from the model matching. This is implemented partly to discard faulty matches made by the model matching, but most of all to separate the observations into already seen objects and new ones. Two objects are declared to represent the same physical object if they:

- share the same object model identity and
- are reasonably close to each other.

Algorithm 4 shows the process of object association, where every newly observed object is matched against every object in the map, following the criteria above. The model identity is fairly easy to check with a simple comparison of the identifier that each object has available. If the identifier of a measured object does not have a correspondence in the map, then one of two alternatives are assumed. Either the observed object is an unseen object that should be added to the map, or the model matching has made an incorrect detection (false positive) and the match should be discarded. In order to distinguish between the two possibilities, the Mahalanobis distance metric is applied as explained in Section 2.4.2 and implemented similarly to the implementation in Section 3.5.4. The gate value is set low enough so that if the object is too physically close to any other object in the map, regardless of orientation, then it is deemed to be the product of an incorrect association. If all values are above the gate value, the object is deemed to be a new object.
Algorithm 4 The object association algorithm.

input: measObjects, mapObjects
output: MatchedObjects, newObjects

for every newly observed object do
    potentialMatches = CompareIdentifiers(measObject, mapObjects)
    if empty(potentialMatches) then
        \[ M = \text{MahalanobisDistance}(\text{measObject, mapObjects}) \]
        if all(M) > PositionalGateValue then
            newObjects = measObject
        end if
    else
        for each potential match do
            M = MahalanobisDistance(potentialMatch)
            if M < MatchGateValue then
                MatchedObjects = potentialMatch
            end if
            if M > NewObjectGateValue then
                newObjects = measObject
            end if
        end for
    end if
end for

newObjects = RemoveNewObjectsWithMatches(newObjects, MatchedObjects)

In a similar way, objects that have the same identifier are not necessarily deemed to be the same, because there might be several objects of the same type in different places. An object in the measurement is matched with an object in the map if the Mahalanobis distance is below a certain gating threshold and the object is declared to be a new object if the distance metric is above another threshold. Everything in between these two thresholds are seen as bad readings of the same object and are discarded.

3.6.5 Map Augmentation

The initialisation of new objects is done with the undelayed initialisation [27], similar to the map augmentation of the EKF-SLAM found in Section 3.5.5. There are however a few differences with the introduction of objects that have to be accounted for. These differences mainly concern the separation and order of landmarks and objects in the map, which causes the map augmentation to be separated into two parts, the initialisation of new objects and the initialisation of new landmarks. They are performed after each respective measurement update.

Initiation of New Objects

Newly observed objects \( y^o \) that are not matched are transformed to the global coordinate system and added to the object map. The state \( X \) is updated according
Methodology

\[
X^+ = \begin{pmatrix}
p \\ M \\ O \\ O^{new}
\end{pmatrix}.
\] (3.35)

\(O^{new}\) contains all new objects \(o^{new,i}\), which are given by

\[
o^{new,i} = g(p, y^{o,i}), \quad i \in [1, \ldots, N_o],
\] (3.36)

where \(N_o\) is the number of new objects \(y^o\) and

\[
g(p, y^o) = \begin{pmatrix}
(r(p^\phi)y^{o,xyz} + p^{xyz})_p \\
-p^{\phi} - y^{o,\phi}
\end{pmatrix}
\] (3.37)

is a transformation of a newly found object in the local coordinate system to the global coordinate system, using the rotation \(r\) from (3.2) and the current sensor pose \(p\).

In addition to adding the new observations to the map, the uncertainties of each added object have to be added to the state covariance matrix by

\[
P^+ = \begin{pmatrix}
ppp & ppm & ppo & (Gp pp)^T \\
ppM & pMM & pMO & (Gp pm)^T \\
pOp & pOM & pOO & (Gp po)^T \\
Gp pp & Gp pM & Gp pO & Gp p + Ga RGa^T
\end{pmatrix},
\] (3.38)

where the uncertainties are initialised using the Jacobian \(G = \text{grad}(g(p, y^o))\), which describes the correlation between the objects and the rest of the system. The Jacobian is implemented as

\[
G = \begin{pmatrix}
\frac{\partial g}{\partial p} & \frac{\partial g}{\partial o}
\end{pmatrix} = \begin{pmatrix}
G^p & G^o
\end{pmatrix},
\] (3.39)

where

\[
G^p = \begin{pmatrix}
-sin(p^\phi) & -cos(p^\phi) & 0 & 0 \\
cos(p^\phi) & -sin(p^\phi) & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\] (3.40)

and

\[
G^o = \begin{pmatrix}
-cos(p^\phi) & sin(p^\phi) & 0 & 0 \\
-sin(p^\phi) & -cos(p^\phi) & 0 & 0 \\
0 & 0 & -1 & 0 \\
0 & 0 & 0 & -1
\end{pmatrix}.
\] (3.41)

