Master's thesis

2021

Martin Eek Gerhardsen

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Engineering Cybernetics

Martin Eek Gerhardsen

Fiducial SLAM For Autonomous Ferry

June 2021

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Fiducial SLAM For Autonomous Ferry

Martin Eek Gerhardsen

Engineering Cybernetics
Submission date: June 2021
Supervisor: Edmund Førland Brekke
Co-supervisor: Rudolf Mester

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Fiducial SLAM for Autonomous Ferry

Martin Eek Gerhardsen

June 2021

Master thesis, Cybernetics and Robotics
Supervisors: Edmund Førland Brekke and Rudolf Mester

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Preface

This report is the resulting master thesis for TTK4900, Engineering Cybernetics, presenting my work during the spring of 2021. I would like to thank my supervisors Edmund Førland Brekke and Rudolf Mester for taking the time to answering my questions and guiding me throughout this period.

I would also like to thank Egil Eide, for his work with milliAmpere and his assistance with the experiments done in this thesis and Erik Wilthil for always being available for practical question. Furthermore, I would like to thank Martin Græsdal, Kristian Auestad and Thomas Hellum for their assistance with planning and executing successfully the experiments, and to Marit Andersen and *Skipnes Kommunikasjon* for printing and transporting the fiducial markers used in the experiments.

Abstract

Traditionally, finding the global position of a vehicle is done using Global Navigation Satellite System (GNSS). GNSS measurements are often gathered using one or two antennae in communication with satellites, which may present a safety problem should something happen to either the antennae or the satellite communication, or if the vehicle should operate in a GNSS-denied environment. This thesis tests the precision and applicability of using fiducial markers as constant and artificial landmarks with known global positions in order to implement a visual Simultaneous Localisation and Mapping (SLAM) method for Autonomous Surface Vehicles (ASVs) operating in urban canal environments. This was performed by designing and implementing a Fiducial SLAM method based on the Incremental Smoothing And Mapping (iSAM2) framework. A fiducial pose factor and fiducial projection factor for adding fiducial markers to factor graphs were designed and implemented for the Fiducial SLAM method.

The Fiducial SLAM method was tested on real data from experiments conducted using the AprilTag fiducial markers, the corresponding detection algorithm and the Norwegian University of Science and Technolog (NTNU)'s autonomous ferry milliAmpere. Testing the method on the data gathered showed that using only the fiducial markers, good estimates of the system pose could be achieved. Testing also showed that if multiple cameras detect the same markers at the same time, better estimates were achieved than if only a single camera observed the fiducial markers. Extending the method further by including Inertial Mesurement Unit (IMU) and infrequent GNSS measurements, the resulting estimates were improved.

Sammendrag

Tradisjonelt sett er Global Navigation Satellite System (GNSS) benyttet for å finne den globale posisjonen til et kjøretøy. GNSS målinger innhentes ofte ved hjelp av én eller to antenner som kommuniserer med satelitter. Dette kan skape et sikkerhetsproblem dersom noe skulle skje med antennene, satellittkommunikasjonen eller dersom kjøretøyet opererer i områder uten GNSS. Denne rapporten tester presisjonen og anvendbarheten til AprilTag fiducial markører som faste og kunstige landemerker med kjent posisjon for å implementere en visuell Simultaneous Localisation and Mapping (SLAM)-metode for autonome overflatekjøretøy som opererer i ukjente kanaler. Dette ble testet ved å designe og implementere en Fiducial SLAM metode basert på Incremental Smoothing And Mapping (iSAM2) rammeverket. En fiducial pose faktor og en fiducial projection faktor for å legge til fiducial markører til i faktorgrafer ble designet og implementert for Fiducial SLAM-metoden.

Fiducial SLAM metoden ble testet på ekte data fra eksperimenter utført ved å bruke AprilTag fiducial markører, den tilhørende deteksjonsalgoritmen og Norges teknisk-naturvitenskapelige universitet (NTNU)'s autonome ferge milliAmpere. Testing av metoden på den innsamlede dataen viste at ved å kun bruke fiducial markører, kunne gode estimater av systemets pose bli oppnådd. Testingen viste også at dersom flere kameraer ser de samme markørene samtidig ble det bedre estimater enn dersom kun ett kamera observerer fiducial markørene. Ved å utvide metoden videre med å inkluderer Inertial Mesurement Unit (IMU) og lavfrekvente GNSS målinger, ble estimatene også forbedret.

Contents

Preface . . . . . **iii**
Abstract . . . . . **v**
Sammendrag . . . . . **vii**
Contents . . . . . **ix**
Acronyms . . . . . **xiii**
Glossary . . . . . **xv**
1 Introduction **1**
1.1 Motivation and Problem Description 1
1.2 Report outline 3
2 Notation **5**
2.1 Variables 5
2.1.1 Vectors 5
2.1.2 Matrices 5
2.1.3 Homogeneous Coordinates 6
2.1.4 Time Dependence 6
2.2 Coordinate Systems 7
2.2.1 The North East Down Coordinate System 7
2.2.2 The milliAmpere Body Frame 7
2.2.3 The Camera Coordinate System 8
2.3 Transformations and Pose 8
2.3.1 Transformations 10
2.3.2 Pose 11
3 Background **13**
3.1 Probability Theory 13
3.1.1 Bayes' Formula 13
3.1.2 Maximum A Posteriori Estimator 14
3.1.3 Multivariate Gaussian 14
3.2 Performance Measure 16
3.3 Odometry 16
3.4 Simultaneous Localisation and Mapping 17
3.5 Camera Geometry 19
3.6 Fiducial Markers 20
3.6.1 ArUco 21
3.6.2 AprilTag 21

3.7 Infinitesimal Plane-Based Pose Estimation . . . 25
3.8 Factor Graphs . . . 26
3.8.1 Factor Graph Implementations . . . 28
3.8.2 Georgia Tech Smoothing and Mapping . . . 28
3.8.3 SLAM++ . . . 29
3.8.4 g2o . . . 29
3.8.5 Caesar . . . 29
3.9 Lie Theory . . . 29
3.9.1 Differentiation on Lie Groups . . . 32
3.10 Incremental Smoothing And Mapping . . . 33
3.10.1 The Bayes Tree . . . 33
3.10.2 Incremental Inference . . . 34
3.10.3 Incremental Reordering . . . 35
3.10.4 Fluid Relinearisation . . . 36
3.11 Preintegration . . . 36
4 Fiducial SLAM . . . 39
4.1 System Overview . . . 39
4.2 Independent Camera Thread . . . 40
4.3 Fiducial Map . . . 41
4.4 Graph Creation . . . 41
4.4.1 Fiducial Factor . . . 42
4.4.2 Fiducial Subgraph . . . 46
4.5 Back-end . . . 47
5 Hardware and Software . . . 51
5.1 Hardware . . . 51
5.1.1 milliAmpere . . . 51
5.1.2 AprilTag Markers . . . 53
5.1.3 360 EO-IR Sensor Rig . . . 56
5.1.4 Global Navigation Satellite System . . . 57
5.1.5 Inertial Measurement Unit . . . 58
5.2 Software . . . 58
5.2.1 Robot Operating System . . . 58
5.2.2 EO Camera Driver . . . 58
5.2.3 AprilTag Detection Algorithm . . . 59
5.2.4 The Open Computer Vision Library . . . 59
5.2.5 AprilTag Fiducial SLAM . . . 59
6 Experiments . . . 61
6.1 Planning . . . 62
6.1.1 AprilTag Markers . . . 62
6.1.2 Data Recording . . . 63
6.2 Scenarios . . . 63
6.2.1 Scenario 1 . . . 64
6.2.2 Scenario 2 . . . 65
6.2.3 Scenario 3 . . . 65

6.2.4 Scenario 4 66
6.2.5 Scenario 5 67
6.3 Pre-processing 68
7 Results 71
7.1 Scenario 1 71
7.1.1 Unconnected Fiducial Pose Factor 72
7.1.2 Connected Fiducial Pose Factor 74
7.1.3 Unconnected Fiducial Projection Factor 77
7.1.4 Connected Fiducial Projection Factor 79
7.2 Scenario 2 81
7.2.1 Unconnected Fiducial Pose Factor 81
7.2.2 Connected Fiducial Pose Factor 86
7.2.3 Unconnected Fiducial Projection Factor 88
7.2.4 Connected Fiducial Projection Factor 90
7.3 Scenario 3 92
7.3.1 Unconnected Fiducial Pose Factor 92
7.3.2 Connected Fiducial Pose Factor 95
7.3.3 Unconnected Fiducial Projection Factor 97
7.3.4 Connected Fiducial Projection Factor 99
7.4 Scenario 4 101
7.5 Scenario 5 101
7.5.1 Unconnected Fiducial Pose Factor 101
7.5.2 Connected Fiducial Pose Factor 104
7.5.3 Unconnected Fiducial Projection Factor 106
7.5.4 Connected Fiducial Projection Factor 108
8 Discussion 111
9 Conclusion and Future Work 115
9.1 Conclusion 115
9.2 Future Work 115
Bibliography 119
A Additional Material 125
A.1 Sensor Transformations 125
A.2 Global Fiducial Marker Transformations 126

Acronyms

AR Augmented Reality. 20, 21

ASV Autonomous Surface Vehicle. v, 1, 3, 16, 71, 115–117

BRIEF Binary Robust Independent Elementary Features. xiv

COLAMD Column Approximate Minimum Degree. 35

DLT Direct Linear Transform. 24

EO Electro-Optical. 56, 58, 62, 63, 68, 84, 125

FAST Features from Accelerated Segment Test. xiv

GNSS Global Navigation Satellite System. v, vii, 1–3, 8, 17, 39, 48, 49, 51, 57, 58, 60, 62, 63, 71, 74, 112, 115, 116, 125

GTSAM Georgia Tech Smoothing and Mapping. 28, 59, 60, 117

IMU Inertial Mesurement Unit. v, vii, 16, 28, 36, 37, 48, 49, 58, 60, 63, 116, 125

INS Inertial Navigation System. 57, 58, 62, 64–68, 112

IPPE Infinitesimal Plane-based Pose Estimation. 25, 26, 42, 43, 59, 81, 82, 85, 111, 112, 115, 116

IR Infra-Red. 56, 63

iSAM Incremental Smoothing and Mapping. 3, 33, 35, 39, 40, 47, 60, 112, 115, 116

MAP Maximum A Posteriori. 14, 27

NED North East Down. 7, 8, 11, 12, 71, 73, 75, 78, 80, 83, 85, 87, 89, 91, 93, 96, 98, 100, 102, 105, 107, 109, 126

NTNU the Norwegian University of Science and Technolog. v, 1, 51, 61, 115

OpenCV Open Computer Vision. 8, 59, 82

ORB Oriented FAST and Rotated BRIEF. xiv, 29

ORB-SLAM ORB-SLAM. 29

PDF Probability Density Function. 14

PnP Perspective-n-Point. 25, 26, 42, 43, 59, 81, 82, 85, 112, 115, 116

RANSAC RANdom SAmple Consensus. 116

RMSE Root Mean Square Error. 16, 71–75, 78, 80, 81, 83, 87, 89, 91, 93, 96, 98, 100, 102, 105, 107, 109

ROS Robot Operating System. 58, 59, 68

RSE Root Square Error. 16, 71–75, 77–81, 83, 86–93, 95–102, 104–109

RTK Real Time Kinematics. 1, 57, 62

SLAM Simultaneous Localisation and Mapping. v, vii, xiv, 2, 3, 13, 14, 17–19, 21, 26–29, 33, 34, 39, 40, 42, 47, 59, 60, 67, 71–73, 75, 77–83, 86–93, 95–102, 104–109, 115–117

VO Visual Odometry. 17, 18, 39, 47, 116

Glossary

360 EO-IR Sensor Rig 360 degree EO and IR camera system. xv, 51, 56, 63, 65, 69, 101, 112, 125

AprilTag Visual fiducial system for fast and accurate detection. Used to refer either to the AprilTag pattern or detection algorithm. v, vii, 3, 21–26, 47, 53, 57, 59, 62–68, 81, 82, 84, 85, 101, 115–117, 126

camera synchronisation interface External synchronisation interface for the EO cameras on the 360 EO-IR Sensor Rig. 56, 58

Maritime Robotics Trondheim based company that, among other things, created the 360 EO-IR Sensor Rig used in this project. 58

milliAmpere NTNU's autonomous ferry. iii, v, vii, 1–3, 7, 39, 51–53, 56–58, 61–68, 71, 72, 74, 77, 79, 81, 82, 85, 86, 90, 92, 95, 99, 101, 113, 115, 125

NVIDIA TX2 Carrier Main computer on the 360 EO-IR Sensor Rig, from Connect Tech, ASG007. 56

pose Representing both the position and orientation of a system. v, vii, 11, 12, 16, 17, 25, 33, 34, 36, 40–45, 47–49, 59, 60, 62, 71–75, 77–83, 85–93, 95–102, 104–109, 111, 112, 115, 116, 126

Chapter 1

Introduction

1.1 Motivation and Problem Description

Autonomous systems are increasingly common in everyday situations, and many large organisations such as Google, Tesla and BMW are working on autonomous vehicles [9]. Autonomous Surface Vehicles (ASVs) are autonomous vehicles operating on the surface of the water, initially developed in 1993 by MIT [45]. After this, ASVs have been continuously developed and applied to different settings, such as autonomous ferries. Human errors are estimated to be the cause of over 61% of the total number of ferry accidents, and over 75% of the total number of fatalities, see Golden and Weisbrod [29]. Developing autonomous ferries could therefore decrease the number of accidents and fatalities significantly. Autonomous ferries have, like other autonomous vehicles, the possibility of being more cost effective and more environmentally friendly than human operated ferries [19]. Another benefit is that these ferries can be a cheaper alternative to both building new bridges and using manned ferries [61].

At the Norwegian University of Science and Technolog (NTNU), autonomous ferries are researched in the Autoferry project. The Autoferry project aims at researching concepts and methods around the development of autonomous passenger ferries for urban areas [2]. This includes developing a fully autonomous ferry to transport passengers between Ravnkloa and Brattøra in Trondheim, as a modern replacement for the previous rowingboat ferry Fløtmann, which was in service in this area until 1965 [66]. NTNU's autonomous ferry milliAmpere is the prototype for this project. The project will result in a fully electric ferry, able to transport passengers autonomously. MilliAmpere is described further in Section 5.1.1.

For autonomous systems, like milliAmpere, to operate without any human input, it is essential for the system to know its position. MilliAmpere uses a Real Time Kinematics (RTK)-Global Navigation Satellite System (GNSS), which can measure its position and heading with very good accuracy. However, this

method depends on the communication between the GNSS and the satellites. If this communication is blocked, the ferry will have a poor idea of its own position. Therefore, a backup-system to this is useful, if not essential, to ensure that milliAmpere can operate fully autonomously. One possible backup-system could be a Simultaneous Localisation and Mapping (SLAM) system, as it can be very accurate, multiple sensors can be used, and a map of the local environment is created. The use of multiple sensors mean that if one sensor fails there are others to take its place, and a more accurate understanding of the surroundings can be created and used for other purposes than just estimating the position of the ferry. Some aspects of SLAM for autonomous ferries have already been researched based on milliAmpere, and Skjellaug [59], Ødven [48] and Dalhaug [14] have explored lidar-based SLAM methods, while Fosso [25] explored infrared sensors.

One important aspect of SLAM methods is that, unless GNSS measurements are included explicitly in the method, the map and position estimates generated are relative the initial conditions. An underlying problem is that all global information used for position estimation is done using sensors on the autonomous system, which means that the entire map is defined relative the position of the system. If part of the map was known globally prior to starting the SLAM system, global information can be introduced to the SLAM method without being defined relative the position of the system. Because the map usually contains a large number of small map points, e.g. representing edges or corners in the environment, it can be difficult to create a map with prior global information.

Introducing artificial landmarks into the harbour environment can therefore be a potential solution, as the environment lacks naturally useful features for a SLAM map and one can easily measure the GNSS position of an artificial landmark. The use of artificial landmarks to estimate positions is not new, e.g. ultra wide-band localisation methods have been shown to be useful tools for position estimation by Strohmeier et al. [63], and have also been used to implement a SLAM method by Segura et al. [58]. A drawback with ultra wide-band technology is that it needs electricity in order to operate, which is difficult to operate with for harbour environments. In addition, the transmitted signals may be blocked.

Other methods such as UcoSLAM by Muñoz-Salinas and Medina-Carnicer [46] and TagSLAM by Pfrommer and Daniilidis [54] have been successful using fiducial markers, which are 2D markers similar to QR-codes, as artificial landmarks. These do not need any active input other than the initial placement, and are detected using camera images. While both UcoSLAM and TagSLAM were considered to be explored in this thesis, this was not done mainly because there was an interest in creating a multi-sensor SLAM method for milliAmpere and

the software would have to be modified to achieve this.

The purpose of this master thesis is to develop a SLAM system using fiducial markers as easily detectable and GNSS-fixed landmarks for ASVs operating in urban harbour environments. This is implemented using the Incremental Smoothing and Mapping (iSAM)2 optimiser, so that this might easily be added to a multi-sensor SLAM system, using the AprilTag fiducial marker. The SLAM problem was formulated as a factor graph, and two different fiducial factors were implemented. To test this method and the factors, data was gathered using milliAmpere for multiple scenarios.

1.2 Report outline

The structure of this thesis will follow the structure outlined in this Section. Chapter 2 will define the notation and the primary coordinate frames used in this thesis. Chapter 3 will outline the background information and mathematics necessary for the rest of the thesis, including some probability theory, SLAM, fiducial markers and factor graphs. The primary contribution of this work, which is the development of the Fiducial SLAM method, is described in Chapter 4. Chapter 5 lists the hardware and software used in the experiments conducted as part of this thesis, as well as discussing the specific implementation of the Fiducial SLAM method. Chapter 6 describes the experiments conducted as part of this thesis, and describes the datasets used to test the developed method, while Chapter 7 shows the results of these experiment by applying the Fiducial SLAM method to the different datasets. Chapter 8 and 9 contain the discussion and conclusion for this thesis, as well as future work.

Chapter 2

Notation

There exists multiple well known and yet different notational conventions for working with 3D geometry. In this chapter, the conventions used in this thesis will be outlined. This is an important part of any project, as otherwise uncertainty and mistakes may easily be introduced by mixing conventions.

2.1 Variables

In this section, the special notation for vectors, matrices, homogeneous coordinates, and time dependence will be introduced.

2.1.1 Vectors

Vectors are collections of numbers which represent some n variables, where n is called the dimension of the vector. In this thesis, vectors generally are represented as column vectors, unless otherwise stated. They are denoted using lower case, boldface letters:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2.1}$$

2.1.2 Matrices

Matrices can be thought of as a collection of m column vectors of dimensions n. In this thesis they are denoted using upper case, boldface letters:

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \dots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,m} \end{bmatrix} \tag{2.2}$$

2.1.3 Homogeneous Coordinates

Homogeneous coordinates are extensions to a state, where an n dimensional state is represented by an $n+1$ dimensional vector, see Equation (2.3). This allows e.g. projective transformations to be easily represented as matrices. When working explicitly with homogeneous coordinates, the vector notation will be expanded using a tilde, to indicate that the final dimension represents projection. A vector can be transformed into homogeneous coordinates by adding an extra value to the vector, defined as 1, see Equation (2.5). By dividing the vector element-wise by x_{n+1} and removing element x_{n+1} from the vector, a homogeneous coordinate may be converted back to its original coordinate system. If the elements within a homogeneous vector are divided by the last dimension, such that $x_n = 1$, the vector is denoted using a tilde, while the individual elements are not.

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x}_1 & \tilde{x}_2 & \dots & \tilde{x}_n & \tilde{x}_{n+1} \end{bmatrix}^\top \tag{2.3}$$

$$\tilde{\mathbf{x}} = \begin{bmatrix} \frac{\tilde{x}_1}{\tilde{x}_{n+1}} & \frac{\tilde{x}_2}{\tilde{x}_{n+1}} & \dots & \frac{\tilde{x}_n}{\tilde{x}_{n+1}} & \frac{\tilde{x}_{n+1}}{\tilde{x}_{n+1}} \end{bmatrix}^\top \tag{2.4}$$

$$\tilde{\mathbf{x}} = \begin{bmatrix} x_1 & x_2 & \dots & x_n & 1 \end{bmatrix}^\top \tag{2.5}$$

2.1.4 Time Dependence

As all these variables may be time dependent, variables may be denoted as either continuous or discrete time dependent. Continuous time dependence is denoted with round brackets and time defined as t, see Equation (2.6). Discrete time dependence is denoted with square brackets and time defined as k, see Equation (2.7). It is also sometimes useful to look at multiple time dependent variables at the same time. This is only really applicable for discrete time variables, so the set of variables in the range $[k_i, k_j]$ are denoted as Equation (2.8).

$$\mathbf{x}(t) \tag{2.6}$$

$$\mathbf{x}[k] \tag{2.7}$$

$$\mathbf{x}[k_i : k_j] \tag{2.8}$$

2.2 Coordinate Systems

A coordinate system, or coordinate frame, is a system in which geometric objects can be represented using base unit vectors. To indicate that a vector is represented in a specific coordinate system C, a subscript is used, see Equation (2.9). In this thesis, 3D coordinate systems will be used, and will, for consistency, follow the right hand rule. The right hand rule defines the orientation of the cross product of two vectors. Given two vectors $\mathbf{u}$ and $\mathbf{v}$, the orientation of the cross product $\mathbf{u} \times \mathbf{v}$ will follow the right hand rule: place the index finger in the direction of $\mathbf{u}$ and the middle finger in the direction of the $\mathbf{v}$, then the thumb will define the direction of $\mathbf{u} \times \mathbf{v}$. A coordinate system which follows the right hand rule, called a right handed coordinate system, is defined as one where the right hand rule is fulfilled between all unit vectors of the coordinate system. A few especially relevant coordinate systems will be quickly listed and described bellow.

$$\mathbf{x}_C = \begin{bmatrix} x_{0,C} & x_{1,C} & \dots & x_{n-1,C} \end{bmatrix}^\top \tag{2.9}$$

2.2.1 The North East Down Coordinate System

The North East Down (NED) coordinate system is defined as a local tangent plane to the Earth, relative to a specific geographical position, defined by latitide, longitude and altitude. The coordinate system is defined with the first dimension pointing north, the second pointing east and the final axis pointing down towards the centre of the Earth, see Equation (2.10). The NED notational convention will follow Fossen [24, pp. 16–19]. In this thesis, this will be defined as the world coordinate system, and is defined relative the geographic point *Piren* at latitude 63.4389029083°, longitude 10.39908278° and altitude 39.923. The calculations converting from global to local NED coordinates are done using software by Karney [41].

$$\mathbf{x}_{\text{NED}} = \begin{bmatrix} \text{north} & \text{east} & \text{down} \end{bmatrix}^\top \tag{2.10}$$

2.2.2 The milliAmpere Body Frame

The milliAmpere body frame, also called the *vessel center* ($\mathcal{V}$ for short), or simply the *body frame*, is defined by a black cross by the main hatch on milliAmpere. The first axis points towards the front of milliAmpere, the second points towards the starboard (right side) of milliAmpere, and the third axis points downwards. The transforms from milliAmpere to the mounted sensors relevant for this thesis are listed in Appendix A.1.

2.2.3 The Camera Coordinate System

The camera coordinate system is denoted using the Open Computer Vision (OpenCV) convention. Observing from the cameras point of view, the first axis is defined to the right, the second is defined downwards and the third is defined in the viewing direction. Conventionally these axes are called x, y, and z respectively. However, when specifically working with pixel values, u and v will be used instead of x and y respectively. Given an image, u starts at 0 in the top left corner, and increments by one for each pixel moving horizontally to the right, and v starts at 0 in the top left corner, and increments by one for each pixel moving vertically down.

2.3 Transformations and Pose

When working with robotic systems, it is often useful to define multiple coordinate systems, which makes the system more modular, and can be both easier to understand and work with. As a simple example, consider a robot with a GNSS using a camera to operate in a city. It would be useful to know the global position of the robot, which might be expressed in the NED coordinate system, but anything detected by the camera would be expressed in the camera frame. To know the global position of what the camera detects, one must transform the detection from the camera frame, to the global frame. See Figure 2.1 for an example of how such a scene might look. While the *detected object* may be defined as the point three metres in front of the camera, this is not necessarily its position in NED.