Initiation of New Landmarks

The state augmentation with new landmarks is similar to the regular EKF-SLAM in Section 3.5.5, but with the addition of correlations with objects in the map.
The state is updated by

\[ X^+ = \begin{pmatrix} p \\ M \\ M_{\text{new}} \\ O \end{pmatrix} \]  

(3.42)

and the state covariance is updated accordingly

\[ P^+ = \begin{pmatrix} p^{pp} & p^{pM} & (G^p p^{pp})^T & p^{pO} \\ p^{Mp} & p^{MM} & (G^p p^{pM})^T & p^{MO} \\ G^p p^{pp} & G^p p^{pM} & G^p p^{pT} + G^2 R G^z^T & G^p p^{pO} \\ p^{Op} & p^{OM} & (G^p p^{pO})^T & p^{OO} \end{pmatrix}, \]  

(3.43)

where \( G = (G^p \ G^z) \) is given by (3.24).

### 3.6.6 Landmark Reduction

One of the possible advantages of the ATR-SLAM algorithm, is the replacement of landmarks with objects and thus reducing the size of the map. This is also done to prevent the same information in the scans to be used twice, which is a consequence of using the same 3D data to estimate both objects and landmarks. In a way, this can be seen as an association of extracted NARF landmarks and objects, where positive matches means the removal of landmarks. This is performed before the landmark association, preventing point inside the objects from being associated with points in the map.

Extracted NARF landmarks are removed if the space already is occupied by an object. This is implemented by checking what 3D points are inside and outside the object models, similar to the weight function used in the model matching. Every object stored in the map has an identifier that corresponds to a model in the library, which is loaded to determine the size and shape of the objects. Together with the stored position and orientation, the objects receive a 3D representation in the map. The object models are transformed from the global to the local coordinate system in order to align with the NARF points in the sensor frame. This is performed using the position estimate produced by the object based measurement update. Normals for each face of the objects are also calculated and made to point outward from the objects.

For every object in the map, the scalar product between each landmark and each face is calculated. Any negative result of this calculation marks the landmarks as being inside the object. All marked landmarks are removed from the measurements before entering the landmark association.

### 3.7 Evaluation

In order to evaluate the performance of the ATR-SLAM algorithm, a number of tests are performed:
• Test one: Single path. A one-directional run through a corridor.

• Test two: Back and forth run through a corridor.

• Test three: Large Scale SLAM. A run through the whole facility.

These tests are meant to cover aspects of the SLAM problem such as consistency, loop closure and computational complexity, as explained in Section 2.1.2. The impact of the ATR algorithm is investigated as both developed SLAM methods are put through these tests.

All test are performed offline in Matlab with scenarios based on the fire facility data set described in Section 1.5.2. Real point clouds taken from the data set are used, but the order of the scans and the field of view sections of the point clouds are manipulated. The order in which the scans are loaded represent the trajectory of the sensor carrier as it is moving throughout the fire facility. By only using specified 120° sections of each 360° scan, as explained in Section 3.4.1, a single scan can serve as two separate measurements, as long as the same data is not used twice. This becomes important in the evaluation of loop closure as it provides the possibility to have the sensor carrier move along a path in the data set and back again, going through previously visited scan locations.

In all tests, the impact of the ATR algorithm is analysed by first running the regular EKF-SLAM and then the ATR-SLAM. The following characteristics are evaluated in each test:

• Consistency. A comparison between the model and the true trajectory and map is made. The true scan positions and a layout of the test area are available from the data, and are superimposed on the model in 2D. A corresponding 3D model of the SLAM map is also included for clarity.

• Level of uncertainty. A confidence interval level (95%, 2σ) in two dimensions on the pose of the sensor carrier, the landmarks and the objects is included to illustrate the level of uncertainty. This is especially important in the loop closure test, where the consistency of the map should increase and the uncertainty decrease. Note that estimated uncertainties are demonstrated in the tests, which are more optimistic than the true uncertainty.

• Computational complexity. The increased time consumption of the ATR algorithm versus the savings gained from a reduced dimensionality of the map is analysed. This includes timings of the different algorithms as well as the ratio between the number of removed landmarks and the rest of the map.

• Target Detection. The likelihood of correct target detection by the ATR algorithm is also evaluated by comparing found objects with objects that are matched by the object association. Note that this is not a complete and conclusive evaluation of any improvements of the ATR algorithm, but it is more a evaluation of the ATR-SLAM’s potential to reduce the number of false positives.
There are many parameters that affect the performance of the SLAM algorithms. A list of all parameters along with their respective values and a brief description are shown in Appendix A. The state is initialised in the true position of a scanning location with low initial uncertainty. This represents the assumption that the original position of the sensor is known to always start in the origin.

In all tests, the noise of the inertial input and the point clouds are implemented as described in Section 3.4. Noise is added to the signals and is assumed to be white Gaussian noise (WGN) with a certain standard deviation for specific variables. Measurement noise in each 3D point from the laser scanner is set to have a standard deviation of 0.001 m in all dimensions. The standard deviation of the noise from the inertial input is set to 0.1 m in $xy$ and 0.01 m in $z$ and 1° (or 0.0175 rad) in bearing. In every test, the same seed of noise is used for both algorithms. The 5 mm$^3$ resolution data set was selected for all test as the performance of the NARF extraction was significantly improved over the 5 cm$^3$ data set.

3.7.1 Test One: Single Path

The first test is a short pathway from left to right through the scan positions seen in Figure 3.12. This is the part of the data set with the highest density of scans and also the area of the fire facility with three distinct objects that are recognisable by the ATR algorithm. The purpose of this test is to examine the impact of the ATR algorithm in an environment where detectable objects are present. Each measured scan has an 120° field of view and the scans are selected to overlap so that matches of already seen landmarks and objects can be found.

3.7.2 Test Two: Back and Forth

The purpose of this test is to investigate if the use of objects gives a more robust loop closure. The test scenario starts in the same location as the start of test one and moves through the same scan positions to the end of the corridor. Here, the sensor carrier also leaves the room for a short while before returning to the starting location. When sensor carrier re-enters the room, loop closure should occur, and the built up uncertainty from leaving the room should decrease. The ground truth of the scenario is shown in Figure 3.12.

3.7.3 Test Three: Large Scale SLAM

The third test will run through most of the fire facility, as seen in Figure 3.13. It is meant to demonstrate how the two implemented algorithms handle a prolonged scenario, where the map grows large and the error accumulation increases. The test will also show whether the reduction of landmarks will be more or less effective in a large map.
Figure 3.12: Section of the fire facility data set used in test one (top) and test two (bottom). The arrows show the direction in each scan. There are three distinct objects in the test area: sofa(b), armchair(a) and a sitting person(c). There is also a loose sofa pillow on the floor in front of the sofa.
Figure 3.13: Section of the fire facility data set used in test three. The sensor carrier starts in the top, moves down to the bottom and then returns to the starting location. There are six distinct objects in the test area: a sofa(a), an armchair (b), a sitting person(c), a table(d), two fireplaces(e) and a tiny table(f).
This chapter presents the results of tests used for evaluation of the system.

### 4.1 Test One: Single Path

The results highlight the consistency and computational complexity of the regular EKF-SLAM and the ATR-SLAM. These are produced under the conditions described in Section 3.7.1.

#### 4.1.1 Consistency

Figure 4.1 shows the resulting 2D map of both the EKF-SLAM and ATR-SLAM. Overall, the test shows that both implemented methods can be used for localisation and mapping, but that the ATR-SLAM is more consistent.

The results from the EKF-SLAM in the top graph show that the measurement update using NARF points is closer to the true positions than the predicted position. The map of the room is also fairly consistent, i.e. still useful for localisation, even though parts of the objects and the walls are slightly bent and contracted. However, the uncertainties in 2D position are high. The growth in uncertainty, especially in the direction orthogonal to the corridor, is due to the fact that new observations are concentrated within the field of view, giving little bilateral information. Table 4.1 show the number of found and matched landmarks in each position. The relatively high number of matches in the early scans are concentrated on the objects that are part of the field of view.

The results from the ATR-SLAM in the bottom graph show an improved positioning of the sensor carrier compared to the EKF-SLAM. This also applies to the
Figure 4.1: Result of the EKF-SLAM (top) and the ATR-SLAM (bottom) after a run of the test scenario, seen from above. The trajectory goes from the top left corner to the bottom right. The red ellipses show the uncertainty of the 2D sensor position (xy) within an 95% confidence interval.
4.1 Test One: Single Path

Table 4.1: Found and matched NARF points in each scan position for EKF-SLAM.

<table>
<thead>
<tr>
<th>Scan Position</th>
<th>#Found</th>
<th>#Matched</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>299</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>308</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>227</td>
<td>41</td>
</tr>
<tr>
<td>4</td>
<td>220</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>172</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>158</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>358</td>
<td>8</td>
</tr>
</tbody>
</table>

mapping, where the walls align better with the ground truth, when objects are included. The uncertainty in position is significantly lower for ATR-SLAM than for EKF-SLAM. This effect on the uncertainty show that the objects are better defined features with less ambivalence, allowing the system to trust those measurements more than the landmarks. Test one is a scenario where objects are always present in the facility that the ATR algorithm can detect. Found and matched objects in each scan are displayed in Table 4.2. It also shows that a large number of NARF matches have been replaced by objects compared to the EKF-SLAM.