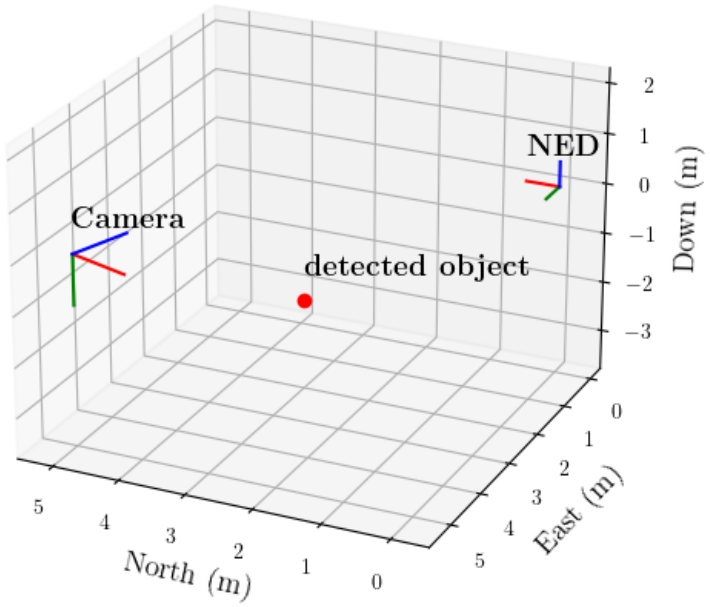


Figure 2.1: Example of a robotic system with multiple coordinate systems. For the visualised coordinate systems: red represents the first axis, green represents the second axis and blue represents the third axis. Generated using software from [34] and [18].

2.3.1 Transformations

Transformations between coordinate systems can be represented by a translation vector $\mathbf{t}$ and a rotation matrix $\mathbf{R}$. Transforming a point from coordinate system C_1 to C_2 is notationally defined as Equation (2.11). A variable which transforms some state from one coordinate system to another is subscripted with the original coordinate system and superscripted with the new coordinate system.

$$\mathbf{x}_{C_2} = \mathbf{R}_{C_1}^{C_2}\mathbf{x}_{C_1} + \mathbf{t}_{C_1}^{C_2} \tag{2.11}$$

An n dimensional translation vector, $\mathbf{t} \in \mathbb{R}^n$, represents how an object can be moved from one n dimensional coordinate system to another in terms of the coordinate systems unit vectors.

An n dimensional rotation matrix, $\mathbf{R} \in \mathbb{R}^{n \times n}$, represent how an object in an n dimensional coordinate frame must be rotated to be transformed into another coordinate frame. The rotation matrix is in the special orthogonal group in 3D ($SO(3)$), as defined by Equation (2.12).

$$SO(3) = \left\{\mathbf{R} \in \mathbb{R}^{3\times 3} \mid \mathbf{R}\mathbf{R}^\top = \mathbf{I}, \det(\mathbf{R}) = 1\right\} \tag{2.12}$$

When thinking of rotations as a combination of rotations around the coordinate system axes, one must keep in mind that the $SO(3)$ group is not commutative. Given the same angles to rotate around the three axes, different results may arise when the rotation starts with one axis before another. The rotation matrices around the three axes will follow the convention of Fossen [24, pp. 16–19]. The rotation matrix around the first axis, roll (ϕ), is defined as Equation (2.13), while the rotation matrix around the second axis, pitch (θ), is defined as Equation (2.14), and the rotation matrix around the third axis, yaw (ψ), is defined as Equation (2.15). Finally, a single rotation matrix composing all three angles can be defined as Equation (2.16).

$$\mathbf{R}_1(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.13}$$

$$\mathbf{R}_2(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.14}$$

$$\mathbf{R}_3(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

$$\mathbf{R}(\phi,\theta,\psi) = \mathbf{R}_1(\phi)\mathbf{R}_2(\theta)\mathbf{R}_3(\psi) \tag{2.16}$$

Homogeneous matrices are the standard for computing Equation (2.11) using a single matrix multiplication, and to simplify composing multiple transformations. This also means that 3D points are represented as homogeneous vectors. For 3D coordinate systems, these transformations are defined by 4×4 matrices, representing both the rotation and translation between the coordinate systems. This is why these transformations are denoted using the same convention as the matrices in Equation (2.2). These matrices are in the special Euclidean group in 3D ($SE(3)$), as defined by Equation (2.17) using the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$.

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \mid \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\} \tag{2.17}$$

As described earlier in this section, part of the usefulness of transformations is using multiple, modular transformations to e.g. express a point within another coordinate system. Given two transforms, e.g. the transform from NED to the robot, $\mathbf{T}_{\text{NED}}^{\text{robot}}$, and the transform from the robot to the camera, $\mathbf{T}_{\text{robot}}^{\text{camera}}$, the transform directly from NED to the camera, $\mathbf{T}_{\text{NED}}^{\text{camera}}$, can be found as:

$$\mathbf{T}_{\text{NED}}^{\text{camera}} = \mathbf{T}_{\text{NED}}^{\text{robot}} \mathbf{T}_{\text{robot}}^{\text{camera}}. \tag{2.18}$$

2.3.2 Pose

Pose describes a state containing both the rotation and translation between two coordinate system. Like with the transformations, notationally poses follow the matrix notation defined in Equation (2.2), as these are also represented as matrices. Poses are very closely related to the transformation matrices discussed above, though are more often used to describe a specific state rather than the transform necessary to go from one coordinate system to another. Using the example illustrated in Figure 2.1, if one is interested in both the position

and the rotation of the camera over time relative to NED, it can be simpler to imagine this specifically as a state. Still, the difference between a pose and a transformation is primarily due to convention. If one has the pose of a robot in NED, $\mathbf{X}^0_{\text{ned}}$, and then the pose odometry, $\mathbf{X}^1_0$, describing the change in pose, one may find $\mathbf{X}^1_{\text{ned}}$ similarly to above as the composition Equation (2.19).

$$\mathbf{X}^1_{\text{ned}} = \mathbf{X}^1_0 \mathbf{X}^0_{\text{ned}} \tag{2.19}$$

Chapter 3

Background

This chapter outlines the background information for the work done in this thesis.

3.1 Probability Theory

Probability theory is very useful for formulating and solving SLAM problems, as it allows states and measurements to be defined with uncertainty. This uncertainty often represent the inherent noise in measurements and states, as defined by e.g. the inaccuracies in measuring devices. Another advantage is that the more uncertain measurements and states can be *weighted* numerically by the noise, which means that untrustworthy measurements are suppressed. If a SLAM problem is formulated using probability theory, the problem can be defined as finding the most likely state given the known conditions. In this section, some background on relevant probability theory will be covered.

3.1.1 Bayes' Formula

Bayes' Formula (also known as Bayes' theorem, Bayes' rule, or Bayes' law), is an important equation in probability theory and in probabilistic SLAM methods as it shows how knowledge of variables may be updated given new evidence. As shown by Bar-Shalom et al. [3, pp. 47–48], this can be defined for events (Equation (3.1)), variables or probability density functions (Equation (3.2)), as well as between events and variables (Equation (3.3)).

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{3.1}$$

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)} \tag{3.2}$$

$$P(A \mid x) = \frac{p(x \mid A)P(A)}{p(x)} \tag{3.3}$$

3.1.2 Maximum A Posteriori Estimator

As the name implies, the Maximum A Posteriori (MAP) estimator, see Bar-Shalom et al. [3, p. 92], calculates the estimate of a state $\hat{\mathbf{x}}$ by maximising the probability function of the state $\mathbf{x}$ given the measurements $\mathbf{z}$, which can be summarised mathematically in Equation (3.4). Using Bayes' formula, this can be rewritten as Equation (3.5), and defining the likelihood function $l(\mathbf{x};\mathbf{z}) \propto p(\mathbf{z} \mid \mathbf{x})$, Equation (3.6). Therefore, the MAP estimate can be defined as the maximisation of the product of the likelihood function and the prior probability $p(\mathbf{x})$.

$$\hat{\mathbf{x}}^{\text{MAP}} = \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{x} \mid \mathbf{z}) \tag{3.4}$$

$$= \underset{\mathbf{x}}{\operatorname{argmax}}\, \frac{p(\mathbf{z} \mid \mathbf{x})p(\mathbf{x})}{p(\mathbf{z})} \tag{3.5}$$

$$= \underset{\mathbf{x}}{\operatorname{argmax}}\, l(\mathbf{x};\mathbf{z})p(\mathbf{x}) \tag{3.6}$$

3.1.3 Multivariate Gaussian

The Gaussian distribution is one of the most commonly used probability distributions, and as the states used in SLAM problems are almost always multivariate, their probability distributions must also be described as multivariates. For a vector state $\mathbf{x}$, a multivariate Gaussian probability distribution for this state can be defined by an expectation vector $\boldsymbol{\mu}$ and a symmetric positive definite covariance matrix $\mathbf{P}$ like Equation (3.7), as described by Brekke [7, pp. 41–50].

$$\mathcal{N}(\mathbf{x};\boldsymbol{\mu},\mathbf{P}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\mathbf{P}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\top}\mathbf{P}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{3.7}$$

The shape of the univariate Gaussian Probability Density Function (PDF) is well known, and an example generated with $\boldsymbol{\mu} = 0$, $\mathbf{P} = 1$, plotted for $\mathbf{x} \in [\boldsymbol{\mu} - 3 * \mathbf{P}, \boldsymbol{\mu} + 3 * \mathbf{P}]$ can be seen in Figure 3.1. Like a univariate Gaussian distribution, the peak of a multivariate Gaussian distribution is found at $\boldsymbol{\mu}$, while $\mathbf{P}$ describes the spread of the distribution. However, as the multivariate Gaussian describes a function $\mathbb{R}^n \mapsto \mathbb{R}$, these distributions are often visualised e.g. as probability ellipses, like in Figure 3.2. With multiple ellipses inside each other, the probability distribution can be visualised and an idea of where the weight of the distribution lies can be ascertained.

Because a large part of the multivariate Gaussian PDF is there to scale the function, one usually only study the exponent, defined by the quadratic form Equation (3.8). This defines the shape of the distribution, and as one is normally only interested in maximising a PDF, the scaling parameters are of less

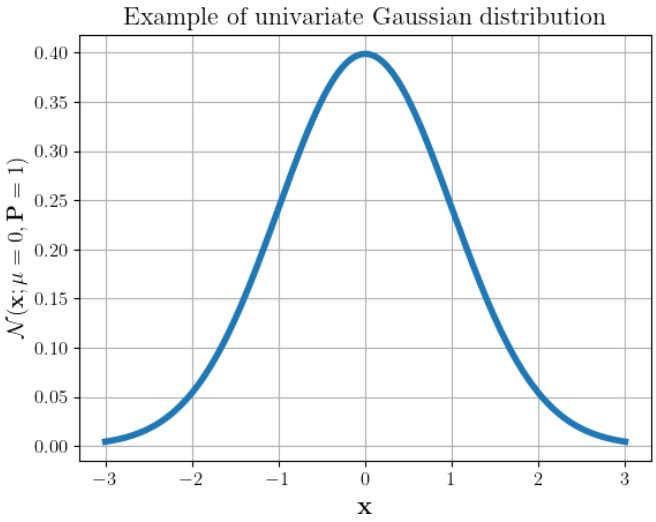


Figure 3.1: Univariate Gaussian distribution. Generated using the SciPy[36] and the Matplotlib[34] Python-libraries.

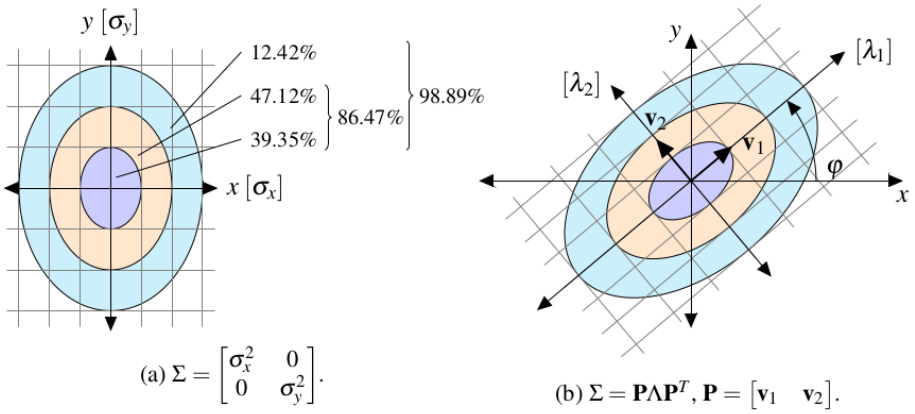


Figure 3.2: Multivariate, two dimensional, Gaussian distribution, showing probability ellipses. Image source[7, p. 42].

importance and can usually be discarded, often resulting in simpler calculations.

$$q(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu})^{\top} \mathbf{P}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \tag{3.8}$$

One will sometimes also find the multivariate Gaussian represented in *canonical form*. This form is represented as Equation (3.8), where $\mathbf{a}$ is defined by Equation (3.10), $\boldsymbol{\eta}$ is defined by Equation (3.11) and $\boldsymbol{\Lambda}$ is defined by Equation (3.12).

$$\mathcal{N}(\mathbf{x};\boldsymbol{\eta},\boldsymbol{\Lambda}) = \exp\left(\mathbf{a} + \boldsymbol{\eta}^\top \mathbf{x} - \frac{1}{2}\mathbf{x}^\top \boldsymbol{\Lambda}\mathbf{x}\right) \tag{3.9}$$

$$\mathbf{a} = -\frac{1}{2} n \ln 2\pi - \ln|\Lambda| + \boldsymbol{\eta}^\top \boldsymbol{\Lambda}\boldsymbol{\eta} \tag{3.10}$$

$$\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu} \tag{3.11}$$

$$\boldsymbol{\Lambda} = \mathbf{P}^{-1} \tag{3.12}$$

3.2 Performance Measure

To measure the performance of the estimation methods given a set of errors, the Root Mean Square Error (RMSE) and maximum Root Square Error (RSE) methods will be used. The error plots shown in this work will also be the RSE, which in essence will be an absolute value function as only the positive root will be used. Defining Equation (3.13) as the error trajectory, the RMSE is calculated as Equation (3.14), the RSE is calculated as Equation (3.15) and the maximum RSE is calculated as Equation (3.16).

$$\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\} \tag{3.13}$$

$$\mathbf{e}_{\text{RMSE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \mathbf{e}_i^2} \tag{3.14}$$

$$\mathbf{E}_{\text{RSE}} = \left\{\sqrt{(\mathbf{e}_1^2)}, \sqrt{(\mathbf{e}_2^2)}, \ldots, \sqrt{(\mathbf{e}_N^2)}\right\} \tag{3.15}$$

$$\mathbf{e}_{\text{max RSE}} = \max_{\mathbf{e}_i} \mathbf{E}_{\text{RSE}} \tag{3.16}$$

3.3 Odometry

Odometry can be defined as the estimation of the pose of a robotic system relative to the systems initial pose using data from available motion sensors [1]. Some of these sensors are Inertial Mesurement Units (IMUs), wheel encoders, lidars, cameras and others. IMUs measure linear acceleration and rotational rate, which are subject to certain errors, primarily biases and slow drift over time in the biases. In addition, factors like temperature change and vibrations will also affect the measurements [67, pp. 23–25]. IMUs are still very useful in odometry as these errors can be modelled and estimated. Wheel encoders have been very common for ground-based systems, though odometry methods based on this sensor are prone to errors, primarily due to wheel slippage in slippery or uneven terrain, which is difficult to estimate [22, 49]. Furthermore, this sensor cannot be utilised on ASVs as it is constrained to systems with wheels.

Lidars can be used in laser odometry, and camera sensors are used for Visual Odometry (VO), which are generally more accurate than the previously discussed sensors. All these sensors are affected by noise to some degree, which over time leads to drift in the estimated pose. Solving this is an integral part of SLAM, where long-term tracking is performed with loop closure. This is further discussed in Section 3.4.

3.4 Simultaneous Localisation and Mapping

SLAM has been an interesting and popular field of research for some time, and has become an even more popular research field over the last decades, as autonomous systems have become increasingly viable for general usage, see Bimbraw [4]. For a robotic system to operate truly autonomously, knowledge of the local environment and the system state can be very important. Relying exclusively on external positioning systems like GNSS to get the system state will make the system less autonomous and secure. This is primarily due to the fact that GNSS systems are not always viable, and generally do not work indoors, under water or if the GNSS communication is otherwise obstructed or jammed. Another problematic situation could be that the GNSS does not implicitly include information on the local surroundings of the robotic system, while the construction of a map is an essential part of SLAM, and therefore could be utilised for navigation as well.

SLAM can be defined as the combined creation of a map of the surroundings of a robotic systems and the localisation of this system within this map [8]. To allow the systems to operate autonomously, SLAM methods generally only utilise on-board sensors to estimate the robotic system's pose relative to the map. As the localisation side of SLAM often includes the estimation of other useful states such as velocity or biases, calling it *state estimation* may be more correct. Therefore, the term *localisation* should be used when only discussing pose estimation, and *state estimation* for more general situations. The SLAM map is a model of the robotic systems surroundings based on the data gathered by the available sensors. The map is usually considered either unknown, and has to be created from scratch, or has to build based on an initial map, which is either incomplete or uncertain. The mapping side of SLAM therefore has to be able to both initialise and update the map.

For a method to be considered a SLAM method, generally both the state estimation and mapping must be present. Methods where an accurate and reliable GNSS system can be used would be simplified to a mapping problem, as the need for state estimation can be ignored. In surroundings with a known map, the methods can be simplified to a localisation problem. Even though SLAM methods solve both these problems, different SLAM methods are designed with different situations and goals in mind, and therefore different methods often

focus more on either the localisation side or the mapping side of SLAM.

Generally, SLAM methods are divided into two subcategories, *full* SLAM and *online* SLAM. Using $\mathbf{x}$ as the state of the robotic system, $\mathbf{x}[1\colon k]$ as all states up to time k, $\mathbf{z}[1\colon k]$ as the combined system measurements and actuations, and $\mathbf{m}$ as the map. Then full SLAM can be defined as Equation (3.17) and online SLAM can be defined as Equation (3.18). Full SLAM will estimate both the map and all states $\mathbf{x}[1\colon k]$, that is the full trajectory through the map. Online SLAM aims to only estimate the current state $\mathbf{x}[k]$, and is usually estimated by recursively updating the state for each iteration. This has been the standard solution for using SLAM in real time, as full SLAM methods have been too computationally complex to work in real time systems. However, methods utilising the sparsity inherit to SLAM systems now allow full SLAM methods to be usable in real time as well.

$$p(\mathbf{x}[1\colon k], \mathbf{m} \mid \mathbf{z}[1\colon k]) \tag{3.17}$$

$$p(\mathbf{x}[k], \mathbf{m} \mid \mathbf{z}[1\colon k]) = \int \dots \int p(\mathbf{x}[1\colon k], \mathbf{m} \mid \mathbf{z}[1\colon k]) d\mathbf{x}[1] \dots d\mathbf{x}[k-1] \tag{3.18}$$

SLAM methods are usually divided into two separate components, the front-end and the back-end. The front-end is responsible for using raw sensor data for feature detection and data association, where both short-term and long-term data association is performed. Short-term data association refers to feature tracking, and long-term data association refers to the detection of loop closure. The back-end performs the state and map estimation using data from the front-end, and potentially from other sensors. The results from the back-end optimisation can then be sent back to the front-end for the data association. One important reason why the distinction between front-end and back-end exist, is that they are easily parallelised. This is particularly important as the front-end usually is a lot faster than the back-end, and needs to handle system measurements in real time, while the optimisation in the back-end usually takes more computation time. See Figure 3.3 for an illustration of this structure.

The long-term data association performed in the front-end is loop closure, and is the primary defining feature differentiating SLAM from odometry methods like VO. The idea of loop closure is to correct the small errors built up in the odometry method by recognising previously visited areas and applying this newly discovered constraint to the map and state. The errors which may have been built up between leaving and revisiting this area are recalculated. This means that a more accurate topological map of the explored environment can be constructed compared to the results of an odometry-based method, as illustrated in Figure 3.4.

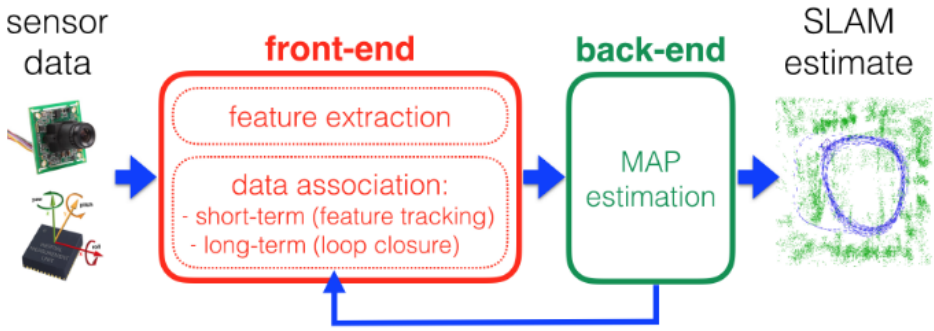


Figure 3.3: Illustration of a generic SLAM method divided into front-end and back-end. Image source [8].

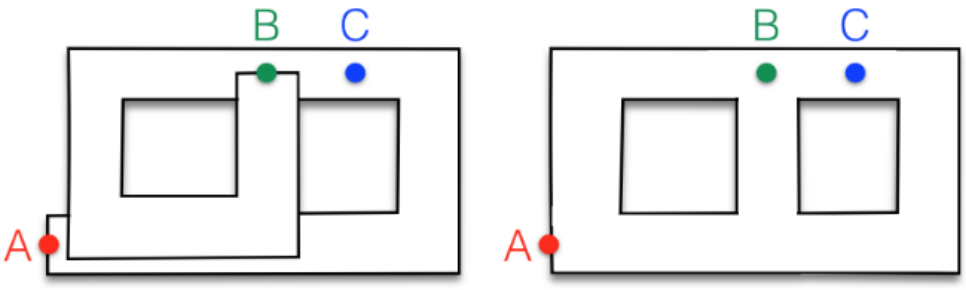


Figure 3.4: Illustration of map constructed by odometry to the left and map constructed by SLAM to the right. The left map is unable to recognise previously visited areas, and therefore drift in measurements may results in the offset illustrated near both areas A and B. The right map is able to recognise previously visited areas with loop closure, resulting in the topology of the surroundings being more accurately mapped. Image source [8].

3.5 Camera Geometry

Some details regarding camera geometry and mathematics will be described here in this section, and is primarily based on Hartley and Zisserman [32] and Corke [13]. A camera model can be defined as the projection function Equation (3.19), where the projection function $\pi : \mathbb{R}^3 \rightarrow \mathcal{I}$ projects a 3D point to a valid pixel, where $\mathcal{I} \subset \mathbb{R}^2$ is the set of all valid pixels, $\mathbf{x}_c$ is a 3D point defined in the camera coordinate system and $\mathbf{u} \in \mathcal{I}$ is a pixel.

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \pi(\mathbf{x}_c) \tag{3.19}$$

The inverse function of projection is called *backprojection*, and is defined as Equation (3.20). Using the pixel $\mathbf{u}$ and corresponding depth d, the original 3D point can be calculated.

$$\mathbf{x}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \pi^{-1}(\mathbf{u}, d) \tag{3.20}$$

All projections in this thesis will be done with the pinhole model, which is a simple and useful model. Note that homogeneous coordinates, which were introduced in Section 2.1.3, are used here, and that to obtain the actual pixel value, the two first elements of $\tilde{\mathbf{u}}$ must be divided by the third. The pinhole model can be defined using Equation (3.21).

$\mathbf{K}$, defined by Equation (3.22), is called the *camera calibration matrix*, and contains the internal camera parameters. These are individual for each camera, and must be estimated using a method such as the one implemented in MATLAB and described by Corke [13, pp. 335–336]. $\alpha_{u,v}$ denotes the focal length of the camera in terms of pixels in the u and v direction. u_0 and v_0 denotes the principal point in the u and v directions, and s is the skew parameters, and is primarily added for completeness as this parameter is often zero.

$$\tilde{\mathbf{u}} = \mathbf{K}\mathbf{P}\tilde{\mathbf{x}}_c \tag{3.21}$$

$$\mathbf{K} = \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.22}$$

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.23}$$

3.6 Fiducial Markers

A fiducial marker is an object placed within the field of view of a camera system which gives additional context to images produced. One simple example could be placing a ruler next to something you're taking a picture of, such that one can find the size of the object using only the information from the ruler in the picture. This can further be extended to introduce more context to images. By introducing a fiducial marker at a *known location*, the camera system can add positional context to the scene.