Table 4.2: Found and matched objects in each scan position for ATR-SLAM, along with the number of matched NARF points. Note that NARF matches in early scan have been replaced by objects.

<table>
<thead>
<tr>
<th>Scan Position</th>
<th>Found Objects</th>
<th>Matched Objects</th>
<th># NARF Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>abc</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>abc</td>
<td>abc</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>ac</td>
<td>ac</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>a</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>ac</td>
<td>ac</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>c</td>
<td>c</td>
<td>25</td>
</tr>
<tr>
<td>7</td>
<td>ab</td>
<td>ab</td>
<td>6</td>
</tr>
</tbody>
</table>

The uncertainty of landmarks and objects in test one are shown separately for clarity in Figure 4.2. The size and shape of the ellipses depend on both the modelled measurement variance and the uncertainty of the sensor position estimates. The results show that the uncertainty of the objects are significantly lower than for the landmarks in ATR-SLAM. This represents the modelling of the measurement noise for both types features fairly well, as the objects are trusted more than the landmarks. It is worth noting that the landmarks are equally uncertain in both ATR-SLAM and EKF-SLAM. The graph also shows where re-identified landmarks are concentrated, because only the uncertainty of those associated at least once are shown.

Figure 4.3 shows the actual error in all state variables for both the EKF-SLAM
Figure 4.2: The uncertainty of associated landmarks (red ellipses) and objects (magenta) in test one for EKF-SLAM (top) and ATR-SLAM (bottom). The uncertainty of the objects are much lower than the uncertainty of the landmarks.
and the ATR-SLAM after test one. The actual error is the difference between the estimated state and the true state. The error is increased after each prediction, which adds a noise with a standard deviation of one degree. This is ideally corrected with the help of new observations. The performance of the EKF-SLAM varies between measurements, while the ATR-SLAM comparatively keeps a consistently low difference from the true poses. The estimated uncertainty in the corrected positions are also significantly lower for the ATR-SLAM than the EKF-SLAM. This is because of the higher confidence put into the object. Also note that both the error and the uncertainty rises slightly in bearing for the ATR-SLAM when only one object is seen.

Figure 4.4 shows a 3D image of the EKF-SLAM map and the ATR-SLAM map. Note that the landmarks are concentrated close to places of high surface change, resulting in the fragmented map, with landmarks around edges, corners and objects. The two images illustrates how the objects make the map more intelligible.

### 4.1.2 Computational Complexity

The result of test one also shows what impact the removal of landmarks has compared to the cost of running the ATR algorithm. A comparison of the number of landmarks replaced by objects are displayed in Table 4.3. The use of the landmark reduction results in a 39.3% removal of the landmarks. This is a significant reduction, as large chunks of the map are replaced by the centre points of the objects. The effect on time consumption is shown in Table 4.4. Although the concentration of objects is relatively high in test one, the area is relatively small and the reduced size of the map has little to no effect on the computational complexity compared to the cost of the ATR algorithm.

<table>
<thead>
<tr>
<th>Method</th>
<th>#Landmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKF-SLAM</td>
<td>933</td>
</tr>
<tr>
<td>ATR-SLAM</td>
<td>566</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Time Consumed [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKF-SLAM</td>
<td>8</td>
</tr>
<tr>
<td>ATR-SLAM</td>
<td>376 (370+6)</td>
</tr>
<tr>
<td>ATR-SLAM (no LR)</td>
<td>379 (370+9)</td>
</tr>
</tbody>
</table>

Table 4.3: Number of landmarks after the last scan for both algorithms. The effect of the landmark reduction in ATR-SLAM is shown by the reduced number of landmarks.

Table 4.4: Process time consumption from test one. The effect of landmark reduction (LR) is also shown. The values inside the parentheses show ATR algorithm’s time consumption and the rest of the ATR-SLAM separately.
Figure 4.3: The actual error between the true and the SLAM estimated sensor poses for EKF-SLAM (left) and ATR-SLAM (right) in test one. A 95% (2σ) confidence interval derived from the estimated covariance matrix is included for both predicted and corrected pose. The corrections made by the ATR-SLAM are more consistent than those made by the regular EKF-SLAM.
Figure 4.4: Resulting 3D map of the EKF-SLAM (top) and ATR-SLAM (bottom). The colour coding displays the points’ height from blue to red. The NARF extraction concentrates the landmarks along the wall edges, corners and objects. Note the removal of landmarks in the place of the objects.
4.2 Test Two: Back and Forth

The results from test two highlight the effects of loop closure in both the regular EKF-SLAM and the ATR-SLAM algorithm. Details of the test, including a ground truth map, are found in Section 3.7.2.

Figures 4.5 and 4.6 show the first and second half of test two, illustrating the results from both methods in parallel. The first figure shows a scenario similar to that of test one, but the sensor carrier exits the room, causing it to lose track as no matches can be found. This is visible through the lack of any correction of the estimated state and the increase in uncertainty. The sensor carrier then turns around in second figure, re-entering the room where loop closure follows in scan 10–12. This is visible from a drastic reduction of the uncertainty in conjunction with an improved estimate. After that, fewer landmarks and no objects can be detected in the remaining scans, causing the error accumulation to increase together with the uncertainty.

Note that the loop closure for the EKF-SLAM is less robust than ATR-SLAM, as the estimated state is relatively off mark, while simultaneously becoming more sure of its position due to false associations. This leads to further inconsistencies visible by the slight bending of the walls and a worse positioning than the ATR-SLAM. The ATR-SLAM on the other hand handles the loop closure much better due to the use of more robust object features that are more likely to be correctly re-identified.

Figure 4.7 shows the uncertainty of the landmarks for EKF-SLAM and 4.8 shows the uncertainty of the landmarks and objects for the ATR-SLAM. The uncertainties of the landmarks are slightly smaller for ATR-SLAM than the EKF-SLAM. However, a reduction in uncertainty from the loop closure is not visible in the associated landmarks.

Figure 4.9 confirms the indications from the 2D plot results from test two. The difference between the estimated and the true sensor pose increases, along with the estimated variance, when the sensor carrier leaves the room. Then it significantly corrects the error at loop closure, visible in scan positions 10–12. The corrections made by the ATR-SLAM are more consistent than those made by the EKF-SLAM.
4.2 Test Two: Back and Forth

Figure 4.5: Result of the first half of test two for the EKF-SLAM (top) and the ATR-SLAM (bottom), seen from above. The trajectory goes from the top left to the bottom. The red ellipses show the uncertainty of the 2D sensor position ($xy$) within an 95% confidence interval.
Figure 4.6: Result of the second half of test two for the EKF-SLAM (top) and ATR-SLAM (bottom), seen from above. The trajectory goes from the bottom to the top. The red ellipses show the uncertainty of the 2D sensor position (xy) within an 95% confidence interval.
4.2 Test Two: Back and Forth

Figure 4.7: The uncertainty of associated landmarks (red ellipses) using EKF-SLAM in the first (top) and second (bottom) halves of test two.
**Figure 4.8:** The uncertainty of associated landmarks (red ellipses) and objects (magenta) using ATR-SLAM in the first (top) and second (bottom) halves of test two.
Figure 4.9: The actual error between the true and the SLAM estimated states in each scan position for EKF-SLAM (left) and ATR-SLAM (right) after test two. A 95% (2σ) confidence interval derived from the estimated covariance matrix is also included for both the predicted and corrected pose.
4.3 Test Three: Large Scale SLAM

The results from test three are produced under the conditions described in Section 3.7.3. These results show the performance of both methods during a prolonged period of time, where the map grows large and the risk of inconsistencies increases.

4.3.1 Map Consistency

Figure 4.10 shows the result from the first half of the scenario in test three, plotted against the ground truth walls of the fire facility. At the turning point (end of the first half), the EKF-SLAM has a moderately inconsistent map, where the entire facility is bent. This is most likely due to a false association made early on, probably when the sensor carrier enters the second room. The ATR-SLAM on the other hand manages to stay fairly consistent, without any major inconsistencies visible. Note that the uncertainty increases steadily for both methods when the sensor carrier leaves the first room due to few landmarks being identified in the second and third rooms. The graphs are in 2D and from this perspective, landmarks seem to still be part of some objects. Most of these landmarks are far above the objects, as there are structures close to the ceiling from which the landmarks are extracted.

Figure 4.11 shows the result from the second half of test three. Here, the sensor carrier moves through already covered areas. The inconsistencies from the first half are partly corrected by the EKF-SLAM, as the landmarks in the third room starts to fit the walls. However, many inconsistencies remain and are added and as the sensor carrier reaches the end, the error between 2D position of the estimated sensor position and the true position are off, and not encompassed by the estimated variance. The ATR remains fairly consistent on the way back, with moderate reduction of uncertainty at the re-entry of the second building and a loop closure of the same significance as in test two.