Some of the initial usages of fiducial markers were as aids to Augmented Reality (AR) systems. Generally, the purpose of AR systems is to use visual computing techniques to introduce artificial objects into a real scene, as captured by e.g. a camera, often in real time. Kato and Billinghurst [42] describes an AR system designed conferencing system, allowing a user to interact with a predefined white board, which is shared with the other users attending the

conference. As the white board acts as a fiducial marker within the conference context, it is easier to share the board with the rest of the conference in an interactive manner for the other attendees. Rekimoto [55] describes an AR system which, aided by 2D matrix codes, allows for accurate estimates of positions of real objects within the AR system. These systems both utilise fiducial markers to improve the interactivity between the two AR systems and the users.

3.6.1 ArUco

Multiple fiducial markers have been developed and used for different purposes. One interesting type is the ArUco, or UcoTag, which was developed as part of the ArUco project by Garrido-Jurado et al. [26]. This fiducial marker was generated by mixed integer linear programming to maximise the diversity of the markers, see Garrido-Jurado et al. [27]. The detector algorithm for these markers, see Romero-Ramirez et al. [57], is developed to be both fast and accurate at detecting the fiducial markers, and is also capable of detecting a number of other fiducial marker types, making it a versatile detector to base a project upon. This marker also has a history of being used in SLAM systems, as it was included in the UcoSLAM method by Muñoz-Salinas and Medina-Carnicer [46]. UcoSLAM worked by fusing natural keypoint features with the artificial markers placed around the system operating area, and aimed to solve a number of problems with monocular visual SLAM methods.

3.6.2 AprilTag

AprilTag is a specialised fiducial marker designed by Olson [51] and improved further by Wang and Olson [68]. These markers work similarly to a QR code, because they are easily distinguishable from the natural features in the surrounding environment and are also able to encode data. Despite having some similar properties, the QR codes and AprilTags are designed with different purposes in mind. The AprilTag software can be found on the AprilRobotics github page[1], see Olson [50].

QR codes usually encode a URL, so that one can use the camera on your phone to read the QR code and get the encoded URL. As QR codes are designed to be utilised by people, a larger amount of data can be encoded. This is because the process is simplified by only decoding one QR code at a time, and the camera must be placed within a specific view of the QR code before it can be successfully decoded. AprilTags are designed as visual aids to autonomous systems, and are therefore designed with this purpose in mind. Multiple AprilTags can be within the same image, and while the camera does have to be within certain views of the markers, these restrictions are less severe than those for QR codes. The coding system used is a modified lexicode which generates a set

[1] See `https://github.com/AprilRobotics`

of binary codewords, where each codeword can be interpreted as the physical AprilTag marker, see Figure 3.5 as an example.

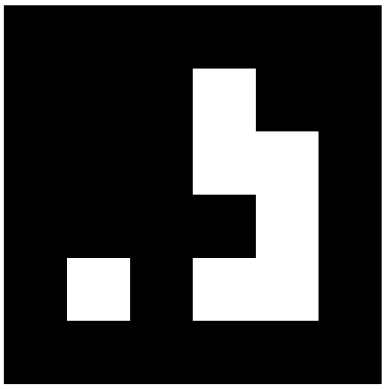


Figure 3.5: AprilTag family 16h5, marker with id 0.

The AprilTag Coding System

As the lexicode system can be parametrised by the number of bits, n, of the codeword and the minimum Hamming distance, d, between each codeword, AprilTags are divided into several distinct instances, referred to as *families*. A useful property of the lexicode is that it can detect $d/2$ bit errors and correct up to $\lfloor (d-1)/2 \rfloor$ bit errors. The marker illustrated in Figure 3.5 is a part of the 16h5 family, which indicates that each codeword is defined by 16 bits, where each codeword in the family has a minimum Hamming distance of 5. A codeword c in a family of bit size n and hamming distance d can be defined as Equation (3.24). The Hamming distance, given two codewords ${}_0c^{n,d}$ and ${}_1c^{n,d}$ can be defined as Equation (3.26) and gives a measure of the difference between the codewords. This means that similar codewords have a small Hamming distance, like 0000 and 0001 have a Hamming distance of 1. Because the codeword size is defined as 16 bits, this family would originally be defined as the binary numbers $[0, 2^{16}-1]$, but because it is defined with a minimum Hamming distance, a large number of these codewords are not allowed in the family because they are too similar. The families are further reduced by also removing the codewords which, if interpreted as a marker and rotated (90, 180 and 270 degrees), also fall below the minimum Hamming distance relative to the rest of the family. The final guarantee for each family is that they are geometrically complex enough to decrease the chance that the pattern may be one which can occur naturally.

$$c^{n,d} = \{c_0^n, c_1^n, \ldots, c_{d-1}^n\}, \tag{3.24}$$

$$\text{where } c_i^n \in \{0, 1\} \tag{3.25}$$

$$h({}_0c^{n,d}, {}_1c^{n,d}) = \sum_{i=0}^{d-1} h_i({}_0c_i^n, {}_1c_i^n), \tag{3.26}$$

$$\text{where } h_i({}_0c_i^n, {}_1c_i^n) = \begin{cases} 0, & \text{if } {}_0c_i^n = {}_1c_i^n \\ 1, & \text{otherwise} \end{cases} \tag{3.27}$$

The families can have vastly different design, and some of the different types can be seen in Figure 3.6. Several different families which have already proved useful in different situations have been generated by the April robotics lab, and are released on their github page[2]. While these standard families are implemented in the AprilTag source code, one may generate new families based on different properties and add the new families to the source code[3]. This makes the AprilTag software very flexible for handling specific situations where particular properties are useful. On the AprilTag user guide page[4], some of the interesting properties of the standard families are noted, and are also listed here. The examples listed can be seen in Figure 3.6.

1. The tag52h13 contains nearly 50000 different patterns, and is the go-to family where large numbers of markers are necessary.
2. The circular AprilTag like tagCircle49h12 and tagCircle21h7 maximise the use of space on circular circular objects.
3. The tagCustom48h12 family includes an unused square within itself, allowing a smaller marker to be placed within recursively. This allows the detector to e.g. observe one marker at a great distance and one close up.
4. The Tag36h11 family is compatible with the ArUcO detector.

[2]See `https://github.com/AprilRobotics/apriltag-imgs`

[3]See `https://github.com/AprilRobotics/apriltag-generation`

[4]See `https://github.com/AprilRobotics/apriltag/wiki/AprilTag-User-Guide`

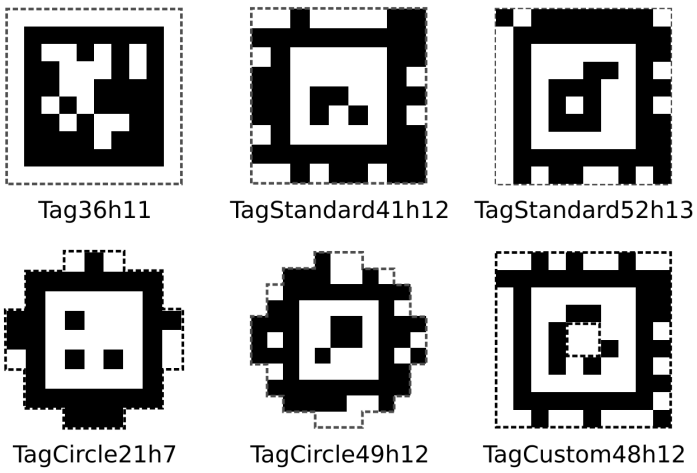


Figure 3.6: Multiple AprilTag families, with the family name written below each. Image source [50].

The AprilTag Detector

The AprilTag detector attempts to find all possible AprilTags of a specified family in a given gray-scale image. The method implemented by Olson [51] will be described shortly here. Initially, the algorithm deals with the detection of line segments. The gradient direction and magnitude are computed for all pixels, and then clustered using a method similar to the graph-based method described by Felzenszwalb and Huttenlocher [21]. This part of the detection algorithm is both the slowest and the most susceptible to noise. The detector therefore relies heavily on the coding system described in Section 3.6.2 to ensure a low false detection rate. This part of the algorithm results in a set of directed line segments.

The next part consists of using the directed line segments to detect four-sided shapes, called quads. This is done using a depth-first search with a depth of four. Initially, all line segments are considered. Then, all lines which are both *close enough* to the previous line and follow a counter-clockwise order, are considered. This pattern continues until a depth of four is reached and a potential quad is detected.

Then the 3×3 matrix which projects 2D homogeneous points from the AprilTag coordinate system to the 2D image coordinate system, called the homography matrix, is estimated using the Direct Linear Transform (DLT) algorithm as described by Hartley and Zisserman [32, pp. 88–93]. This is finally used to read the bits from the detected tag. If a codeword in the AprilTag family matches the read bits, this quad can be approved as a valid detection. This detection contains the following data:

- *id*: The integer id number of the detected AprilTag marker.
- *centre*: The pixel coordinates of the centre of the marker.
- *points*: A list of pixel coordinates, defined in this order: left bottom, right bottom, right top, left top.
- *hamming*: How many error bits were corrected.
- *decision_margin*: The average difference between the intensity of a data bit versus the decision threshold.

3.7 Infinitesimal Plane-Based Pose Estimation

The problem of estimating a 3D pose from a set of 2D points is called Perspective-n-Point (PnP), see Szeliski [65, pp. 669–670]. Several different methods for solving this problem have been developed based on the structure of the problem and the number of points used. We get four detected 2D points from a fiducial marker, and we know also something of the structure of these points. We know that they lay on a flat surface, and the problem can be solved using the Infinitesimal Plane-based Pose Estimation (IPPE) method, as described by Collins and Bartoli [11]. The method estimates the pose by using the transform about an infinitesimally small region on the surface of the marker, which is what the name of the method is derived from.

As we also know the distance between the corners, we can add scale to the pose estimate and thus estimate the relative pose between the camera frame and the fiducial frame. The fiducial frame is defined in the center of the marker with the axes defined as illustrated in Figure 3.7. There, the camera is placed in the fiducial frame as calculated by the IPPE PnP algorithm.

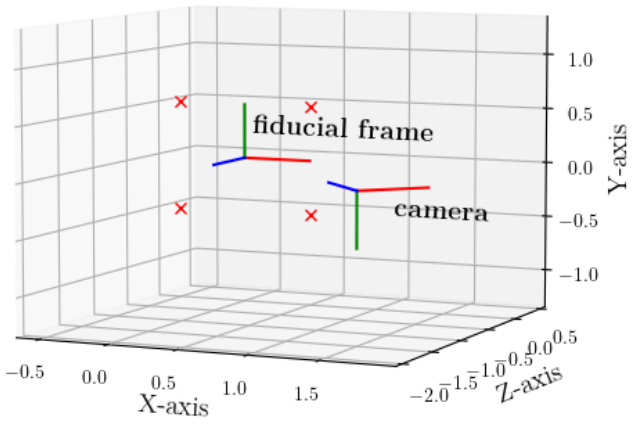


Figure 3.7: Visualisation of the coordinate frame used for solving PnP using the IPPE method. The AprilTag marker is defined by the four red crosses, the detected corners of the marker. The coordinate frame is, when facing the marker, defined with a positive z axis out of the marker towards the observer, x to the right and y up. The IPPE method then calculates the pose of the camera within this frame with a z axis pointing out the front of the camera, y down and x to the right.

3.8 Factor Graphs

Factor graphs are a group of graphical, probabilistic models which provide useful abstractions for modelling many different problems, though for solving SLAM problems it is especially interesting, see Dellaert and Kaess [17]. Other similar structures, like Bayesian networks and Markov random fields are well known in problems of statistical modelling and machine learning, see Yang [69] and Suh [64]. The graphical nature of these models allows an easier way of visualising the designed model, and lends itself well to writing modular systems. A factor graph is defined as a bipartite graph $\mathcal{F} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$, that is a graph defined with two different types of nodes. These are the **factors** $\phi_i \in \mathcal{U}$ and the **variables** $x_j \in \mathcal{V}$, with the edges $e_{ij} \in \mathcal{E}$ between factor nodes and variable nodes. A visual example of a factor graph can be seen in Figure 3.8, where the black nodes represent a factor, the lines represent the edges and the labeled nodes represent the unknown states. The factors can e.g. represent odometry, measurements and priors, while for the variables, x_i represents a system state (like the robot state) and l_i represents a landmark.

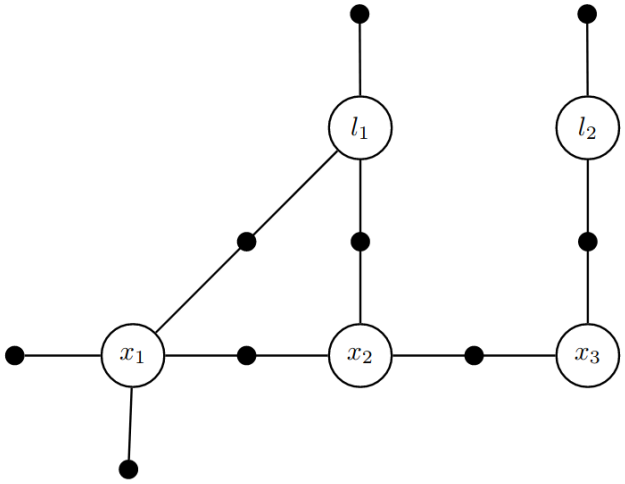


Figure 3.8: Example of a factor graph showing factors, variables and edges. The black dots represent factors and the white circles represent variables. These are the two nodes in the bipartite graph. The lines connecting the factors and variables are the edges. Image taken from [17, p. 12].

Factor graphs are currently the de facto standard for formulating SLAM problems. One of the reasons is because it is easily translated to the front-end / back-end tasks as described in Section 3.4. The front-end then represents the construction of the graph by connecting variables with factors based on sensor measurements. Then the back-end represents the optimisation of the graph and calculates the most likely values of the variables in the graph given the factors. Due to the structure of factor graphs, they are easily factorised and can be evaluated for any given value by evaluating every factor and multiplying the results, given that the measurements are assumed independent. The optimisation problem defined over these factors is a MAP problem, as discussed in Section 3.1.2. This problem can then be solved using nonlinear optimisation methods like Gauss-Newton, Levenberg-Marquardt, Powell's Dogleg, or other nonlinear optimisation methods.

As the graph optimisation is sensor agnostic, factor graphs are well suited to solve sensor fusion problems. This is because it is simple to introduce new sensors to the graph in the form of new variables and factors. These can be added to the original graph without much, if any, alteration in the original graph. As shown by Chiu et al. [10], the use of multi-sensor factor graphs can provide satisfactory navigation solutions using multiple sensors. The expansion of the graph, see Figure 3.9, shows that the underlying structure can remain the same and the primary object may be to estimate the current robot state $\mathbf{x}_4$,

while additional sensors only give more information to help the optimisation method solving the graph.

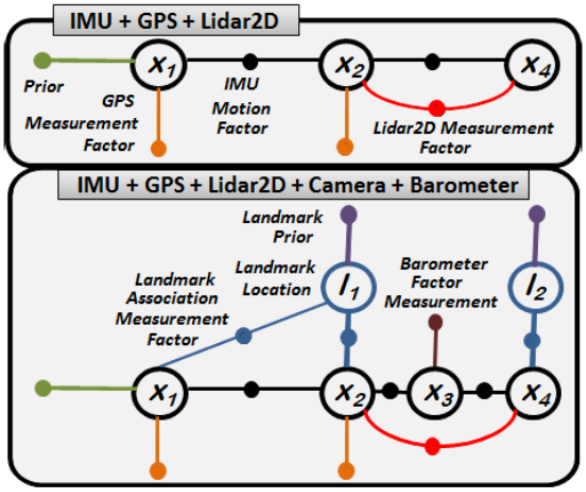


Figure 3.9: Example of multi-sensor factor graphs. The top graph is constructed using IMU, GPS and lidar. The bottom graph is further expanded by adding camera and barometer sensors. Image taken from [10, p. 12].

3.8.1 Factor Graph Implementations

Multiple different libraries for creating and optimising factor graphs have been implemented, and are in many cases open-source. They are often implemented using C++, as the speed and efficiency is essential for using factor graph solvers for online SLAM problems. Exceptions do exist, such as Caesar.jl for the Julia programming language, and bindings to other popular languages like Python and MATLAB are also available for some libraries.

3.8.2 Georgia Tech Smoothing and Mapping

The Georgia Tech Smoothing and Mapping (GTSAM) toolbox is also an open-source C++ implementation for creating and solving factor graphs, developed by Dellaert [18] and Georgia Institute of Technology. As the name suggests, GTSAM implements smoothing and mapping using factor graphs. It is primarily based on Bayesian networks rather than using sparse matrices directly as many SLAM systems have used in the past, and exploits the sparsity in factor graphs for efficiency. State-of-the-art solvers for SLAM problems such as iSAM, see Kaess et al. [37] and iSAM2, see Kaess et al. [39, 40], are implemented in the library. GTSAM is also used as the back-end for many new and efficient SLAM methods, such as TagSLAM, see Pfrommer and Daniilidis [54]. GTSAM

also comes with wrappers for both Python and MATLAB.

3.8.3 SLAM++

The SLAM++ library is an open-source library implementing factor graphs, written in C++ to especially benefit from the incremental nature of online SLAM problems, developed by Ila et al. [35]. It is especially interesting as it differs from several other implementations as it preforms all matrix operations by blocks. This leads to very fast matrix manipulation and arithmetic operations.

3.8.4 g2o

The g2o library is also an open-source C++ library for optimising nonlinear graph-based error functions developed by Kümmerle et al. [43]. The library has been designed to be general and therefore be easily extensible to a wide range of problems. Solutions to several different variants of SLAM problems are also included in the library. It also utilises the sparsity inherent to factor graphs, making it an efficient library. It has a long history of being used in other state-of-the-art SLAM systems, as e.g. ORB-SLAM (ORB-SLAM) by Mur-Artal et al. [47] and LSD-SLAM by Engel et al. [20] use g2o for their back-end. The library has bindings to both Python and .NET, though they are not officially supported.

3.8.5 Caesar

Caesar.jl is an open-source library for factor graph optimisation written in the Julia programming language, see Contributors and Packages [12]. The library focuses primarily on problems related to SLAM, but is written to be very extendable and suitable to many different problems, as it also allows non-Gaussian and multimodal probability distributions to be worked on. This library is very new, and looks to solve many issues which have arisen in previous SLAM solutions like solving under-defined systems, inference with non-Gaussian measurements and more, and because the Julia language itself also seems to be an interesting direction for robotics in the future.

3.9 Lie Theory

A Lie group is defined as a smooth manifold which satisfies the group axioms, which can be considered as smooth topological surfaces, which are especially useful in robotics because $SE(3)$ and $SO(3)$ are not trivially differentiable, see Solà et al. [62]. This means that for both groups, representation of uncertainty,

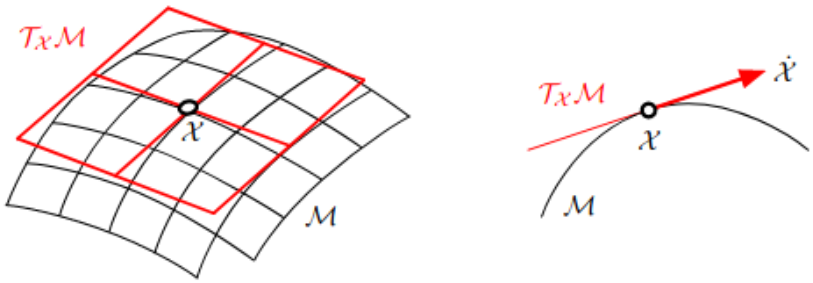


Figure 3.10: A manifold $\mathcal{M}$ and the vector space $\mathcal{T}_{\mathcal{X}}\mathcal{M}$ tangent at the point $\mathcal{X}$, and a convenient side-cut. The velocity element $\dot{\mathcal{X}} = \partial\mathcal{X}/\partial t$, does not belong to the manifold $\mathcal{M}$, but to the tangent space $T_{\mathcal{X}}\mathcal{M}$. Image and caption source [62, p. 2].

incrementing and differentiating states within these groups is more difficult than a state representing only the position of a system, which can be illustrated using Equations (3.28) to (3.33). These operations can however still be performed using Lie algebra because $SO(3)$ and $SE(3)$ are Lie groups.

$$\mathbf{R} \in SO(3) \tag{3.28}$$
$$\delta\mathbf{R} \in \mathbb{R}^{3\times 3} \tag{3.29}$$
$$\mathbf{R} + \delta\mathbf{R} \notin SO(3) \tag{3.30}$$

$$\mathbf{T} \in SE(3) \tag{3.31}$$
$$\delta\mathbf{T} \in \mathbb{T}^{4\times 4} \tag{3.32}$$
$$\mathbf{T} + \delta\mathbf{T} \notin SE(3) \tag{3.33}$$

The concepts of a smooth manifold and a group necessitates some further description. A smooth, or differentiable, manifold is a topological space which locally behaves like a linear space. One may visualise this using Figure 3.10, which illustrates a manifold $\mathcal{M}$ and a tangent space $\mathcal{T}_{\mathcal{X}}\mathcal{M}$ defined at $\mathcal{X}$, though $SE(3)$ and $SO(3)$ obviously inhabit spaces of greater dimensions than can be visualised here. In this tangent space, a state can evolve, and here the perturbations which were impossible in Equations (3.28) to (3.33) can be performed within the tangent space.