Figures 4.12 show the uncertainty of the landmarks for EKF-SLAM, while Figure 4.13 show the uncertainty of the associated landmarks and objects for ATR-SLAM. Unlike test two, the difference in uncertainty before and after loop closure is clearly visible by the reduced size of the ellipses in both EKF-SLAM and ATR-SLAM. A comparison of the two figures show that the landmark uncertainties for ATR-SLAM are smaller than for EKF-SLAM.

Figure 4.14 shows how the uncertainty stays fairly high as soon as the sensor carrier leaves the first room, as well as a distinct loop closure at re-entry. The figure also shows that the uncertainty stays on par with the predicted for a large part of the scenario, i.e. no significant correction are made, as too few associations are made. The corrected variance then starts to decrease after the turning point at scan 17.
Figure 4.10: Result of the first half of test three for the EKF-SLAM (top) and ATR-SLAM (bottom), seen from above. The trajectory goes from the top to bottom. The red ellipses show the uncertainty of the 2D sensor position (xy) within an 95% confidence interval.
Figure 4.11: Result of the second half of test three for the EKF-SLAM (top) ATR-SLAM (bottom), seen from above. The trajectory goes from the bottom to the top. The red ellipses show the uncertainty of the 2D sensor position (xy) within an 95% confidence interval.
Figure 4.12: The uncertainty of associated landmarks (red ellipses) using EKF-SLAM in the first (top) and second (bottom) halves of test three.
Figure 4.13: The uncertainty of associated landmarks (red ellipses) and objects (magenta) using ATR-SLAM in the first (top) and second (bottom) halves of test three.
Figure 4.14: The actual error between the true and the SLAM estimated states in each scan position for EKF-SLAM (left) and ATR-SLAM (right) after test three. A 95\% (2\sigma) confidence interval derived from the estimated covariance matrix is also included for both the predicted and corrected pose.
4.3.2 Computational Complexity

The result of test three also shows what impact the removal of landmarks has on a large map. A comparison of the number of landmarks replaced by objects are displayed in Table 4.5. The use of the landmark reduction results in a 17.7% removal of the landmarks. This is a less significant reduction compared to test one, because the objects make up a smaller share of the world. Figure 4.15 shows the growth of number of landmarks throughout the test.

Table 4.5: Number of landmarks after the last scan of test three for both algorithms. The effect of the landmark reduction (LR) in ATR-SLAM is shown by running the same scenario twice, with and without the LR.

<table>
<thead>
<tr>
<th>Method</th>
<th>#Landmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKF-SLAM</td>
<td>3758</td>
</tr>
<tr>
<td>ATR-SLAM</td>
<td>3174</td>
</tr>
</tbody>
</table>

Figure 4.15: The number of landmarks at each scan of test three for the EKF-SLAM (red dotted line) and the ATR-SLAM (blue line).

The effect on time consumption is shown in Table 4.4. These numbers show two things:

- The use of landmark reduction improves the time consumption when comparing ATR-SLAM with and without LR.
- A comparison between time consumption the EKF-SLAM and ATR-SLAM show that the cost of the ATR algorithm is too significant to be compensated by the reduction of landmarks.
However, it is worth noting that the 1133 s spent by the ATR algorithm would be reduced to 245 s if 5 cm$^3$ data set was used instead of the 5 mm$^3$ data set. The performance of the ATR is barely changed between the data sets, whereas the NARF becomes less optimal.

**Table 4.6:** Process time consumption from test three with NARF extraction time excluded, because it is the same for all three processes. The effect of landmark reduction (LR) is also shown. The numbers within the parentheses show the ATR algorithm’s time consumption and the rest of the ATR-SLAM separately.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time Consumed [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKF-SLAM</td>
<td>240</td>
</tr>
<tr>
<td>ATR-SLAM</td>
<td>1309 (1133+176)</td>
</tr>
<tr>
<td>ATR-SLAM (no LR)</td>
<td>1374 (1133+241)</td>
</tr>
</tbody>
</table>
This chapter develops on the results presented in the previous chapter and the implemented methods.

5.1 Results

The results from the evaluation show that the ATR-SLAM outperforms the EKF-SLAM whenever objects are available for matching. In all tests, the ATR-SLAM produces a more consistent map, with less uncertainty in sensor pose. The faults of the EKF-SLAM such as its sensitivity to false associations and loop closures can be compensated using well defined features in the form of objects. Association of objects have a high chance of being correct, making it possible to assign high confidence in the information provided by newly observed objects. This significantly improves the consistency of the SLAM map, especially when multiple objects are matched. The results also show that the 3D point features extracted using NARF are useful landmarks for localisation when objects are not available. They are however more ambiguous than the objects, forcing the algorithm to model these measurements with higher uncertainty in order to moderate the impact of incorrect associations. Matching of objects also improves the landmark association by replacing the predicted sensor pose with a better estimate. This makes it easier for the landmark association to discard false matches, which is partly based on the uncertainty of the sensor pose.

In addition to a more robust SLAM algorithm, the results also show that the use of objects reduces the amount of data needed to be stored in the map. When the environment contains a lot of objects, which would be normal in a residence, test one shows that the number of landmarks are reduced by 40%. This leads to
a considerably smaller sized version of the estimated covariance matrix that the
SLAM algorithm has to use in its calculations. However, the ATR-SLAM does not
produce a significant enough reduction to overcome the complexity introduced
by the implemented ATR. This depends on the length of the scenario as the cost
of the ATR algorithm does not increase with map size, while the SLAM algorithm
does. There is therefore a point where the ATR-SLAM using landmark reduction
becomes less computationally demanding than EKF-SLAM using a larger map.
It is important to note that the tests do not include the extraction of the interior,
i.e. the removal of floors and walls, which is performed manually in pre-process
instead of automatically. An autonomous program based on RANSAC (Section
2.3) would add an significant increase in complexity. Even though the EKF scales
quadratically with the number of landmarks, the gains of the landmark reduc-
tion does not overcome the added cost of the entire ATR algorithm in its current
form. Due to the lack of automation, there are no conclusive results regarding the
improvements of the ATR algorithm’s chance for correct target detection. How-
ever, the potential for better target detection manifests through the rejection of
false positives by the object association, but because of that the ATR-SLAM also
misses a lot of targets that should have been seen (false negatives). The position-
ing might have been improved by more correct detections, but the consequences
of one false object association would have been worse.

In general, the results presented here indicate that ATR-SLAM offers a trade-off in
favour of a more consistent and robust SLAM algorithm instead of speed. How-
ever, only one data set has been tested and there are several modifications and
improvements that can be made, which might make it possible to run a similar
algorithm more optimally in the future.

5.2 EKF-SLAM Implementation

There are a few issues concerning the EKF that can be raised. One issue is that
EKF-SLAM estimates the uncertainty more optimistic than the true uncertainty
[3]. The true uncertainty continues to grow as the map grows large while the es-
timated uncertainty stagnates. This is due to that the EKF performs linearisation
around the estimated states instead of the true states and the noise is modelled
as a Gaussian distribution, even though the actual noise might not be Gaussian.
These are also the reasons why EKF can not guarantee convergence. EKF-SLAM’s
sensitivity to false associations is noticeable when using NARF landmarks, where
sudden inconsistencies in the map become visible.

The use of NARF can be more effective, as its lacking performance is the reason
why the denser and more time consuming data set is used. The main problem
is that the NARF descriptors do not provide enough consistently re-identifiable
points between scans. The reasons behind this might be that the calibration is
insufficient, or that the distance between scans are too large. There might also
exist feature extraction methods more suitable for the used data set.

The association of landmarks based on NARF can be improved, partly because
of the descriptor issues mentioned above, but also because of inefficient use of the Mahalanobis gating. A combinatorial optimization algorithm, such as the Hungarian method [13], based on the Mahalanobis distance could be performed before the association of NARF descriptors, or possibly in conjunction with it if it were to be implemented in C++. This would most likely improve the latter association as a large number of improbable matches with similar NARF descriptors would be discarded. This would also replace the current rough Euclidean gate, which at the moment prevents the algorithm from making correct matches if the error in the estimated sensor position is large enough.

If time was not a valuable resource, an alternative association method would be to use batch validation techniques. Batch validation is a term that describes methods that associate multiple data points at the same time, such as joint compatibility branch and bound (JCBB) [19]. These methods exploit the geometrical relationships between landmarks in order to provide robust association.

5.3 ATR-SLAM Implementation

Many of the problems concerning the EKF-SLAM are shared by the ATR-SLAM. The impact of these issues is smaller thanks to better position estimates based on physical objects that are calculated before the association and measurement update based on NARF landmarks. The implemented ATR based on rectangle estimation might not have been the best choice for target detection in indoor environments, since it is developed for the detection of military ground vehicles. Detection of objects in indoor environments are harder as they often are close to other objects and the number of possible locations are higher.

It is important to note that the implemented ATR makes assumptions about the targets and fills in the gaps when information is missing (fractions of targets), which makes it much worse when it is wrong. A reasonable criticism of the evaluated scenarios is that the target extraction, object models and matching of objects are specifically adapted for the data set so that no false matches are made. For example, the extraction of the interior is performed manually even though it could be handled automatically. The clustering of the interior is corrected manually in cases of failure and the object models are made to fit the known targets of the data set. If a false object association is made, the damage to the localisation would be the equivalent of a false loop closure, causing inconsistencies as the error in position would increase while the uncertainty decreased. A general and more autonomous ATR than the implemented version could be created, such as one based on boosting [20] or one that could handle cluttered environments [12]. This is however deemed to be out of scope, since this thesis is more focused around the relation between SLAM and ATR.
5.4 Future Work

Since the work performed in this thesis is a proof of concept rather than a fully functional positioning system, the developed ATR-SLAM algorithm can be improved in several ways. Examples of possible improvements include:

- Additions to the ATR algorithm that properly automates the segmentation and clustering of the interior and/or replacement with methods that can handle cluttered environments.