A group $(\mathcal{G}, \circ)$ with set $\mathcal{G}$ and a composition operation $\circ$, where $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in \mathcal{G}$, has to satisfy the axioms described by Equations (3.34) to (3.37). Using the definitions Equation (2.12) of $SO(3)$ and Equation (2.17) of $SE(3)$, it can be observed that the axioms listed below hold for both groups.

$$\text{Closure under } \circ : \mathcal{X} \circ \mathcal{Y} \in \mathcal{G} \tag{3.34}$$

$$\text{Identity } \mathcal{E} : \mathcal{E} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{E} = \mathcal{X} \tag{3.35}$$

$$\text{Inverse } \mathcal{X}^{-1} : \mathcal{X}^{-1} \circ \mathcal{X} = \mathcal{E} \tag{3.36}$$

$$\text{Associativity} : (\mathcal{X} \circ \mathcal{Y}) \circ \mathcal{Z} = \mathcal{X} \circ (\mathcal{Y} \circ \mathcal{Z}) \tag{3.37}$$

To allow Lie groups to transform elements for other sets, the group action $\mathcal{X} \cdot v$ can be defined, given a Lie group $\mathcal{M}$ and a set $\mathcal{V}$, of $\mathcal{X} \in \mathcal{M}$ on $v \in \mathcal{V}$ as Equation (3.38). For a group action, $\cdot$, to be valid it must satisfy the axioms defined by Equations (3.39) and (3.40).

$$\cdot : \mathcal{M} \times \mathcal{V} \to \mathcal{V}; (\mathcal{X} \cdot v) \tag{3.38}$$

$$\text{Identity} : \mathcal{E} \cdot v = v \tag{3.39}$$

$$\text{Compatibility} : (\mathcal{X} \circ \mathcal{Y}) \cdot v = \mathcal{X} \cdot (\mathcal{Y} \cdot v) \tag{3.40}$$

The tangent space at the identity of a Lie group $\mathcal{M}$ is called the Lie algebra of the group, and is defined as Equation (3.41). This defines a vector space of elements $\tau^\wedge \in \mathbf{m}$, where $(\cdot)^\wedge$ is called the hat operator and is defined as Equation (3.42), where the $\mathcal{E}_i$ values are called the *generators* of $\mathbf{m}$. This essentially maps a matrix to a vector. The inverse operator $(\cdot)^\vee$ is called the vee operator and is defined as Equation (3.43), where the $\mathbf{e}_i$ values are the vectors of the base of $\mathbb{R}^m$, that is $\mathbf{e}_i = E_i^\wedge$. As it defines the inverse to the hat operator, this essentially maps a vector to a matrix.

$$\mathbf{m} \triangleq T_{\mathcal{E}}\mathcal{M} \tag{3.41}$$

$$\text{Hat} : \mathbb{R}^m \to \mathbf{m}; \tau \to \tau^\wedge = \sum_{i=0}^{m} \tau_i E_i \tag{3.42}$$

$$\text{Vee} : \mathbf{m} \to \mathbb{R}^m; \tau^\wedge \to (\tau^\wedge)^\vee = \tau = \sum_{i=0}^{m} \tau_i \mathbf{e}_i \tag{3.43}$$

Transforming elements of the Lie algebra to elements of the manifold, the exponential map is used. The exponential map is defined using Equation (3.44). The inverse transform is done with the logarithmic map, which is defined using Equation (3.45). As it is often more convenient, vectorised versions of the exponential and logarithmic maps are also listed in Equations (3.46) and (3.47). Using the Exp and Log operations, perturbations on the manifold can now be represented as tangent space vectors, and then transformed back on the manifold.

$$\exp : \mathbf{m} \to \mathcal{M};\ \mathcal{X} = \exp(\tau^{\wedge}) \tag{3.44}$$
$$\log : \mathcal{M} \to \mathbf{m};\ \tau^{\wedge} = \log(\mathcal{X}) \tag{3.45}$$
$$\mathrm{Exp} : \mathbb{R}^m \to \mathcal{M};\ \mathcal{X} = \mathrm{Exp}(\tau) = \exp(\tau^{\wedge}) \tag{3.46}$$
$$\mathrm{Log} : \mathcal{M} \to \mathbb{R}^m;\ \tau = \mathrm{Log}(\mathcal{X}) = \log(\mathcal{X})^{\vee} \tag{3.47}$$

3.9.1 Differentiation on Lie Groups

Using the above theory, differentiation on Lie groups can be defined. For simplicity, some useful results are noted here from Solà et al. [62] and Haavardsholm [30]. The chain rule can also be defined for manifolds:

$$\mathcal{Z} = g(\mathcal{Y}) = g(f(\mathcal{X})) \tag{3.48}$$
$$\frac{\partial \mathcal{Z}}{\partial \mathcal{X}} = \frac{\partial g(\mathcal{Y})}{\partial \mathcal{Y}} \frac{\partial f(\mathcal{X})}{\partial \mathcal{X}} \tag{3.49}$$

The cross product matrix can be defined as:

$$[\mathbf{t}]_{\times} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ t_2 & t_1 & 0 \end{bmatrix}, \mathbf{t} \in \mathbb{R}^3 \tag{3.50}$$

The following will be special cases for the $SE(3)$ group. The Adjoint is defined as:

$$\mathbf{Ad}(\mathbf{X}) = \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_{\times}\mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{6\times 6} \tag{3.51}$$

The derivative of the inverse function:

$$f(\mathbf{X}) = \mathbf{X}^{-1} \tag{3.52}$$
$$\frac{\partial f(\mathbf{X})}{\partial \mathbf{X}} = -\mathbf{Ad}_{\mathbf{X}} \tag{3.53}$$

The derivative for composition:

$$f(\mathbf{X}, \mathbf{Y}) = \mathbf{XY} \tag{3.54}$$
$$\frac{\partial f(\mathbf{X}, \mathbf{Y})}{\partial \mathbf{X}} = \mathbf{Ad}_{\mathbf{Y}}^{-1} \tag{3.55}$$
$$\frac{\partial f(\mathbf{X}, \mathbf{Y})}{\partial \mathbf{Y}} = \mathbf{I}_{6\times 6} \tag{3.56}$$

To simplify the equation, the function $\mathbf{Rot}(\mathbf{X}) : SE(3) \rightarrow SO(3)$ extracts the rotational part of the pose. The derivative for group action:

$$f(\mathbf{X}, \mathbf{y}) = \mathbf{X} \cdot \mathbf{y},\ \mathbf{X} \in SE(3), \mathbf{y} \in \mathbb{R}^3 \tag{3.57}$$

$$\frac{\partial f(\mathbf{X}, \mathbf{t})}{\partial \mathbf{X}} = \begin{bmatrix} \mathbf{Rot}(\mathbf{X}) & -\mathbf{Rot}(\mathbf{X})[\mathbf{y}]_\times \end{bmatrix} \tag{3.58}$$

$$\frac{\partial f(\mathbf{X}, \mathbf{t})}{\partial \mathbf{y}} = \mathbf{Rot}(\mathbf{X}) \tag{3.59}$$

3.10 Incremental Smoothing And Mapping

The iSAM algorithm, as described by Kaess et al. [37], is an approach to solving SLAM based on fast incremental matrix factorisation. In the iSAM algorithm, the complete problem is solved for each batch, which can lead to unnecessary calculations being performed because local changes to the problem usually don't affect far-away parts of the system. iSAM2 is a new version of this algorithm, where an important distinction is that iSAM2 does not use the batch relinearisation presented by Kaess et al. [37] and uses the Bayes tree data structure. iSAM2, as described by Kaess et al. [39, 40], is then an incremental, graph-based algorithm, written for solving SLAM problems by utilising the advantages of the sparse nature of SLAM problems defined by factor graphs. The algorithm uses the Bayes tree data structure, incremental reordering and fluid relinearisation, which eliminates the need for the batch relinearisation performed in the iSAM algorithm.

3.10.1 The Bayes Tree

The Bayes tree data structure was originally developed by Kaess et al. [38]. It is very similar to clique trees, see Blair and Peyton [5], though is directed and similar to the Bayes net, see Pearl [53], which encodes factored probability densities. This data structure provides a better understanding of batch matrix factorisation for probability densities. Updates to matrix factorisation can be done by editing the Bayes tree and combined with iSAM2, this provides an efficient sparse, nonlinear incremental optimiser. Factor graphs can be converted into chordal Bayes nets by using variable elimination, see Figure 3.12 for a visualisation of this process. The resulting Bayes net can be converted to a Bayes tree by discovering the cliques. The Bayes tree for the graph seen in Figure 3.11 is visualised in Figure 3.12.

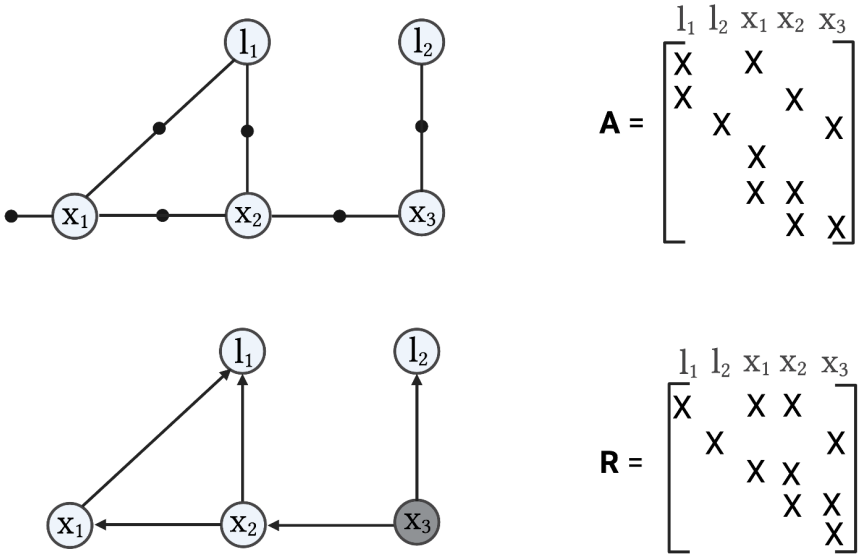


Figure 3.11: Top: The factor graph and associated Jacobian matrix **A** for a small SLAM example. The robot is located at successive poses x_1, x_2, and x_3, and observes the landmarks l_1 and l_2. x_1 has an absolute measurement as well. Bottom: The chordal Bayes net and the associated square root information matrix **R** resulting from eliminating the factor graph with the elimination ordering l_1, l_2, x_1, x_2, and x_3. The root of the net is the last variable to be eliminated, x_3, and is therefore shaded darker than the other nodes. Image and caption taken from [38].

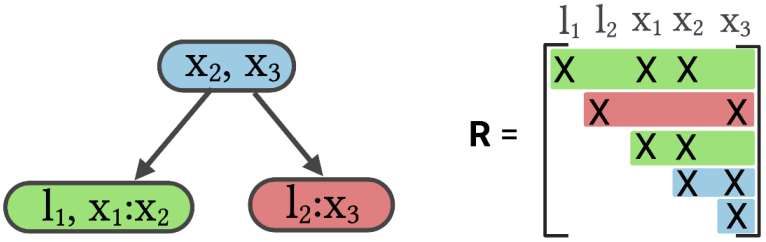


Figure 3.12: The Bayes tree and associated square root information matrix **R** describing the clique structure in the Bayes net from Figure 3.11. A Bayes tree is similar to a clique tree, but captures the formal equivalences between sparse linear algebra and inferences in graphical models better. Image and caption taken from [38].

3.10.2 Incremental Inference

Incremental inference, such as adding new measurements, can be done with a simple edit of the Bayes tree. This is done by reinterpreting the top of the Bayes tree as a factor graph, adding this to the new factors and performing the elimination again. This can then be reattached to the subtrees, which will

remain unaffected. An example of this process is visualised in Figure 3.13.

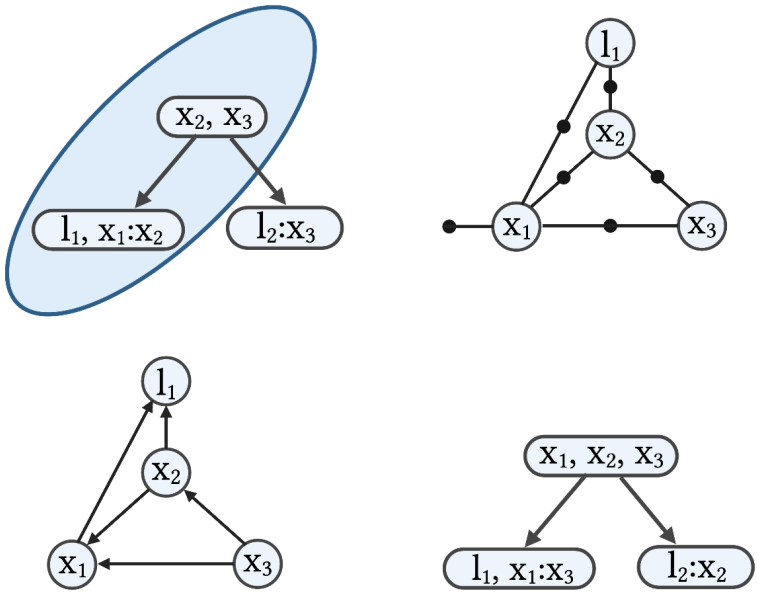


Figure 3.13: An update has been added to the Bayes tree from Figure 3.12. Top right: A factor connecting x_1 and x_3 as been added. Top left: The affected parts of the tree have been highlighted. Bottom left: The factor graph generated for the relevant part of the tree. Bottom right: The resulting Bayes tree, created from the chordal Bayes net, with the unmodified right subtree from the original tree. Image and caption isnpired by [38], and created with BioRender.com.

3.10.3 Incremental Reordering

When eliminating variables as described in Section 3.10.1, the ordering is essential for efficiency. An optimal ordering will minimise the fill-in, that is the additional entries in the square root information matrix created during the elimination process. Fill-in in Bayes trees translate to larger clique sizes, which increase computation time, and unless the Bayes net already is chordal, fill-in cannot be avoided. As finding the optimal ordering is an NP-hard problem, the Column Approximate Minimum Degree (COLAMD) algorithm by Davis et al. [15] can provide near-optimal orderings. This algorithm was used for the original iSAM. However, it was found by Kaess et al. [38] that the variables could be reordered at every incremental update. Then the reordering could be performed only for the affected variables, and a constrained implementation of the COLAMD algorithm was used.

3.10.4 Fluid Relinearisation

With fluid relinearisation, the validity of the linearisation point for each variable is kept track of, and is only relinearised when necessary. When relinearising a variable, the information relevant to this variable must be removed from the Bayes tree, and is replaced by relinearising the corresponding nonlinear factors. The relinearised cliques also have to take into account the marginal factors from the subtrees, which must be passed up the tree. These factors can also be cached, so that these results can be reused and computation time can be saved.

3.11 Preintegration

The purpose of preintegration is to combine a larger number of IMU measurements between two poses into a single odometry constraint, as initially proposed by Lupton and Sukkarieh [44]. Because IMUs produce data at a much greater rate than other sensors, it is not feasible to add the individual measurements to the optimisation problem. Instead, the preintegration method *integrates* the IMU measurements between two time steps, producing one odometry constraint. This was continued by Forster et al. [23], where preintegration was extended to the $SO(3)$ group and defined as a factor graph problem.

An IMU consists of an accelerometer and a gyroscope, which generate acceleration measurements and rotation rate measurements, with respect to the inertial frame. These measurements are affected by a slowly varying bias $\mathbf{b}$, and additive white noise $\boldsymbol{\eta}$, denoted with the superscript a for the accelerometer and g for the gyroscope. Defining the true acceleration as $\mathbf{a}_W[k] \in \mathbb{R}^3$, the accelerometer measurement as $\mathbf{a}_B^z[k] \in \mathbb{R}^3$, the true rotation rate as $\boldsymbol{\omega}_W[k] \in \mathbb{R}^3$, the gyroscope measurement as $\boldsymbol{\omega}_W^z[k] \in \mathbb{R}^3$, the gravitational acceleration vector as $\mathbf{g}_W$. B is the sensor body frame, and W the world frame. The pose of the sensor body frame in the world frame is denoted as the rotation $\mathbf{R}_W \in SO(3)$ and $\mathbf{t}_W$. The measurement equations are then:

$$\boldsymbol{\omega}_W^z[k] = \boldsymbol{\omega}_W[k] + \mathbf{b}^g[k] + \boldsymbol{\eta}^g[k] \tag{3.60}$$

$$\mathbf{a}_W[k] = \mathbf{R}_W^\top[k](\mathbf{a}_W^z[k] - \mathbf{g}_W) + \mathbf{b}^g[k] + \boldsymbol{\eta}^g[k] \tag{3.61}$$

To estimate the system motion from these measurements, this kinematic model defines the evolution of the pose and velocity:

$$\dot{\mathbf{R}}_W[k] = \mathbf{R}_W[k]\boldsymbol{\omega}_W^\wedge[k] \tag{3.62}$$

$$\dot{\mathbf{v}}_W[k] = \mathbf{a}_W[k] \tag{3.63}$$

$$\dot{\mathbf{t}}_W[k] = \mathbf{v}_W[k] \tag{3.64}$$

Then the state at a time $k+\Delta k$ can be calculated by integration:

$$\mathbf{R}_W[k+\Delta k] = \mathbf{R}_W[k]\mathrm{Exp}\left(\int_k^{k+\Delta k} \boldsymbol{\omega}_W(\tau)d\tau\right) \tag{3.65}$$

$$\mathbf{v}_W[k+\Delta k] = \mathbf{v}_W[k] + \int_k^{k+\Delta k} \mathbf{a}_W(\tau)d\tau \tag{3.66}$$

$$\mathbf{t}_W[k+\Delta k] = \mathbf{t}_W[k] + \int_k^{k+\Delta k} \mathbf{v}_W(\tau)d\tau + \int\int_k^{k+\Delta k} \mathbf{a}_W(\tau)d\tau^2 \tag{3.67}$$

Assuming $\mathbf{a}_W$ and $\boldsymbol{\omega}_W$ constant for the time interval $[k\colon k+\Delta k]$, the integration above is trivial. Then inserting for Equations (3.60) and (3.61):

$$\mathbf{R}_W[k+\Delta k] = \mathbf{R}_W[k]\mathrm{Exp}\left((\boldsymbol{\omega}_W^z[k] - \mathbf{b}^g[k] - \boldsymbol{\eta}^g[k])\Delta k\right) \tag{3.68}$$

$$\mathbf{v}_W[k+\Delta k] = \mathbf{v}_W[k] + \mathbf{g}_W\Delta k + \mathbf{R}_W[k](\mathbf{a}_W^z - \mathbf{b}^g[k] - \boldsymbol{\eta}^g[k])\Delta k \tag{3.69}$$

$$\begin{aligned}\mathbf{t}_W[k+\Delta k] = \mathbf{t}_W[k] + \mathbf{v}_W[k]\Delta k + \frac{1}{2}\mathbf{g}_W\Delta k^2 \\ + \frac{1}{2}\mathbf{R}_W[k](\mathbf{a}_W^z - \mathbf{b}^g[k] - \boldsymbol{\eta}^g[k])\Delta k^2\end{aligned} \tag{3.70}$$

Using these equations, the full motion combining IMU measurements on the time range $[\colon j]$ can be calculated as:

$$\mathbf{R}_W[j] = \mathbf{R}_W[i]\prod_{k=i}^{j-1}\mathrm{Exp}\left((\boldsymbol{\omega}_W^z[k] - \mathbf{b}^g[k] - \boldsymbol{\eta}^g[k])\Delta k\right) \tag{3.71}$$

$$\mathbf{v}_W[j] = \mathbf{v}_W[i] + \sum_{k=i}^{j-1}\left[\mathbf{g}_W\Delta k + \mathbf{R}_W[k](\mathbf{a}_W^z - \mathbf{b}^g[k] - \boldsymbol{\eta}^g[k])\Delta k\right] \tag{3.72}$$

$$\begin{aligned}\mathbf{t}_W[j] = \mathbf{t}_W[i] + \sum_{k=i}^{j-1}(\mathbf{v}_W[k]\Delta k + \frac{1}{2}\mathbf{g}_W\Delta k^2 \\ + \frac{1}{2}\mathbf{R}_W[k](\mathbf{a}_W^z - \mathbf{b}^g[k] - \boldsymbol{\eta}^g[k])\Delta k^2)\end{aligned} \tag{3.73}$$

This can also be defined without the initial state, so that only the relative motion is calculated, which is the goal of preintegration. Note that this implies that the biases, $\mathbf{b}^a$ and $\mathbf{b}^g$, are constant in the time range $[i\colon j]$.

Chapter 4

Fiducial SLAM

The primary contribution of this thesis was a factor graph-based framework implementing an fiducial marker based full SLAM method for use in a canal environment by the autonomous ferry milliAmpere. It is important to create a factor graph-based framework for the fiducial SLAM method, as it could easily be extended in the future to a multi-sensor SLAM system using factors such as VO, lidar-based laser odometry and GNSS. As discussed in Section 3.8, a system utilising its full sensor capabilities is generally the most beneficial for the system, and factor graphs are a simple way of integrating multiple sensors. For milliAmpere to have the best idea of both its own position and knowledge of its surroundings, a tailored SLAM method utilising the full sensor capabilities of the ferry would be safe, as sensor measurements can simply not be added to the graph if the sensor is malfunctioning. Furthermore, unreliable measurements can be modelled with high noise and reliable measurements with low noise, which will weigh the corresponding measurements in during the optimisation.

4.1 System Overview

The general structure of the method will be discussed in this section, with focus on how the different parts communicate. The Fiducial Map was loaded from a predefined set of fiducial markers, and defined the current map of the individual fiducial markers and their location in the world coordinates. For each camera used, an instance of the Independent Camera Thread was executed separately. All detections produced by all cameras were combined after this, and were sent to the Graph Creation, where one Fiducial factor was created for each detection. These factors were inserted into a subgraph, connecting the current system pose to all detections for all cameras. Finally, the subgraph was inserted into the iSAM2 back-end. Any other sensor measurements were added to the optimiser. At a fixed rate, the back-end optimisation algorithm was executed and the result was sent to the Fiducial Map to update it. This structure is visualised in Figure 4.1.

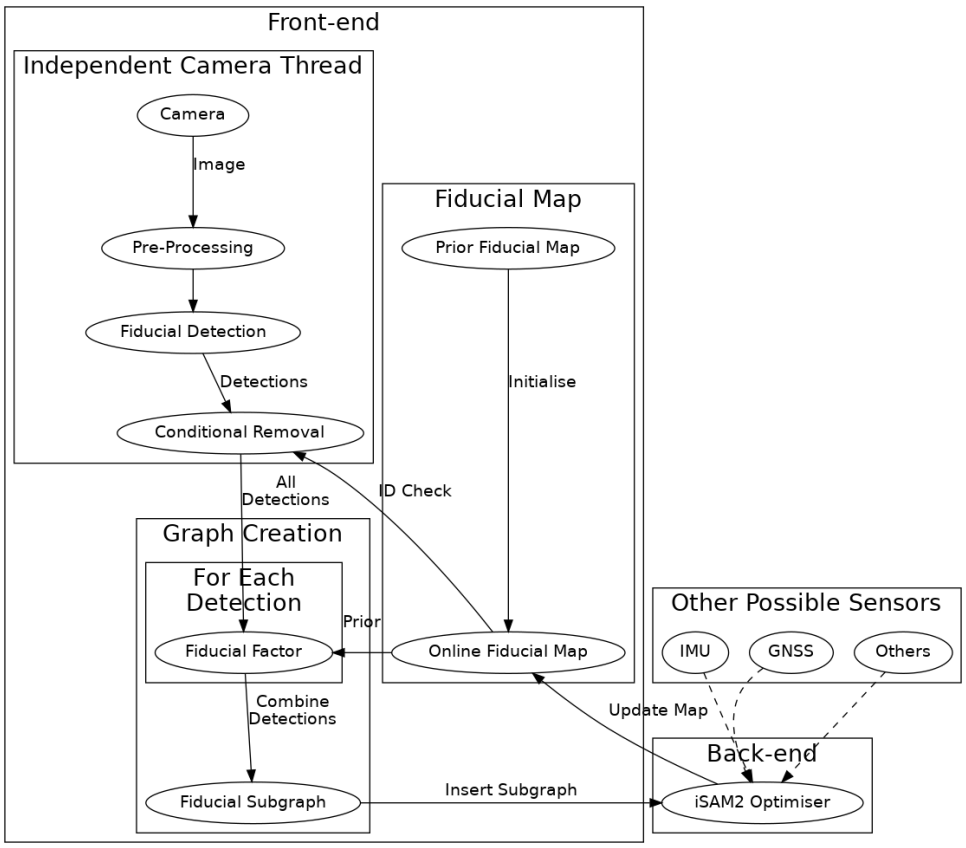


Figure 4.1: The full Fiducial SLAM system, divided into the Fiducial SLAM front-end, iSAM2 back-end optimiser, and some other possible front-end nodes. The Fiducial Map is initialised by the prior information of the fiducial marker pose in the world coordinate frame. The Independent Camera Thread processes each image from the camera system in separate threads, and finally merges the detections. A Fiducial Factor is created for each detection, then inserted into a subgraph in the Graph Creation. The subgraph is inserted into the iSAM2 back-end optimiser, together with data from any other potential sensors. The result of the optimisation is used to update the Fiducial Map. Generated using the Graphviz tool.

4.2 Independent Camera Thread

As the name suggest, this part of the algorithm was responsible for processing the images coming from the camera system, and could be done independently of any other camera in a multi-camera system. The pre-processing was primarily to prepare the image for the detection algorithm, by e.g. converting the image to gray scale. Next, the detection algorithm was applied to the image. The primary data fields detected were defined by Equations (4.1) and (4.2), in other words the detected fiducial markers unique ID within the number of

possible fiducial markers, and the four detected pixel corners. Note that there were usually more data fields connected to each detection, though these were not used in this method, and were therefore not noted here.

The resulting detections were passed through a conditional removal step. There, the detected marker ID was checked against the fiducial map, and any detection *not* in the map were discarded. Then, any detection where the detected corners were closer than a set threshold of the image size were removed. Finally, any detection where the corners were outside an ellipsis defined in the centre of the image, with semi-major and -minor axes defined as a specified fraction of the image width and height, were removed.

$$\text{ID} \in \mathcal{F},\ \mathcal{F} = \{x \subset \mathbb{N} | x \leq \text{Num. Fiducial markers}\} \tag{4.1}$$

$$\text{corners} \in \mathcal{I}^4,\ \mathcal{I} = \left\{ \mathbf{i} \subset \mathbb{R}^2 \middle| \mathbf{i} \in \left[\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \text{image width} \\ \text{image height} \end{bmatrix} \right] \right\} \tag{4.2}$$

4.3 Fiducial Map

The fiducial map was a collection of all active fiducial markers, mapping the fiducial ID to prior information about the relevant marker. The prior information stored for each fiducial ID was the side length of the fiducial marker, the pose of the marker in world coordinates with corresponding Gaussian noise and the positions of the fiducial corners in world coordinates with corresponding Gaussian noise.

This map was used to conditionally remove any detection of markers that were not supposed to be there, and was used to give prior and constraining information to the creation of the fiducial factors. These topics were discussed further in Section 4.2 and Section 4.4.1 respectively.

While the map was supposed to already be accurate based on the prior information, the data was regularly updated by the back-end, to ensure that the information was up to date and reflects the estimated uncertainty.

4.4 Graph Creation

The graph creation process was split into two steps, the creation of a fiducial factor for each detected marker and the combination of all these factors as a subgraph for the current system pose.

4.4.1 Fiducial Factor

In this thesis, fiducial markers were introduced as static and dependable landmarks. The fiducial SLAM method that was developed used prior information of the side length of the used fiducial markers and their pose in world coordinates. This could be achieved in multiple ways, and to investigate two different approaches, two separate factors were developed in this thesis. In order to define a new factor for a factor graph, the measurement and error functions must be defined, which was done for each factor. As a result of using factor graphs, the two methods were added to a graph trivially as a subgraph, and were visualised the same way. They also need the same parameters to be defined, the detected corners, the side length of the fiducial marker, the camera calibration parameters and the prior information from the map.

Pose Factor

One method of adding a fiducial marker to a factor graph, was to add it as a pose $\mathbf{L}_{\mathcal{W}} \in SE(3)$ in the world coordinate system $\mathcal{W}$. At the creation of the factor graph, the known fiducial markers were added to the graph as prior pose factors with preferably small Gaussian noise models. This factor structure was illustrated in Figure 4.2a. When the fiducial marker was detected, the IPPE PnP method discussed in Section 3.7 was used on the detected corner points. Scaled by the side length of the marker, this method calculated the transform between the camera frame and the fiducial frame. This provided the measurement $\mathbf{Z}_{\mathcal{C}} \in SE(3)$ in the camera coordinate system $\mathcal{C}$ of the fiducial marker pose. Given a known transform $\mathbf{T}_{\mathcal{B}}^{\mathcal{C}}$ from the body coordinate system $\mathcal{B}$ to the camera coordinate system, and the body pose $\mathbf{X}_{\mathcal{W}}$, a measurement function was defined as:

$$h(\mathbf{X}_{\mathcal{W}}, \mathbf{L}_{\mathcal{W}}, \mathbf{T}_{\mathcal{B}}^{\mathcal{C}}) = (\mathbf{T}_{\mathcal{B}}^{\mathcal{C}} \mathbf{X}_{\mathcal{W}})^{-1} \mathbf{L}_{\mathcal{W}} \tag{4.3}$$

Using the results from Section 3.9.1, the Jacobians of Equation (4.3) were calculated as:

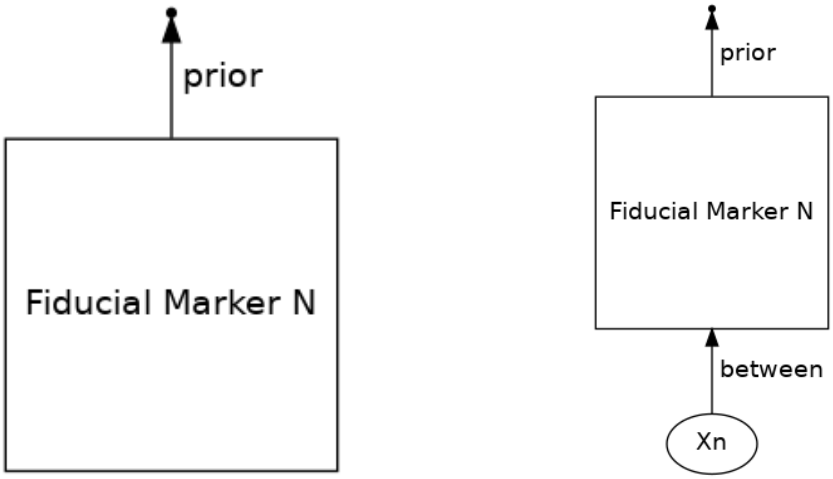
$$\frac{\partial h(\mathbf{X}_{\mathcal{W}}, \mathbf{L}_{\mathcal{W}}, \mathbf{T}_{\mathcal{B}}^{\mathcal{C}})}{\partial \mathbf{X}_{\mathcal{W}}} = -(\mathbf{Ad}((\mathbf{T}_{\mathcal{B}}^{\mathcal{C}} \mathbf{X}_{\mathcal{W}})^{-1}))^{-1} \mathbf{Ad}(\mathbf{T}_{\mathcal{B}}^{\mathcal{C}} \mathbf{X}_{\mathcal{W}}) \tag{4.4}$$

$$\frac{\partial h(\mathbf{X}_{\mathcal{W}}, \mathbf{L}_{\mathcal{W}}, \mathbf{T}_{\mathcal{B}}^{\mathcal{C}})}{\partial \mathbf{L}_{\mathcal{W}}} = \mathbf{I}_{6\times 6} \tag{4.5}$$

The error function was then defined as:

$$e(\mathbf{Z}_{\mathcal{C}}, \mathbf{X}_{\mathcal{W}}, \mathbf{L}_{\mathcal{W}}, \mathbf{T}_{\mathcal{B}}^{\mathcal{C}}) = \mathrm{Log}(h(\mathbf{X}_{\mathcal{W}}, \mathbf{L}_{\mathcal{W}}, \mathbf{T}_{\mathcal{B}}^{\mathcal{C}})^{-1} \mathbf{Z}_{\mathcal{C}}) \tag{4.6}$$

This transform was added to the factor graph connecting the current pose and the marker pose with a predefined Gaussian measurement noise model:



(a) **(b)**

Figure 4.2: Figure 4.2a: A prior pose factor with a specified noise for the fiducial marker. Figure 4.2b: A connection between the detected fiducial marker and a state Xn is added as a factor. Both generated using the Graphviz tool.

$$\mathcal{N}(\mathbf{Z}_{\mathcal{C}}, \mathbf{\Sigma}) \tag{4.7}$$

$$\Sigma \in \mathbb{R}^{6\times 6} \tag{4.8}$$

Where the covariance matrix Σ represents the uncertainty in the estimated pose measurement, and is a tuneable parameter.

Projection Factor

The other method of adding a fiducial marker to a factor graph, was to add the four corners as points in the world coordinate system. This system was slightly more complicated than the pose factor-method, as the four corners must all be added to the factor graph and the physical constraints must be added between these corners. Unlike the method discussed in Section 4.4.1, where the side length and geometric properties of the marker were used in the IPPE PnP method, it was necessary to encode this information in the graph when using the corners individually. This information included constraining the distance between the corners as defined by the side lengths and diagonals of the physical fiducial marker. That is, the side lengths and diagonals of the marker constrained the distance the points can be from each other. Defining the corner points in the world coordinate system as $\mathbf{l}^{N,i} \in \mathbb{R}^3$ and d as the side length, this constraint was defined as the Euclidean norm between the relevant corners:

$$c(\mathbf{l}^{N,i}, \mathbf{l}^{N,j}) = \left\| \mathbf{l}^{N,i} - \mathbf{l}^{N,j} \right\|_2 \tag{4.9}$$

$$\text{constraints} = \begin{cases} c(\mathbf{l}^{N,i}, \mathbf{l}^{N,j}) = \sqrt{2}d & \text{if } i,j \in \{\{0,2\},\{1,3\}\} \\ c(\mathbf{l}^{N,i}, \mathbf{l}^{N,j}) = d & \text{if } i,j \in \{\{0,1\},\{1,2\},\{2,3\},\{3,0\}\} \end{cases} \tag{4.10}$$

This was done using factors with constrained noise models, in other words there was no uncertainty associated with these factors. While this may be technically wrong, as the marker can be deformed slightly e.g. by wind, this captured the assumption that the markers were fixed and static objects in the graph. A prior value was only added to one corner, allowing the entire marker subgraph to be defined relative to this corner. This structure was visualised in Figure 4.3, with Figure 4.3a visualising the internal structure of this factor and Figure 4.3b visualising how the four measured corners were connected to the state. Note that as each corner was a separate variable in the graph, where N was the id and $i \in \{0,1,2,3\}$ was the corner number. These were defined as discussed in Section 3.6.2.

For each detection, the measurement was defined as $\mathbf{Z}^N = \left\{\mathbf{z}^{N,0}, \mathbf{z}^{N,1}, \mathbf{z}^{N,2}, \mathbf{z}^{N,3}\right\}$, where $\mathbf{z}_i^N \in \mathbb{R}^2$, N was the fiducial marker ID and i was the detected corner. The corresponding world points were then defined as $\mathbf{L}^N = \left\{\mathbf{l}^{N,0}, \mathbf{l}^{N,1}, \mathbf{l}^{N,2}, \mathbf{l}^{N,3}\right\}$, where $\mathbf{l}^{N,i} \in \mathbb{R}^3$. Given a known transform $\mathbf{T}_{\mathcal{B}}^{\mathcal{C}}$ from the body coordinate system $\mathcal{B}$ to the camera coordinate system, the body pose $\mathbf{X}_{\mathcal{W}}$, and the projection function defined in Equation (3.19), a measurement function was defined as:

$$h(\mathbf{X}_{\mathcal{W}}, \mathbf{l}_{\mathcal{W}}^{N,i}, \mathbf{T}_{\mathcal{B}}^{\mathcal{C}}) = \pi\left((\mathbf{T}_{\mathcal{B}}^{\mathcal{C}}\mathbf{X}_{\mathcal{W}})^{-1}\mathbf{l}_{\mathcal{W}}^{N,i}\right) \tag{4.11}$$

As the $\pi(\mathbf{p})$ projection function will depend on the camera model used, a generic Jacobian $\frac{\partial \pi(\mathbf{p})}{\partial \mathbf{p}} = \Pi(\mathbf{p})$ was defined. To simplify the calculations, the function $\mathbf{Rot}(\mathbf{X}) : SE(3) \to SO(3)$ extracted the rotational part of the pose. Then, using the results from Section 3.9.1, the following Jacobians were calculated:

$$\frac{\partial h(\mathbf{X}_{\mathcal{W}}, \mathbf{l}_{\mathcal{W}}^{N,i}, \mathbf{T}_{\mathcal{B}}^{\mathcal{C}})}{\partial \mathbf{X}_{\mathcal{W}}} = -\Pi\left((\mathbf{T}_{\mathcal{B}}^{\mathcal{C}}\mathbf{X}_{\mathcal{W}})^{-1}\mathbf{l}_{\mathcal{W}}^{N,i}\right) \begin{bmatrix} \mathbf{Rot}\left((\mathbf{T}_{\mathcal{B}}^{\mathcal{C}}\mathbf{X}_{\mathcal{W}})^{-1}\right) \\ -\mathbf{Rot}\left((\mathbf{T}_{\mathcal{B}}^{\mathcal{C}}\mathbf{X}_{\mathcal{W}})^{-1}\right)\left[\mathbf{l}_{\mathcal{W}}^{N,i}\right]_\times \end{bmatrix}^\top \mathbf{Ad}(\mathbf{T}_{\mathcal{B}}^{\mathcal{C}}\mathbf{X}_{\mathcal{W}}) \tag{4.12}$$

$$\frac{\partial h(\mathbf{X}_{\mathcal{W}}, \mathbf{l}_{\mathcal{W}}^{N,i}, \mathbf{T}_{\mathcal{B}}^{\mathcal{C}})}{\partial \mathbf{l}_{\mathcal{W}}^{N,i}} = \Pi\left((\mathbf{T}_{\mathcal{B}}^{\mathcal{C}}\mathbf{X}_{\mathcal{W}})^{-1}\mathbf{l}_{\mathcal{W}}^{N,i}\right)\mathbf{Rot}\left((\mathbf{T}_{\mathcal{B}}^{\mathcal{C}}\mathbf{X}_{\mathcal{W}})^{-1}\right) \tag{4.13}$$

The error function was then defined as:

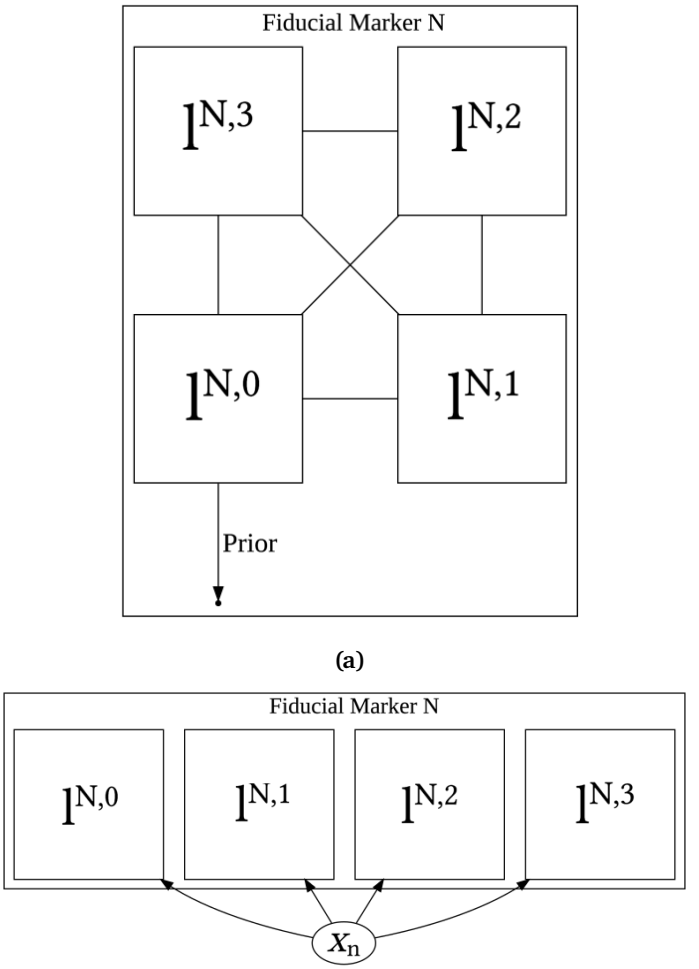
$$e(\mathbf{Z}^N, \mathbf{X}_{\mathcal{W}}, \mathbf{L}^N, \mathbf{T}_{\mathcal{B}}^{\mathcal{C}}) = \sum_{i=0}^{3} \left(h(\mathbf{X}_{\mathcal{W}}, \mathbf{l}_{\mathcal{W}}^{N,i}, \mathbf{T}_{\mathcal{B}}^{\mathcal{C}}) - \mathbf{z}^{N,i} \right)^2 \tag{4.14}$$

This factor was then added to the factor graph, connecting the current pose to the four detected corners with a predefined Gaussian measurement noise model:

$$\mathcal{N}(\mathbf{z}^{N,i}, \mathbf{\Sigma}^i) \tag{4.15}$$

$$\Sigma^i \in \mathbb{R}^{2\times 2} \tag{4.16}$$

Where Σ^i is a tuneable parameter, representing the pixel uncertainty for each detected corner.



s

Figure 4.3: Figure 4.3a: The structure of a fiducial marker as defined in the graph. The lines between the corners are constraints which ensure that the corners are one side length / diagonal away from each other, as defined by the physical marker size. Figure 4.3b: The reprojection factors between a state **Xn** and the four corners $\mathbf{l}^{N,i}$ of a fiducial marker. Note that the subgraph structure shown in Figure 4.3a still exists within this graph, but is hidden to simplify the visualisation. Both generated using the Graphviz tool.

4.4.2 Fiducial Subgraph

Within a single image, multiple fiducial markers can be detected, and can be added to the graph independently of each other, connecting the camera to multiple known landmarks. This can also be done using multiple cameras, as long as the camera frame given in the systems body frame is given. With these constant transforms, the fiducial factors can be transformed to the body frame, making it possible to merge detections from several cameras, and giving the

final pose estimate in the body frame. An example of the graph structure one may get is shown in Figure 4.4. This completed subgraph is then sent to the back-end to be inserted into the iSAM2 optimiser.

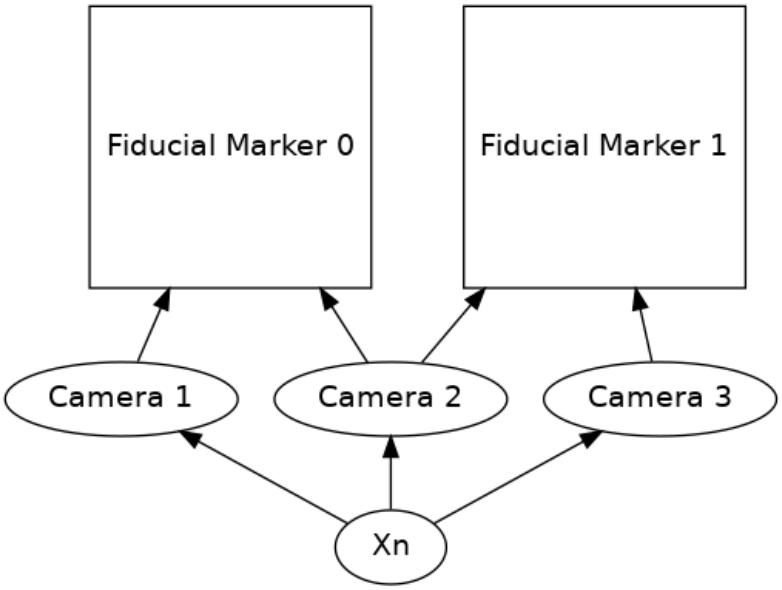


Figure 4.4: An example graph connecting the current system pose to two AprilTag markers through multiple cameras. Generated using the Graphviz tool.

4.5 Back-end

The back-end contained the iSAM2 optimiser, which was executed in a separate thread, and ran the optimisation algorithm at a fixed rate. In between these optimisations, the back-end collected the data from the fiducial subgraph, and any other sensor used. This allowed the subgraph created in Section 4.4.2 to be inserted into the Bayes Tree as described in Section 3.10. As the optimisation was only executed at specific intervals, the iSAM2 update algorithm was executed multiple times after a subgraph was inserted, improving the estimate. The number of times this algorithm was executed after the subgraph was inserted was a changeable parameter, and could be changed e.g. if the optimiser should run faster, and accurate estimates were not necessary. If there were no other sensors involved, the graph was essentially *unconnected*, see Figure 4.5. While the graph was technically connected through the fiducial markers, there were no constraints on the motion between the system states.

The graph could be connected by using odometry from other sensor measurements than the detected fiducial markers, see Figure 4.6a. This could be any form of odometry, such as VO from another visual SLAM system, integrated

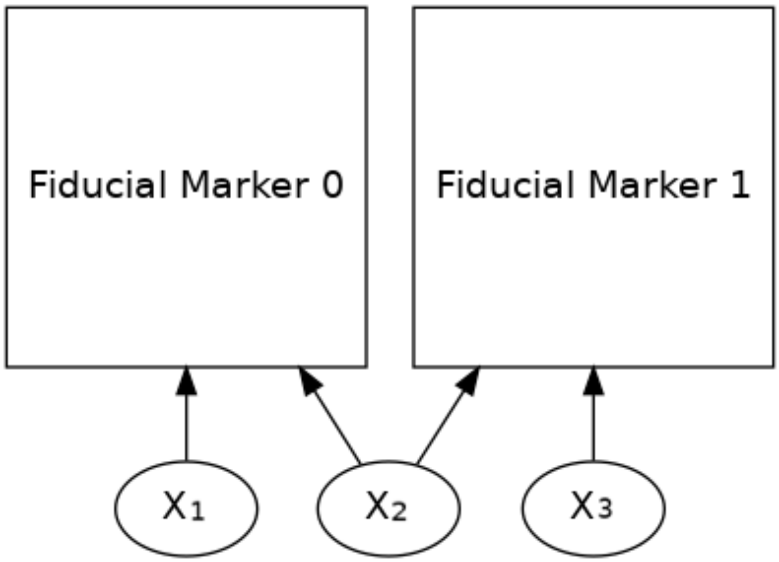
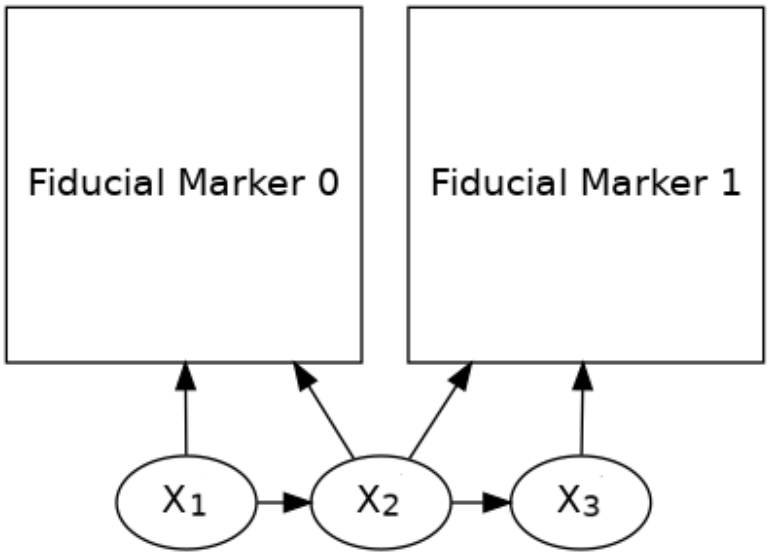
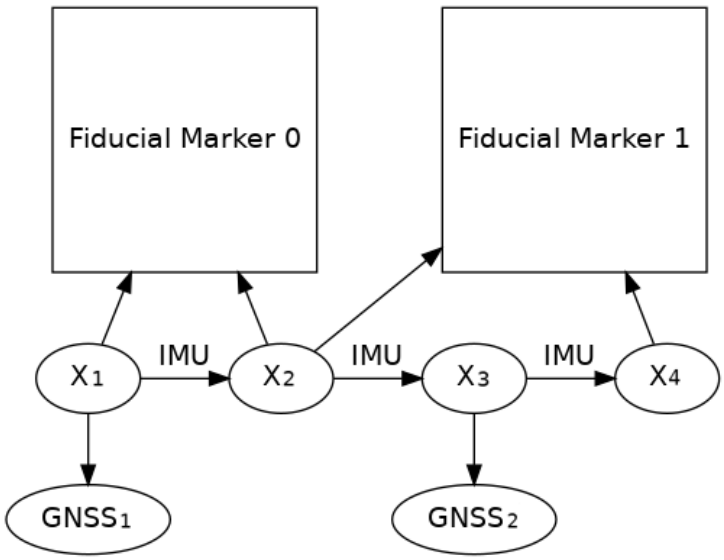


Figure 4.5: An unconnected graph, where each successive pose **X** is connected to the fiducial markers observed at that pose, while not being connected together between each other. Generated using the Graphviz tool.

IMU measurements, lidar odometry, or predictions from motion models. The system could also include other factors than simple odometry, such as GNSS. See Figure 4.6b for an example of such a graph. The measurements don't have to be synchronised, because as long as they were connected by odometry the entire graph will be optimised by the back-end. In cases where the GNSS was either deactivated or very slow, the back-end could use only the IMU and fiducial factors, and then insert the GNSS measurements when they arrived, connecting it between two fiducial factors using the IMU odometry between the measurements.



(a)



(b)

Figure 4.6: Figure 4.6a: A connected graph, where each successive pose **X** is connected to the fiducial markers observed at that pose, and being connected together between each other by odometry. Generated using the Graphviz tool. Figure 4.6b: Connected graph, where each successive pose **X** is connected by odometry, here specifically the IMU measurements between the poses are integrated up. Furthermore, GNSS measurements are also added, which can be done at either the same time as a fiducial marker is added, like at **X1**, or as a standalone subgraph, like at **X3**. Generated using the Graphviz tool.

Chapter 5

Hardware and Software

5.1 Hardware

This section describes the hardware used for the experiments conducted in this work.

5.1.1 milliAmpere

MilliAmpere is a prototype for an autonomous ferry created as part of NTNU's Autoferry project. This project is focused on creating an electric, autonomous ferry for urban environments. The ferry is planned to operate in the canal between Ravnkloa and Fosenkaia, see Figure 5.1, which will result in a 110 m trajectory between the two docks. This will allow cyclists and pedestrians to easily cross the canal, while being less obtrusive than a bridge. The ferry itself is around 5 m × 2.8 m and around 3 m high.