- Optimisation of the implemented ATR algorithm in order to reduce the computational complexity, possibly an adaptation based on C++. Other methods of ATR available in PCL might be useful.

- Attachment of semantic attributes that creates connection between different kinds of objects, e.g. walls and roofs are orthogonal or parallel to other walls and floors.

- Additional feature extraction methods can be tested and potentially replace the implemented NARF.

- Data association with batch validation techniques that use geometric relationships between objects and landmarks could be used, or possibly an introduction of a combinatorial optimization algorithm.

- Tests on more data in different indoor environments, possibly with an imaging sensor more suitable for a first responder.
In this thesis, ATR-SLAM, a method combining SLAM and ATR has been developed. The implemented algorithm is able to locate and identify physical objects from point clouds through ATR based on rectangle estimation. ATR-SLAM produces 3D maps containing these objects as well as landmarks in the form of point features based on NARF. Position estimates of the sensor carrier are performed with model predictions and new observations using EKF. Association of objects is performed separately from association of landmarks, using each object’s well defined characteristics to match observations with models. ATR-SLAM also uses a form of association between landmarks and objects, where those landmarks describing already identified objects are removed.

The purpose of this thesis was to investigate if the combination of SLAM and ATR could produce more consistent maps, more robust loop closure and target detection than each algorithm individually. The performance of ATR-SLAM has been evaluated on high-resolution point clouds generated using a time-of-flight laser scanner. The results show that ATR-SLAM offers more robust associations and more consistent maps than EKF-SLAM using only landmarks based on NARF. The potential for better target detection is demonstrated, although no conclusive results are presented. The introduced cost in computational complexity by the ATR algorithm can be mitigated, because the objects are more space efficient representations. Landmarks that occupy the same 3D space as objects are removed, leaving the ATR-SLAM with a more efficient and more consistent map that is easier to interpret for a first responder.
Parameter Settings

Tuning parameters used in the evaluation of the system.

\[ P_0 = 9 \cdot 10^{-4} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.1 \end{pmatrix} \] (A.1)

\[ Q = 3 \cdot 10^{-2} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{pmatrix} \] (A.2)

\[ R_m = 3.6 \cdot 10^{-2} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \] (A.3)

\[ R_o = \frac{10^{-4}}{6} \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 10 \end{pmatrix} \] (A.4)
Table A.1: Parameters used in all tests. Any changed values between tests or unmentioned parameters are specified in each individual test.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$ (A.1)</td>
<td></td>
<td>Initial covariance matrix</td>
</tr>
<tr>
<td>$Q$ (A.2)</td>
<td></td>
<td>Variance of system noise</td>
</tr>
<tr>
<td>$R_m$ (A.3)</td>
<td></td>
<td>Variance of measurement noise (landmarks)</td>
</tr>
<tr>
<td>$R_o$ (A.4)</td>
<td></td>
<td>Variance of measurement noise (objects)</td>
</tr>
<tr>
<td>$ar$</td>
<td>0.001</td>
<td>Angular resolution (NARF)</td>
</tr>
<tr>
<td>$ss$</td>
<td>0.6</td>
<td>Support size (NARF)</td>
</tr>
<tr>
<td>$dist_{th}$</td>
<td>2.0</td>
<td>Rough Euclidean threshold [m]</td>
</tr>
<tr>
<td>$desc_{th}$</td>
<td>0.2</td>
<td>NARF descriptor threshold</td>
</tr>
<tr>
<td>$g_m$</td>
<td>3</td>
<td>Mahalanobis threshold (landmarks)</td>
</tr>
<tr>
<td>$g_{a,obj}$</td>
<td>20</td>
<td>Lower Mahalanobis threshold (objects)</td>
</tr>
<tr>
<td>$g_{b,obj}$</td>
<td>400</td>
<td>Higher Mahalanobis threshold (objects)</td>
</tr>
</tbody>
</table>


[8] J. Civera, A.J. Davison, and J. Montiel. Inverse depth parametrization for...


[28] Helmuth. Spath. *Cluster dissection and analysis : theory, FORTRAN programs, examples / Helmuth Spath ; translator, Johannes Goldschmidt*. Hor-


Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/

© Kristian Sinivaara