The ferry in operation can be seen in Figure 5.2. The ferry is designed to dock with both the front and back, so that it doesn't need to turn when following the proposed path of Figure 5.1, allowing passengers both on foot and bike to board the ferry. Most of the sensors, like the 360 EO-IR Sensor Rig, lidar, radar and GNSS antennae are placed on top of the enclosed room in the middle of the ferry.

During the spring of 2021, a new, full scale prototype was tested in the water [31]. This ferry, creatively named milliAmpere 2, is about double the length and width of the original described above, and will be able to carry up to 12 passengers. See Figure 5.3 for an image of milliAmpere 2 in the water.

Figure 5.1: Planned operating path of milliAmpere. Illustration by Egil Eide.

Figure 5.2: Image of milliAmpere gathering data. Operated by Martin Græsdal and Martin Eek Gerhardsen. Image by Thomas Hellum.

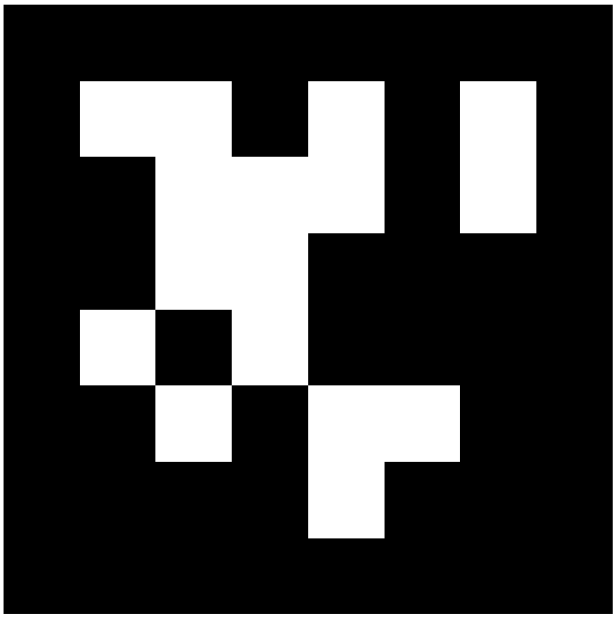
Figure 5.3: Image of milliAmpere 2 being tested in the water. Image source [31].

5.1.2 AprilTag Markers

The markers used in this thesis were produced by *Skipnes Kommunikasjon* as patterns on aluminium plates. The full markers were 1.5 m × 1.5 m, while the size of the pattern, that is the distance between the corners of the AprilTag pattern, was 1.135 m × 1.135 m. This size was chosen as this would allow the markers to be detectable from up to 50 m away, which would cover around half the canal. Then the markers could be placed on either side of the canal, possibly giving milliAmpere full coverage in its operating area.

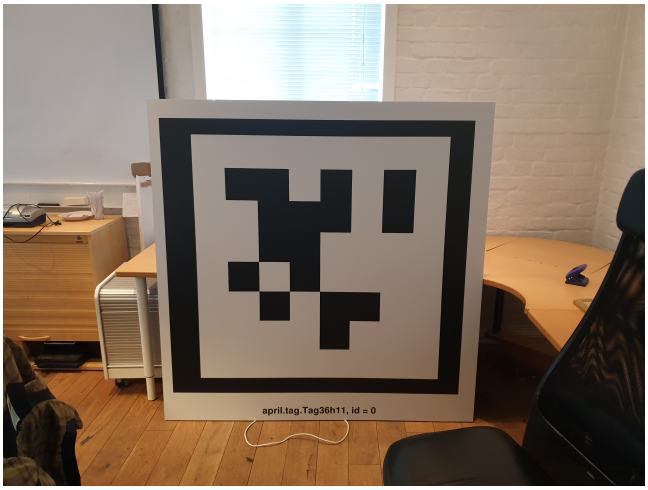
Due to a printing error both markers produced were essentially inverted from the pattern sent to printing. The original pattern for one of the markers, Tag36h11 with id 0, can be seen in Figure 5.4a. The corresponding printed marker can be seen in Figure 5.4b. While the image colours can be inverted in software, this can still pose some problems. One of the primary problems this can cause is due to how the border is only partially inverted. To ensure that the marker stands out from the surroundings, there is supposed to be a border around the marker itself, which is completely white and the size of one bit (that is, one square) in the pattern. Because of the way the pattern was sent to printing, only the section cut from the original sent to *Skipnes Kommunikasjon* was inverted, while the outside was kept white, resulting in the uneven border seen in Figure 5.4b. The border was therefore smaller, and not of uniform size. It is highly unlikely that this will have any major effects, but it will most likely

decrease the detection probability for the markers. Another possible problem lies in using other detectors and algorithms on this test data, as it might be more difficult to invert the images before running the detector.



april.tag.Tag36h11, id = 0

(a)



(b)

Figure 5.4: Figure 5.4a: The original Tag36h11 marker, id 0. Figure 5.4b: The produced Tag36h11 marker, id 0. The colours are inverted and the border outside the pattern is smaller than it should be.

5.1.3 360 EO-IR Sensor Rig

The 360 EO-IR Sensor Rig is the camera sensor system used to gather the datasets for this project. It consists of five Electro-Optical (EO) and five Infra-Red (IR) cameras, fixed with 72° angles between (see Figure 5.5), a camera synchronisation interface, and an NVIDIA TX2 Carrier. The EO cameras are FLIR BlackFly 2, each with an image resolution of $2448x2048$, while the IR cameras are FLIR Boson 640, each with an image resolution of $640x512$. Each EO camera has a theoretical maximum data rate of around $450Mbit/s$, each IR camera has a theoretical maximum data rate of $480Mbit/s$, while the maximum data rate between the NVIDIA TX2 Carrier and a secondary processing computer is $1Gbit/s$. However, as the maximum frame rate of the IR cameras is 9 fps, the actual maximum data rate will be $30Mbit/s$. This should leave the EO cameras a theoretical maximum frame rate of slightly more than 4 for each camera at the maximum image resolution. The relevant transforms for the cameras are listed in Appendix A.1. As the transforms between the milliAmpere vessel centre and the IR cameras were given by the 360 EO-IR Sensor Rig documentation, the transforms between the vessel centre and the EO cameras were calculated by composing the transform to the IR cameras with the measured transform between the IR and EO cameras. This was difficult to do properly, as the components in the 360 EO-IR Sensor Rig were mounted close to each other. The EO cameras were also mounted at an angle, pointing slightly down compared to the IR cameras, which was difficult to measure accurately.

In the experiments conducted in this thesis, the image resolution was set to $1224x1024$ by *only using* the centre of the larger image. This ensured a reasonable data rate which allowed the data to be recorded on an external laptop.

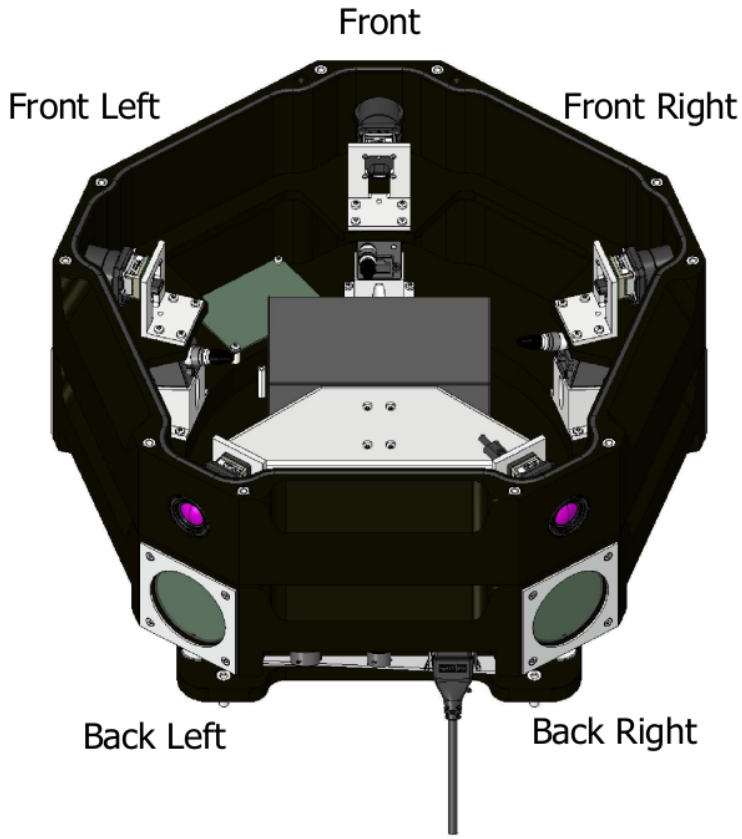


Figure 5.5: The 360 EO-IR Sensor Rig, with its five EO and five IR cameras.

5.1.4 Global Navigation Satellite System

The GNSS receiver mounted on milliAmpere is a Hemisphere Vector VS330, supporting RTK GNSS, allowing the accuracy of the position measurement to be improved from ±0.30 m to ±0.01 m. Using two antennae and an internal gyro, it is also possible to measure the heading with an accuracy of ±0.05°. The receiver communicates using a Satel Radio Link mounted under the roof of milliAmpere. The update rate of the receiver is specified as 20 Hz.

For the measurement of the position of the AprilTag markers, a Spatial Dual GNSS aided Inertial Navigation System (INS) was utilised. The antennae were fixed in place on a wooden plank with 0.5 m between. With RTK, the accuracy of the position measurement can be improved from ±1.2 m to ±0.008 m. The heading could also be estimated with an accuracy of ±0.1° because the device used two antennae, and finally, the update rate of the INS was specified to 20 Hz.

5.1.5 Inertial Measurement Unit

The IMU used on milliAmpere is an Xsens MTI-G-710. This is an INS with a GNSS-receiver, magnetometer and barometer as well as the IMU, which provides acceleration and angular velocities in the body frame. It has an update rate of 100 Hz.

5.2 Software

This section describes the primary software used for this project.

5.2.1 Robot Operating System

Robot Operating System (ROS) is an open source collection of software modules, libraries and tools for robot systems. It is a modular middleware that allows separating different parts of the system into individual nodes, and standardising the communication between these nodes. ROS also comes with a lot of useful tools like *rosbag*, which allows for easy recording of data in a ROS-friendly format, and *rviz*, which allows for easy visualisation of the ROS data.

5.2.2 EO Camera Driver

The EO cameras are interfaced using the FLIR Spinnaker Software Development Kit, and are further extended to work as a ROS module. An external camera synchronisation interface is connected to the cameras, and software for extracting and integrating hardware triggered images was written during the summer of 2020 and expanded upon during the spring of 2021. The hardware triggering was not working for all the cameras until the spring of 2021, see Gerhardsen [28], when a hardware problem was fixed by Maritime Robotics. The hardware triggering was then tested and confirmed to work, but because the hardware triggering circuit needed an input from the GNSS, one had to choose between using the hardware triggering mode for the cameras or using the INS mentioned in Section 5.1.5 during experiments. As this would both be used for the IMU data and as an INS to estimate a ground truth to compare results with, it was decided not to use the hardware triggering mode.

This does however lead to the driver running suboptimally, as the images were not necessarily synchronised, nor were the timestamp as precise as they could've been. The non-hardware triggered method continuously waits for new images produced by the cameras in one thread, loads them into memory, and then at specified intervals publishes them to the ROS environment from another thread. Potentially, this can cause problems if the time it takes for an image to be either produced by the camera or be read by the driver added with the time it takes to publish the image to ROS doesn't fit within the specified frame rate.

This would lead to the driver publishing the same image multiple times. To avoid both reading from the camera and publishing to ROS at the same time, and thus corrupting the data, these processes were locked by a mutex during these operations. This method was tested, and produced images without any of the possible problems listed, and was therefore still able to produce useful images.

5.2.3 AprilTag Detection Algorithm

The AprilTag detection algorithm described by Olson [51] and improved by Wang and Olson [68] was utilised. This algorithm takes in a gray scale image and detects all AprilTags of a specified family within this image, and outputs all detections. See Section 3.6.2 for more information on the markers and detector specifically.

5.2.4 The Open Computer Vision Library

The OpenCV library is an open-source library for computer vision, see Bradski [6]. This project was initially started in 1999 to create a free, easy and useful library for computer vision problems, and has been continuously worked on since then and has therefore been a popular tool in many other computer vision projects since its creation. OpenCV is primarily written in C/C++ for efficiency, and has wrappers for other programming languages, such Python. This has also contributed to the popularity of the library, as the Python programming language is considered more beginner friendly than C/C++. Many standard computer vision tools and algorithms are already implemented in OpenCV, one being the IPPE PnP algorithm discussed in Section 3.7.

5.2.5 AprilTag Fiducial SLAM

The Fiducial SLAM method described in Chapter 4 was implemented in the C++ programming language. The Independent Camera Thread, Fiducial Map, Graph Creation and Back-end modules were implemented as separate nodes in the ROS framework, and using the same framework for intermodular communication. In the Independent Camera Thread, the AprilTag detector was used. A ROS synchronisation scheme was used to synchronise the detections from images taken at the same time, which was sent to the Graph Creation module. Both the fiducial pose factor and the fiducial projection factor were implemented using the GTSAM library and connected to the current system pose. The Independent Camera Thread and the Graph Creation modules are both able to extract information from the Fiducial Map to complete their respective tasks.

Four methods were implemented here, the unconnected fiducial pose factor, the connected fiducial pose factor, the unconnected fiducial projection factor and the connected fiducial projection factor. The method is unconnected if it

only relies on the fiducial factor for pose estimation, and there were no factors constraining the relative motion between the incremental poses. This is illustrated in Section 4.4 by Figure 4.5. The method is connected if there are factors other than the fiducial factor connecting the incremental poses. In this case, the poses were connected using preintegration of the IMU measurements between the poses, using the preintegrator and corresponding factors implemented in GTSAM. Preintegration was described in detail in Section 3.11. As this method was supposed to emulate a more realistic implementation of a mulitsensor SLAM method, a GNSS measurement was added to the graph every 10 seconds to emulate a slow, but reliable GNSS. This is also illustrated in Section 4.4 by Figure 4.6b.

The Back-end module used the iSAM2 method for optimisation, implemented in GTSAM. When given a fiducial subgraph, this was inserted into the optimiser. At a fixed rate, the optimiser was executed and the result used to update the Fiducial Map. Once the system was shut down, the result was extracted from the optimiser and stored for future analysis.

Chapter 6

Experiments

This chapter will discuss the planning, preparations, and execution of the experiments, as well as the gathering and post-processing of the data. The data was gathered using milliAmpere, the prototype for NTNU's autonomous ferry described in Section 5.1.1. These experiments were conducted near milliAmpere's planned operating area, near Ravnkloa and Fosenkaia in Trondheim. The area is illustrated in Figure 6.1[1], see OpenStreetMap contributors [52].

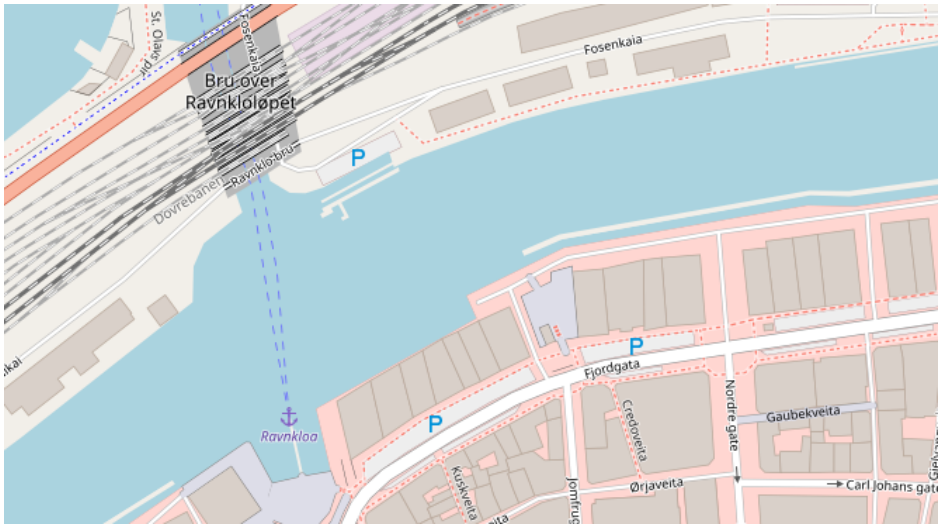


Figure 6.1: Map of the logging area near Ravnkola and Fosenkaia.

The new experiments conducted in this thesis were necessary, as there were essentially no relevant datasets available. As part of the specialisation project by Gerhardsen [28], datasets for similar scenarios as described in Section 6.2 were recorded. However, they were virtually unusable for the purposes of this thesis, as multiple problems with the datasets arose while working with them

[1] https://www.openstreetmap.org/#map=18/63.43478/10.39440

during the specialisation project. The primary problems are described in the specialisation project, though are quickly summarised here:

- EO-camera: The cameras ran on different and slow frame rates, and often didn't produce images for long stretches of time.
- AprilTag GNSS measurements: The logging method relied on data from a noise-prone mobile phone and also didn't take into account the orientation of the tag.
- AprilTag family: The AprilTag family used was the Tag16h5 family, which turned out to have such a high false detection rate that using the detections was very difficult.

These problems were taken into account when preparing for these experiments. The image size for the EO cameras was decreased, the GNSS measurements of the AprilTag markers were done using an RTK-GNSS and another AprilTag family was used.

6.1 Planning

The planning of the experiments was done in cooperation with Kristian Auestad, Martin Græsdal, and Thomas Hellum, who were also interested in using milliAmpere to gather data for their work. Therefore, milliAmpere was booked between the 12th and 16th of April. The cooperation was also mutually beneficial, as multiple people were necessary for operating milliAmpere, in addition to ensuring that the sensors are working correctly and the data is being recorded properly.

6.1.1 AprilTag Markers

The AprilTag fiducial markers were chosen for these experiments. As discussed in the introduction to Chapter 6, the AprilTag family used previously lead to an excessive amount of false detections, which were difficult to deal with. In these experiments, the Tag36h11 family was used instead. This is considered the standard AprilTag family, as it has relatively many different possible patterns, has a high Hamming distance, and is compatible with the ArUcO detector. The size of the AprilTag family means that it can be useful in many situations, while the Hamming distance ensures that there will be limited amounts of false detections.

The AprilTag markers were fastened to the outside of the railing near the edge of the canal by Ravnkloa, and the positions recorded with the INS described in Section 5.1.4. The estimated pose for each AprilTag marker can be seen in Appendix A.2. This position was chosen because it allows a number of different scenarios to be tested, such as viewing only one marker at a time, viewing both, and using multiple cameras to view the markers. The specific

scenarios used to gather data for this thesis will be discussed further in Section 6.2. The markers can be seen in Figure 6.2, where they are close enough to the canal to be observed by the EO cameras on the 360 EO-IR Sensor Rig and not be obstructed by any specific objects like cars, docked boats, or poles.

Figure 6.2: The AprilTag (Tag36h11 family) markers used in this thesis. The AprilTag marker on the right had id 0 and the marker on the left had id 1.

6.1.2 Data Recording

The sensors used in these experiments were, in addition to the IMU, GNSS, and the 360 EO-IR Sensor Rig, as discussed in **??**, a stereo camera system mounted to the roof of milliAmpere and the lidar mounted to milliAmpere. They are not described further here as they are not relevant for this thesis, however this data could still be useful for future works.

The EO and IR data from the 360 EO-IR Sensor Rig was recorded on one laptop, together with the lidar data. The stereo camera data was recorded on a second laptop. This resulted in two rosbags of data for each scenario. The IMU and GNSS measurements were recorded in a rosbag by the milliAmpere control system, lasting from the moment the ferry control system is started to when it is turned off. The data was split for multiple reasons, primarily because amount of data generated by the different camera systems was to large for a single laptop to handle, and because the primary control computer on milliAmpere should not be used for any other purpose than controlling the ferry.

6.2 Scenarios

In this section, the different scenarios for gathering data are described. A varied collection of datasets handling different distances, multiple cameras in view

of the markers, and different number of detectable AprilTag markers in the camera scene were recorded.

6.2.1 Scenario 1

In this scenario, milliAmpere was positioned close to the AprilTag markers, facing the shore and the markers. Because milliAmpere was very close to the harbour, the cameras could only observe one marker initially, though both were quickly within view of the cameras. The ferry then reversed across the canal to the opposite side, stopped and returned towards the initial position. By minimising the amount of change to the orientation, a simpler scenario was created for testing where one may e.g. use a single camera for a proof-of-concept. The relatively straight motion also makes it simpler to test the AprilTag detection algorithm in a controlled environment, where the distance from the ferry to the markers should be the primary reason for not detecting a marker in view of the camera. The path taken, along with the position of the AprilTag markers, can be seen in Figure 6.3.

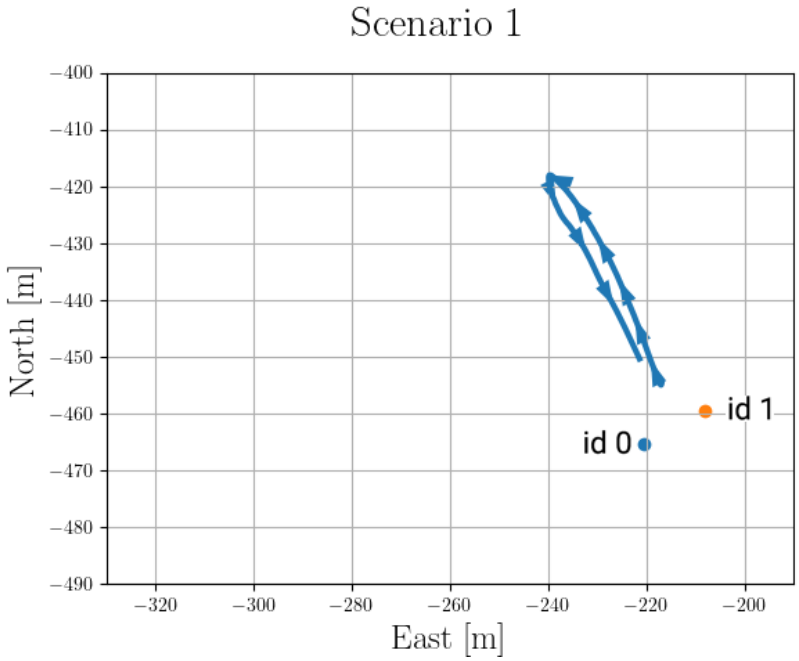


Figure 6.3: Map of milliAmpere's path for scenario 1. Generated from the onboard INS. The points represent the position of the AprilTag markers, and are labeled with the correct id. Note that the arrows show the direction of milliAmpere's movement and not its orientation.

6.2.2 Scenario 2

For scenario 2, milliAmpere was positioned north-east of the markers, up the canal, facing the shore and not directly observing the markers. Then, still facing the shore, milliAmpere moved down the canal, passing the markers. The ferry was again positioned relatively close to the shore. Similarly to the scenario described in Section 6.2.1, there were very little major change to the orientation of the ferry during this scenario, creating a simple scenario for testing. In this case, the short range detection capabilities of the AprilTag detection algorithm and the 360 EO-IR Sensor Rig cameras could be tested when approaching from the side. The path can be seen in Figure 6.4, together with the position of the AprilTag markers.

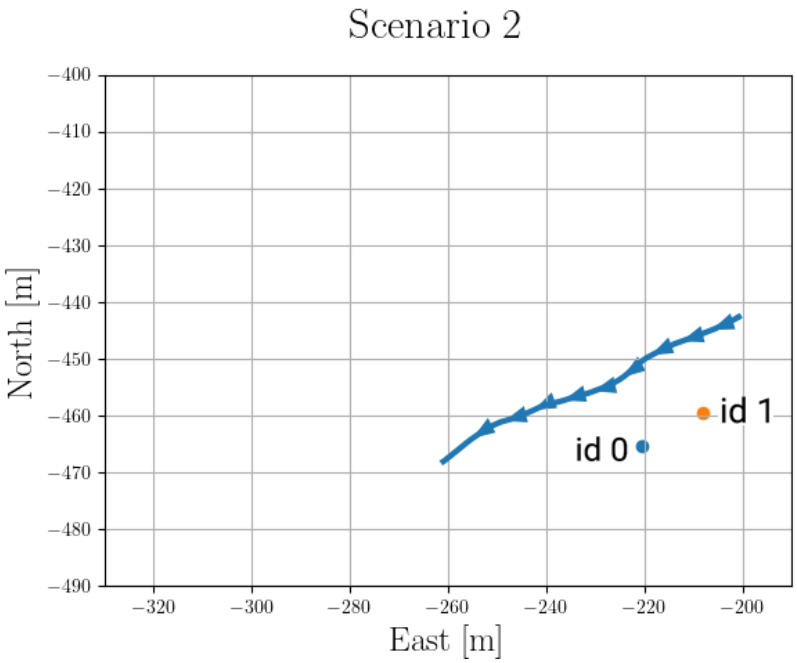


Figure 6.4: Map of milliAmpere's path for scenario 2. Generated from the onboard INS. The points represent the position of the AprilTag markers, and are labeled with the correct id. Note that the arrows show the direction of milliAmpere's movement and not its orientation.

6.2.3 Scenario 3

In scenario 3, milliAmpere was positioned south-west in the canal, facing the shore similarly to scenario 2. MilliAmpere then moved north-east up the canal facing the AprilTag markers, tough with a considerably greater distance to the markers, as milliAmpere was around half way across the canal. As the primary difference here was distance to the marker, the usability of the method at a

reasonable working range can be tested. As this was the greatest distance from either side of the canal, this was the greatest distance the markers should still work at, as in real operations one should place multiple markers on both sides of the canal. This path can be seen in Figure 6.5.

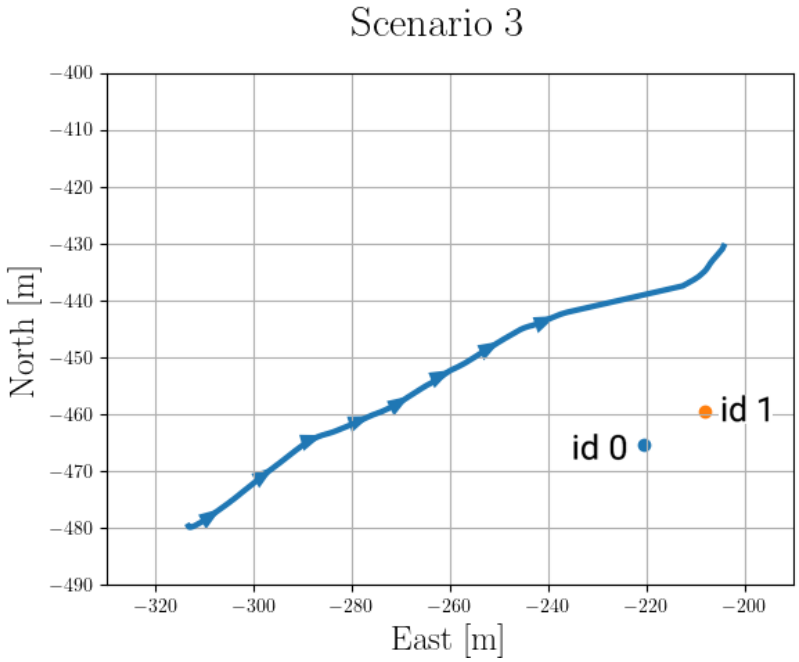


Figure 6.5: Map of milliAmpere's path for scenario 3. Generated from the onboard INS. The points represent the position of the AprilTag markers, and are labeled with the correct id. Note that the arrows show the direction of milliAmpere's movement and not its orientation.

6.2.4 Scenario 4

For scenario 4, milliAmpere was positioned north-east in the canal, facing the shore similarly to both scenario 2 and 3. Then milliAmpere moved south-west down the canal, facing the AprilTag markers, although the distance to the markers again was increased significantly. The ferry moved close to the harbour banks on the opposite side of where the markers were placed, which allowed testing at the extremes of the distances one would expect milliAmpere to meet when working in real situations. The path can be seen in Figure 6.6.

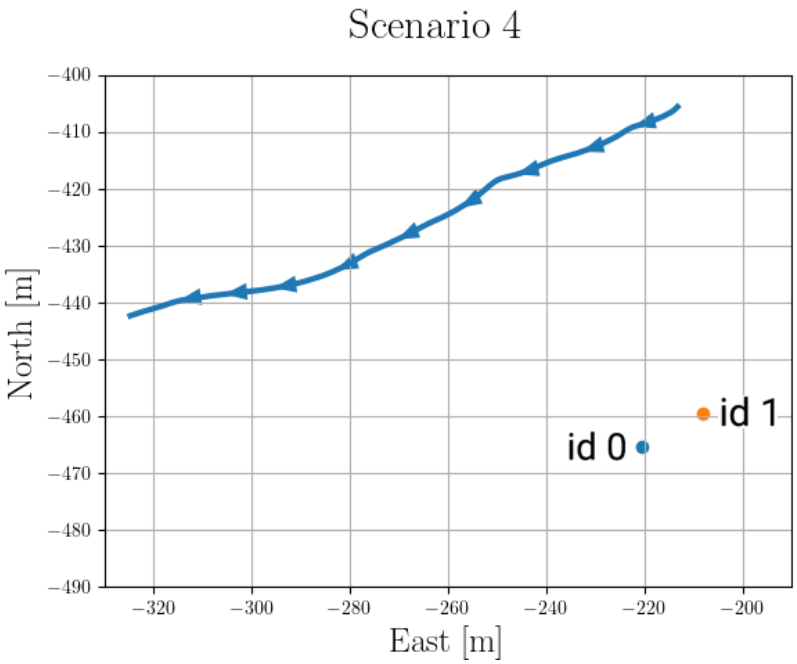


Figure 6.6: Map of milliAmpere's path for scenario 4. Generated from the onboard INS. The points represent the position of the AprilTag markers, and are labeled with the correct id. Note that the arrows show the direction of milliAmpere's movement and not its orientation.

6.2.5 Scenario 5

The final scenario, scenario 5, was conducted as a more realistic alternative to the scenarios described above, as well as serving as a more traditional dataset for general SLAM problems. The ferry started to the west of the markers, well outside their range and moved towards them. When close, milliAmpere turned north to the other side of the canal, turned west and moved south-west down the canal. Then, milliAmpere turned north towards the initial position, such that the path has a similar shape to the number 8. The full path can be seen in Figure 6.7.

In terms of AprilTags specifically, this is an interesting scenario as the ferry initially has no way of observing the markers, and the systems ability to handle long stretches without the markers can be tested. However, the data would also be useful for other situations, and can be especially useful for testing multi sensor SLAM methods for milliAmpere.

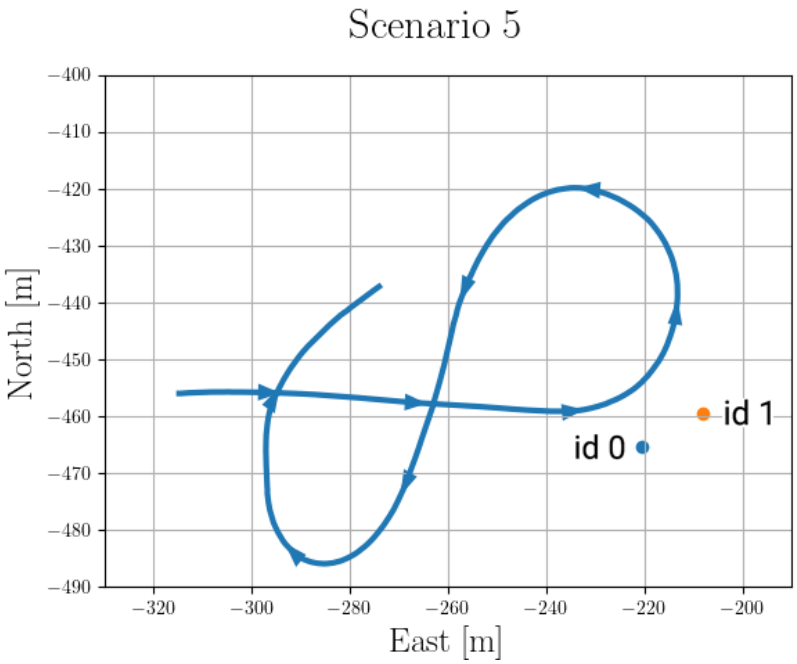


Figure 6.7: Map of milliAmpere's path for scenario 5. Generated from the onboard INS. The points represent the position of the AprilTag markers, and are labeled with the correct id. Note that the arrows show the direction of milliAmpere's movement and not its orientation.

6.3 Pre-processing

As noted in Section 6.1.2, the data was recorded in multiple different rosbags. To simplify future work with the data, these separate rosbags were merged for each scenario.

To merge the relevant data into a single rosbag for each scenario, the three relevant rosbags are all opened in software. Then, based on which has the *latest* start time and the *earliest* end time, the data in this range was merged into a single rosbag containing all the data recorded. As all the computers utilised in these experiments were connected to the internet through milliAmpere, the timestamps were accurate enough that nothing was done with this potential problem.

Some changes were done to the data before it was recorded in the new rosbag. As all EO data was stored in a compressed form, these images were stored in a more user friendly manner. Because the ROS standard states that image data should be published on one topic, and camera calibration data should be published on another, the calibration topic was also added during this process. This was done because the original topics were not always synchronised with

the image data, and by redoing this, synchronisation was ensured. However, this was only done for the data from the 360 EO-IR Sensor Rig, as the stereo camera data calibration was not available during this process. Otherwise, nothing was changed about the data.

Chapter 7

Results

In this chapter, the results of using the Fiducial SLAM method discussed in Chapter 4 on the experimental data discussed in Chapter 6 is presented. To evaluate the methods tested, the equations presented in Section 3.2 were used. The performance measures RMSE, Equation (3.14), and maximum RSE, Equation (3.16), will be used in estimates where at least one marker was detected. To inspect the influence of using multiple cameras, performance measures were calculated for cases where multiple cameras were used for estimation, as well as for cases where a single camera was used for estimation.

For each scenario described in Section 6.2, the results of using the fiducial pose factor described in Section 4.4.1 and the fiducial projection factor described in Section 4.4.1 were tested. Both factors were tested using an unconnected and a connected graph, where the preintegration method described in Section 3.11 was used to connect the graph, and a GNSS measurement was added around every 10 seconds to emulate a reasonable GNSS. Note that the Fiducial SLAM method calculates the estimated pose of milliAmpere in 6 degrees of freedom, though to save space only the 2D pose is shown in the figures, that is the first and second translational axes, and the third rotational axis of the NED coordinate system. The first translational axis is the North axis, the second is the East axis, and the third rotational axis is the yaw axis. As milliAmpere is an ASV, these axes are the most important.

7.1 Scenario 1

This section presents the results of the fiducial SLAM method for scenario 1, which was described in Section 6.2.1. In this scenario, milliAmpere moved away from the fiducial markers, facing them. The number and frequency of detections decrease noticeably in the time range between 640 and 680 seconds, when milliAmpere is the furthest away from the markers. Because multiple cameras detect the markers only in the beginning of the scenario, the performance measures for this case will essentially be a performance measure for where

milliAmpere is very close to the markers.

7.1.1 Unconnected Fiducial Pose Factor

The estimated 2D pose of milliAmpere is shown together with the ground truth values in Figure 7.1, and the number of detections at the given time instance. The RSE for the same values is shown in Figure 7.2. The performance measures are listed in Table 7.1. As can be noted from the table, the RMSE is low for the orientation, being under 8° for all angles, for all detections, and for multiple camera detections and single camera detection. North RMSE is also overall the lowest positional measure. This is the primary axis of motion in this scenario, and is correlated with the distance to the markers, as milliAmpere is first moving north, away from the markers, then back south towards them. The maximum RSE is low for the first axis rotations, and is lower for the multiple camera detections than the single camera detections. This is most likely because the multiple camera detections correspond to the time milliAmpere is very close to the markers. For detections at greater distances, the error is much higher, see Figure 7.2 at e.g. 675 seconds.

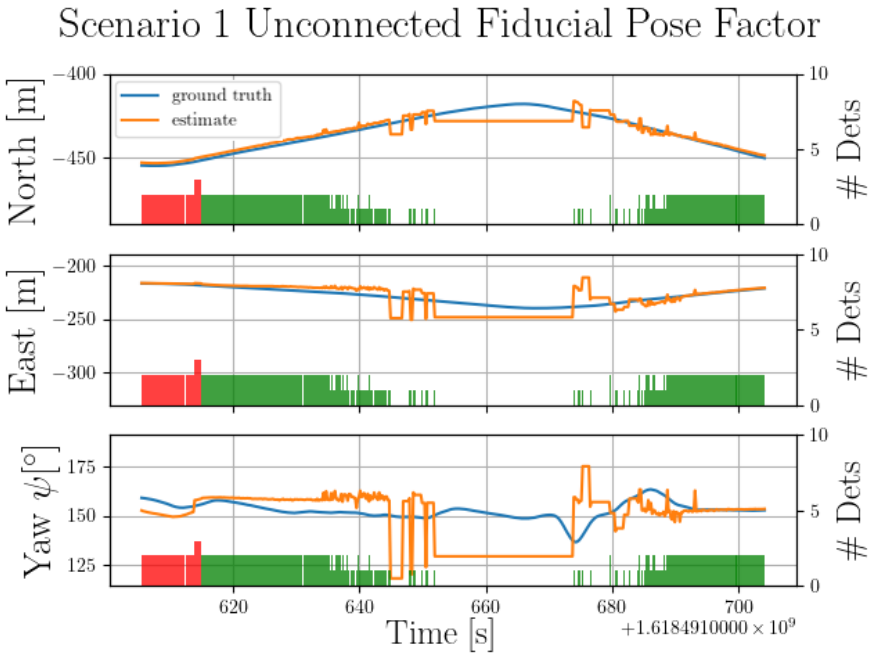


Figure 7.1: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

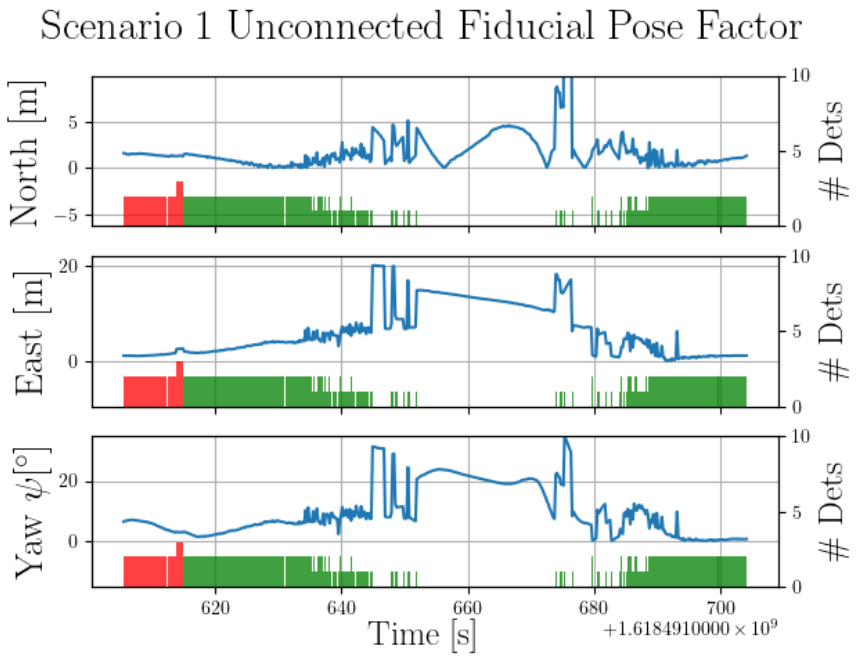


Figure 7.2: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.1. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.1: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	2.2	5.9	3.4	4.3	1.8	5.9
2. axis [°]	6.6	29	0.57	1.0	7.2	29
3. axis [°]	7.2	35	5.4	7.0	7.5	35
1. axis [m]	2.1	24	1.4	1.6	2.2	24
2. axis [m]	4.5	20	1.5	2.6	4.9	20
3. axis [m]	5.0	20	6.4	6.5	4.7	20

7.1.2 Connected Fiducial Pose Factor

The estimated 2D pose of milliAmpere is plotted with the ground truth in Figure 7.3, together with the number of detections used at that time instance. The RSE calculated between the estimated and ground truth 2D pose is shown in Figure 7.4. It can be observed that the error is smaller than for the unconnected fiducial pose factor method both while the system is able to observe the markers, as well as in the time range between 640 and 680 seconds where there were very few detections. The improvement shown in that time range can most likely be put down to the inclusion of accurate and infrequent GNSS measurements, which can be observed where the error plots *bounce* back close to zero.

The performance measures are listed in Table 7.2. For the rotation, the RMSE is slightly better for the first axis, and slightly worse for the second axis when compared to the unconnected method in Section 7.1.1. The third axis is better for all detections and single camera detections, but is distinctly worse for the multi camera detections, both compared to the unconnected method and compared to the performance measures for all detections and single camera detections. As can also be observed for the yaw RSE at the multiple camera detections in Figure 7.4, there seems to be a significant offset in the error. Looking at the maximum RSE measures, the same trend is observed for the first and third axis, while the second axis is slightly better compared to the unconnected method. For the translation, both the RMSE and the maximum RSE is better for all axes.

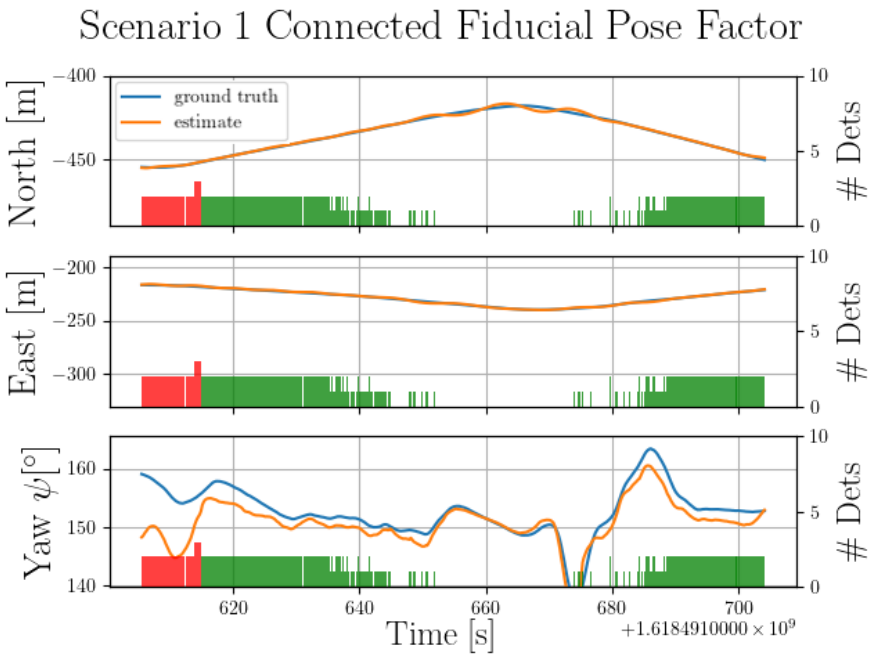


Figure 7.3: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.2: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	1.4	4.6	1.6	2.5	1.4	4.6
2. axis [°]	10	27	19	27	7.8	17
3. axis [°]	4.1	12	9.0	12	2.3	4.0
1. axis [m]	0.36	1.6	0.39	0.64	0.35	1.6
2. axis [m]	0.54	2.1	0.66	0.96	0.51	2.1
3. axis [m]	1.2	3.9	2.6	3.9	0.56	3.3

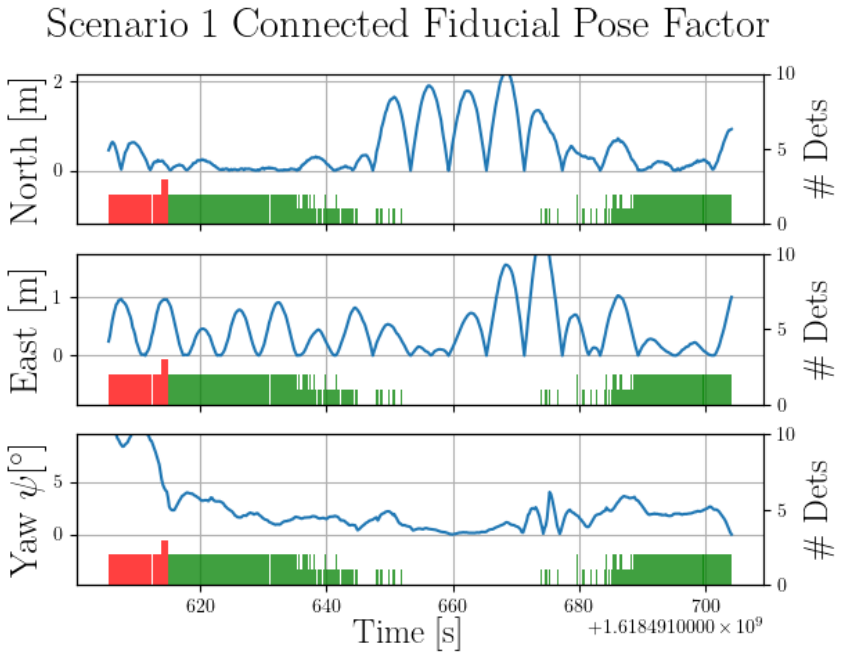


Figure 7.4: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.3. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

7.1.3 Unconnected Fiducial Projection Factor

The estimated 2D pose of milliAmpere is visualised together with the ground truth in Figure 7.5, while the RSE of the estimate and the ground truth is shown in Figure 7.6. The performance measures for both the rotation and translation of the RSE between the estimated and ground truth 3D pose for milliAmpere are presented in Table 7.3. With the exception of both the rotation and translation of the second axis, the performance measures are generally better than the measures using the fiducial pose factor described in Section 7.1.1. The maximum RSE is also generally lower for the projection factor as well.

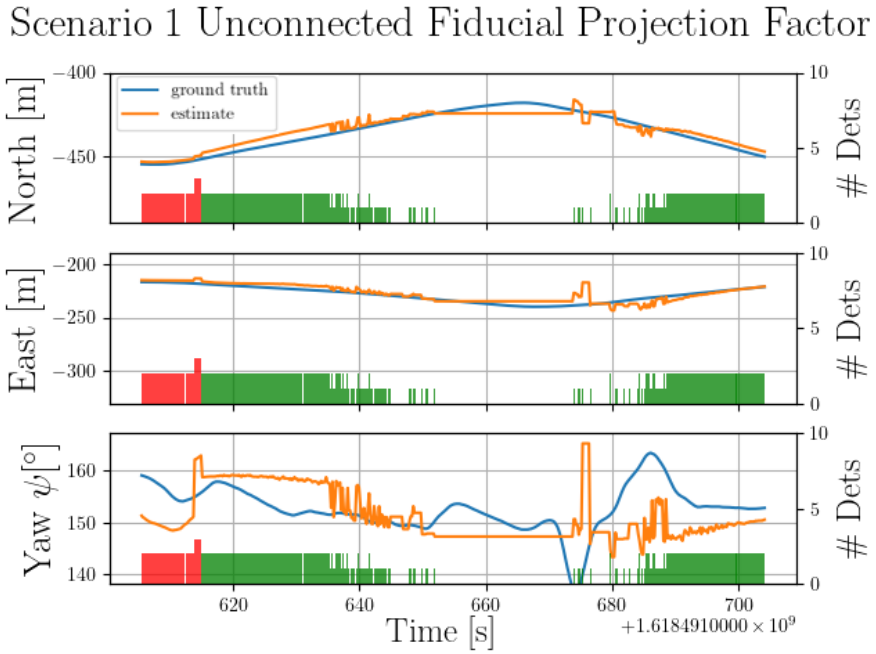


Figure 7.5: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

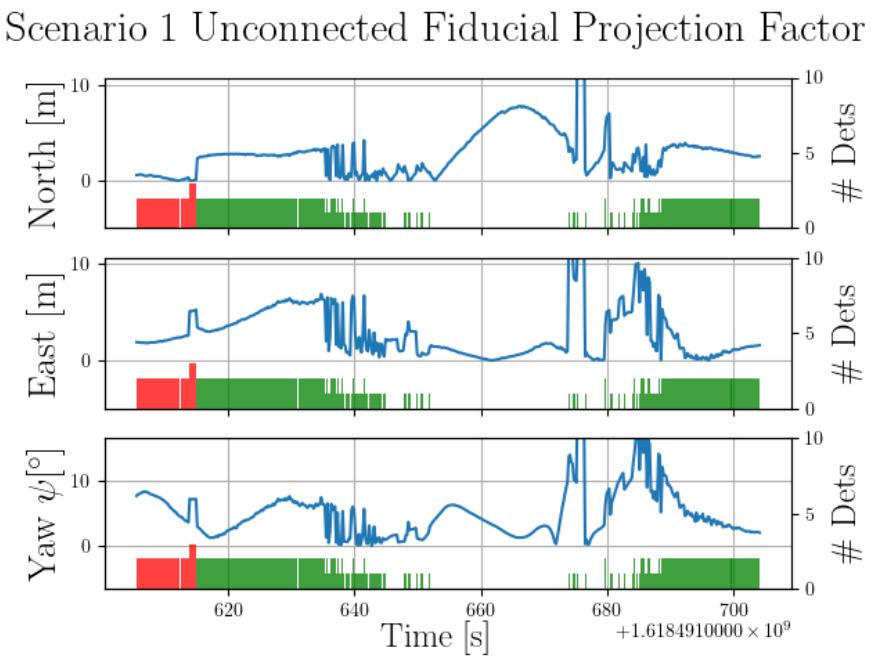


Figure 7.6: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.5. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.3: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	1.4	4.0	2.3	3.1	1.2	4.0
2. axis [°]	7.2	15	1.4	2.2	7.8	15
3. axis [°]	6.3	25	6.8	8.3	6.2	25
1. axis [m]	2.7	19	0.39	0.68	2.9	19
2. axis [m]	4.1	12	2.7	5.2	4.3	12
3. axis [m]	8.4	23	6.0	6.5	8.8	23

7.1.4 Connected Fiducial Projection Factor

The estimated 2D pose of milliAmpere is visualised together with the ground truth in Figure 7.7, while the RSE of the estimate and the ground truth is shown in Figure 7.8. For the multi camera detections, the offset observed for the connected fiducial pose factor method is still observable, especially for the yaw. The error for North and East are also greater here compared to the connected fiducial pose factor. The performance measures are listed in Table 7.4. Comparing these results to the results for the unconnected fiducial projection factor method, the results are better for all measures, which was to be expected after noting the improvement using a connected over an unconnected method as discussed in Section 7.1.2. However, these results are generally worse than the connected fiducial pose factor method. Comparing the RSE plots in Figure 7.8 and Figure 7.4 might give the impression that the connected fiducial projection factor outperforms the connected fiducial pose factor. However the largest error values in the connected fiducial pose factor are concentrated around time ranges with few detections, and are therefore less reliable. This results in the performance measures for the connected fiducial pose factor method outperforming the connected fiducial projection factor.

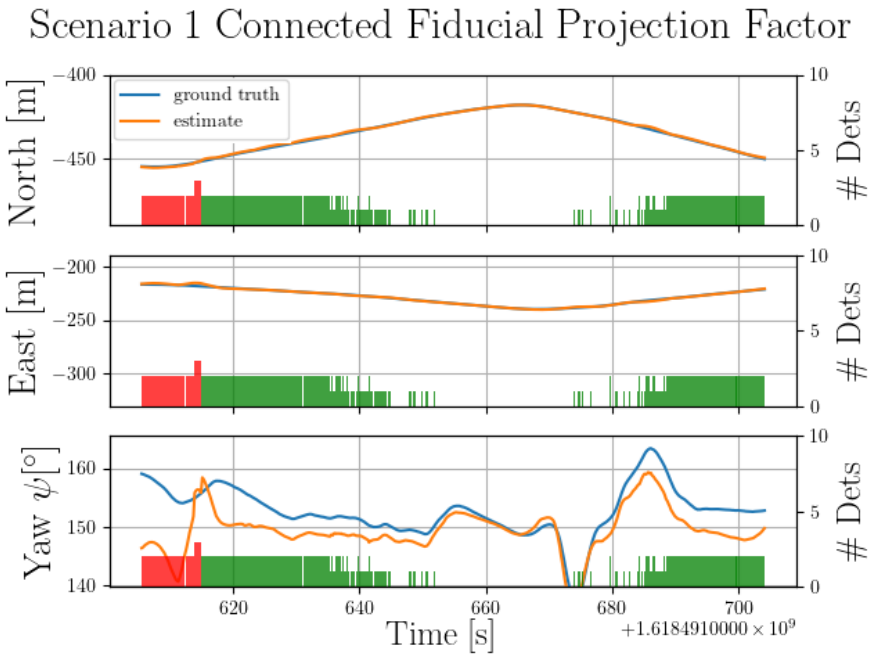


Figure 7.7: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

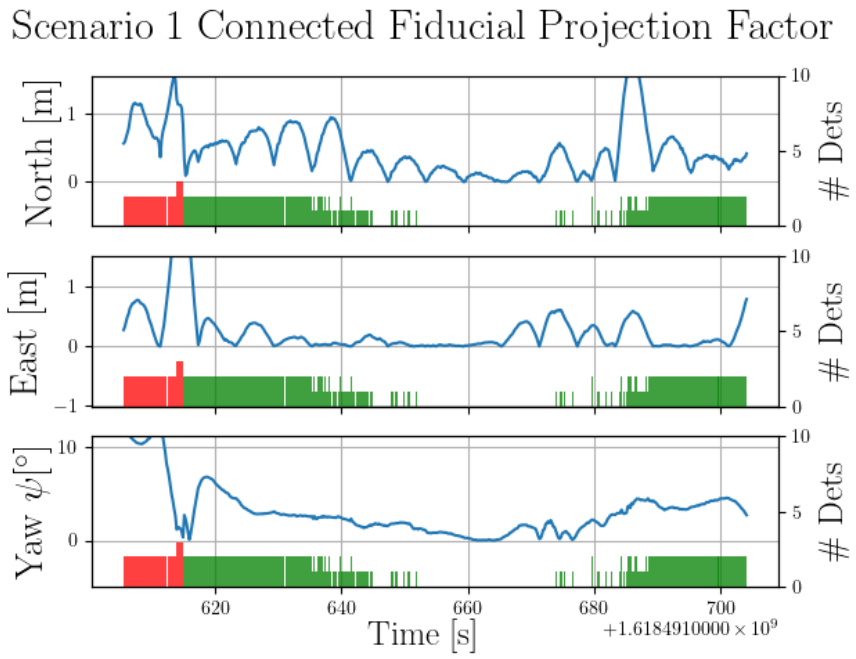


Figure 7.8: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.7. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.4: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	1.8	5.7	3.5	5.7	1.3	4.3
2. axis [°]	15	27	14	27	15	18
3. axis [°]	5.0	12	9.7	12	3.6	6.8
1. axis [m]	0.7	1.8	0.98	1.5	0.64	1.8
2. axis [m]	0.6	2.6	1.2	2.6	0.41	2.5
3. axis [m]	1.5	3.7	2.5	3.7	1.3	3.0

7.2 Scenario 2

This section presents the results of applying the fiducial SLAM method on scenario 2, which was described in Section 7.2. In this scenario, milliAmpere is very close to the markers, and is moving parallel to both the markers and the canal. For the majority of the scenario, at least one marker is within view. There are also two major time ranges, between 840 and 855 seconds and between 865 and 885 seconds, where multiple cameras are used to detect the markers.

7.2.1 Unconnected Fiducial Pose Factor

The estimated and ground truth 2D pose of milliAmpere for the unconnected fiducial pose factor is shown in Figure 7.9, with the RSE between the two shown in Figure 7.10. In the RSE plot, two major increases in the error can be observed. The first can be seen in the time range 830 and 840 seconds, which corresponds with a change from a single camera observing the markers to multiple cameras observing the markers. This spike continues while there are several rapid changes between multiple cameras detecting the markers and a single camera detecting the markers. The second spike is in the time range between 880 and 900 seconds, and will be discussed in depth later. The performance measures are listed in Table 7.5. The RMSE is lower for where multiple cameras detect the markers compared to where only a single camera detects the markers.

In the time range between 880 and 900, there was a significant change in the North position of milliAmpere which can be observed in both Figure 7.9 and Figure 7.10. From the visualisation of the detections, it was also observed that only one camera detects the markers, namely the front left camera. The images from this camera right before, at, and after the event were captured in Figure 7.11, respectively. It can be observed that the AprilTag marker with id 0 is well within view, while the marker with id 1 is further away, still the corner detections seem good for all images. Furthermore, it can be observed that the three images are essentially equivalent in Figure 7.11, and that the detected corners are visualised in the same areas, meaning that the detections themselves did not produce the observed error spike.

The result from the IPPE PnP method is therefore explored and compared with the predicted relative transform between the camera frame and the AprilTag fiducial markers. The predicted transform was calculated using the estimated pose of milliAmpere and the estimated pose of the fiducial markers from the fiducial map. These results can be observed in Table 7.6, where for each image number the pose of milliAmpere, the camera to marker transform (**meas**) and the predicted camera to marker transform (**pred**) can be observed. It can be observed that when comparing **meas** and **pred** for # **img** 336, the differences

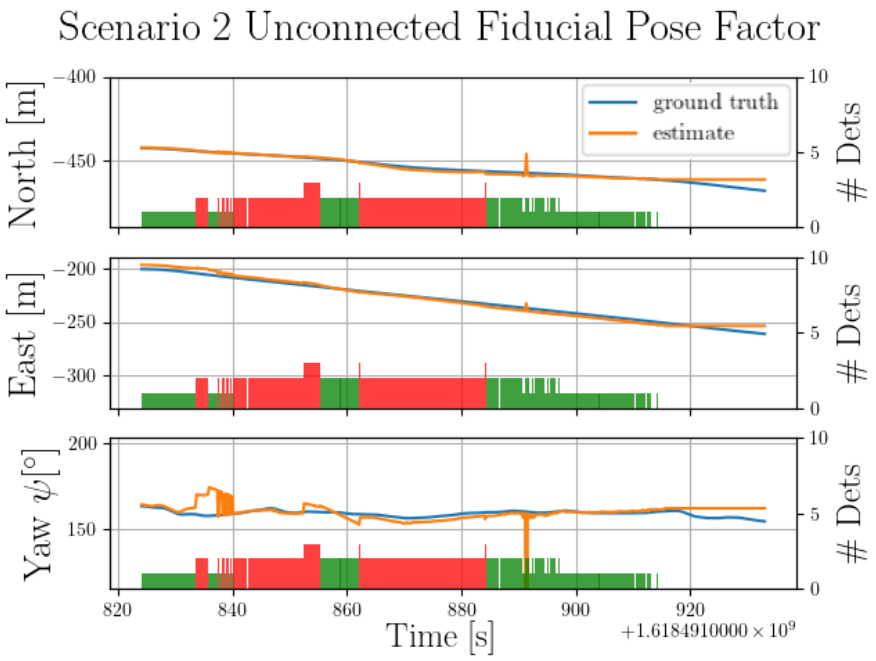


Figure 7.9: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

are minor, though the marker with id 1 is worse than id 0, as it is further away from the camera. However, for id 1 in # **337**, the difference is significantly different for both the second and third axis angles. The cause of this problem could not be found, though it was noted in [11] that the IPPE PnP method can, as other PnP methods, return multiple possible transforms for one set of inputs. However, the algorithm implemented in OpenCV and used in this thesis only produced one possible solution. As the factor graph was unconnected here, and the pose implementation of the fiducial factor was used, the transform was added to the graph weight by the same measurement noise as the transform for the other AprilTag marker in this image. This also meant that this problem did not persist in the estimation after this event, as can be seen from **pose** for # **img 338** in Table 7.6, the pose estimate for milliAmpere was back to a value similar to the **pose** at # **img 336**.

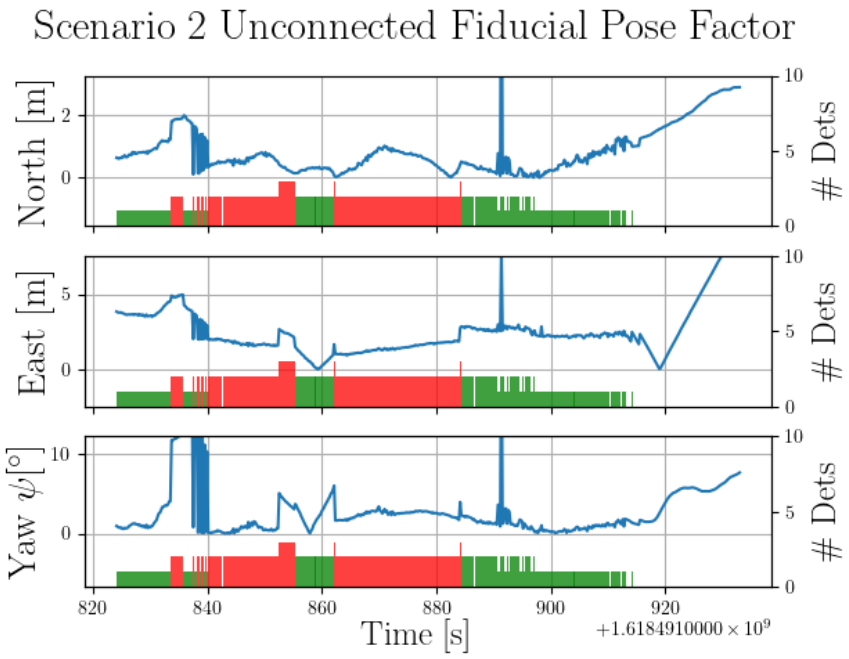


Figure 7.10: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.9. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.5: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	11	21	7.2	13	14	21
2. axis [°]	2.0	5.9	1.5	3.8	2.2	5.9
3. axis [°]	4.0	27	3.5	12	4.3	27
1. axis [m]	0.84	9.3	0.7	1.9	0.94	9.3
2. axis [m]	2.5	8.2	2.0	5.0	2.8	8.2
3. axis [m]	3.1	6.5	1.7	4.1	3.9	6.5

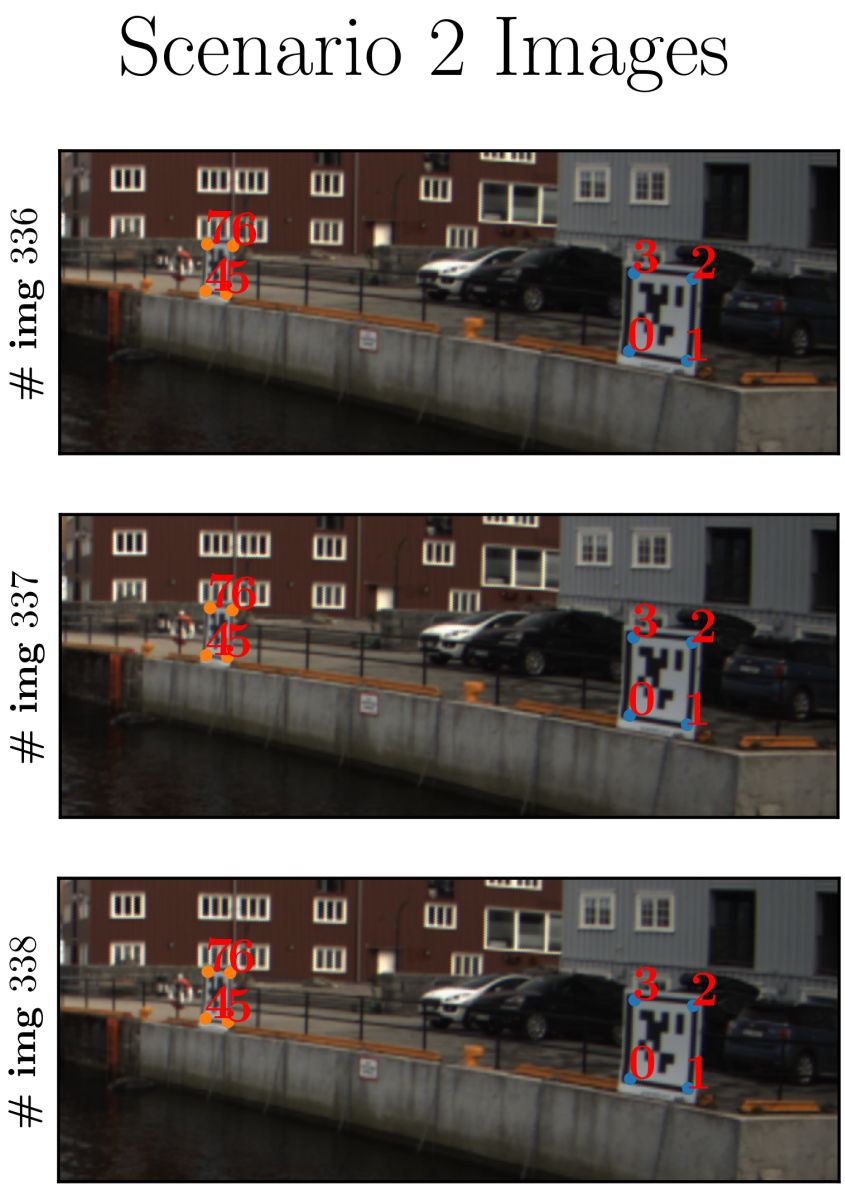


Figure 7.11: Image from the front left EO camera, zoomed in to highlight the AprilTag markers and detected corner points. Image taken before (**# img 336**), at (**# img 337**) and after (**# img 338**) the error spike event. The detected corners are visualised for all detected AprilTag markers. The marker at the right side of the images has id 0, and the marker at the left side of the images has id 1. Note that the corners are visualised as $4 * \text{id} + i$, where $i = 0$ is the lower left corner, $i = 1$ is the lower right corner, $i = 2$ is the upper right corner and $i = 3$ is the upper left corner.

Table 7.6: Data for image numbers (**# img**) 336 to 338. **pose** is the pose of milliAmpere estimated at that image number, given in the NED frame. **meas** is the estimated pose of the given AprilTag id using the IPPE PnP method, given in the camera frame. **pred** is the predicted pose of the given AprilTag id based on the predicted milliAmpere pose and the predicted AprilTag marker pose, given in the camera frame. The axes given in degrees [°] are the angles around the given axis, while the axes given in metres [m] are the translations along the given axis, all within the coordinate systems described here.

# img		axis	1. [°]	2. [°]	3. [°]	1. [m]	2. [m]	3. [m]
336	**pose**		-13.39	-0.86	159.29	-458.49	-239.48	-7.65
	id 0	**meas**	-133.57	65.78	39.51	7.97	-3.20	17.42
		pred	-132.93	65.78	39.81	7.95	-3.17	17.30
	id 1	**meas**	-150.75	61.23	29.98	2.83	-6.66	29.89
		pred	-131.24	64.08	43.91	3.32	-6.52	29.71
337	**pose**		-12.51	-4.62	-172.65	-446.11	-232.70	-8.13
	id 0	**meas**	-133.21	65.89	39.55	7.97	-3.21	17.51
		pred	-140.10	63.97	35.94	8.00	-3.29	17.56
	id 1	**meas**	-158.01	-55.39	-14.43	2.87	-6.68	30.08
		pred	-138.29	62.50	40.24	2.81	-6.58	29.76
338	**pose**		-13.42	-2.21	158.79	-458.65	-239.71	-8.00
	id 0	**meas**	-133.09	65.87	39.69	7.97	-3.22	17.62
		pred	-156.17	39.26	12.76	12.25	-3.61	18.06
	id 1	**meas**	-146.71	62.68	31.81	2.88	-6.64	30.05
		pred	-156.36	38.42	15.70	1.70	-5.67	26.51

7.2.2 Connected Fiducial Pose Factor

The estimated and ground truth 2D pose for milliAmpere based upon the connected fiducial pose factor are shown in Figure 7.12, with the RSE shown in Figure 7.13. One detail of note in these plots is that the error spike event discussed above is smoothed out with this method. While the same faulty pose measurement happens, the method is better able to handle it, as the motion between the poses are constrained by the preintegration. The performance measures are listed in Table 7.7, and the performance measures are generally improved compared to the unconnected fiducial pose factor method.

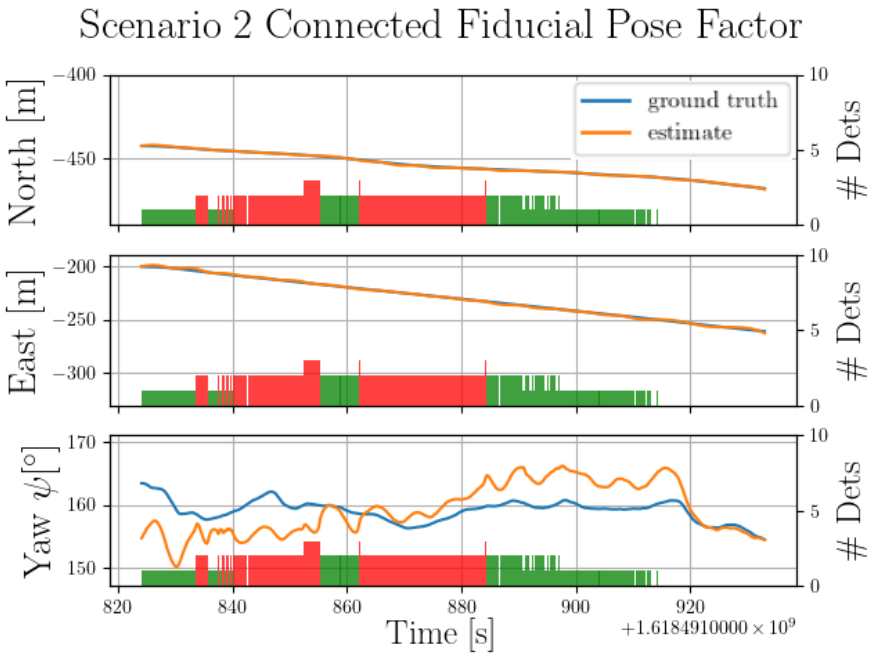


Figure 7.12: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

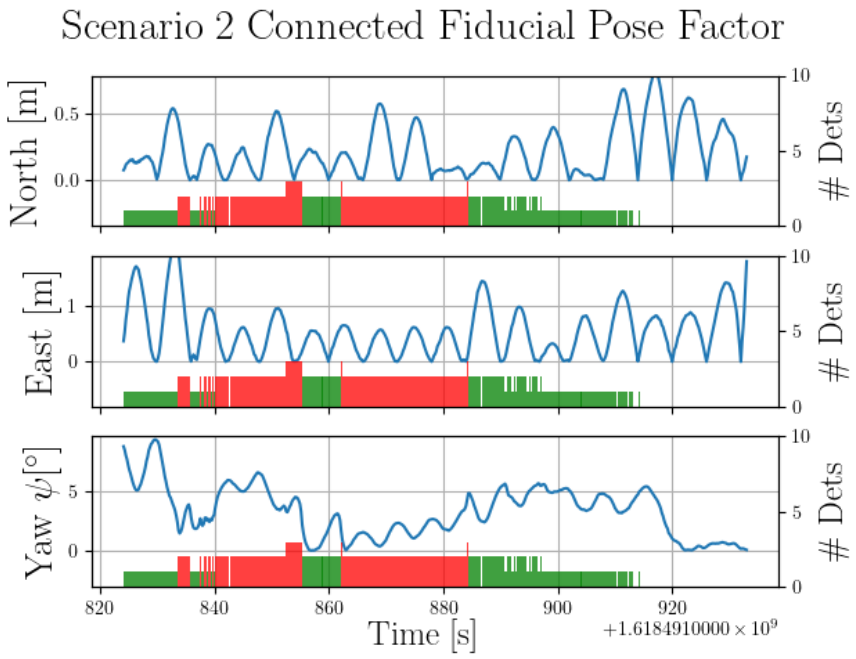


Figure 7.13: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.12. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.7: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	6.2	11	5.0	8.6	7.1	11
2. axis [°]	5.4	8.8	4.9	7.6	5.9	8.8
3. axis [°]	4.1	9.3	3.4	6.5	4.6	9.3
1. axis [m]	0.24	0.69	0.25	0.58	0.24	0.69
2. axis [m]	0.69	2.2	0.52	2.0	0.81	2.2
3. axis [m]	0.2	0.66	0.18	0.64	0.21	0.66

7.2.3 Unconnected Fiducial Projection Factor

The estimated and ground truth 2D pose can be seen in Figure 7.14 and the RSE plot can be seen in Figure 7.15, and it can be seen that the error explodes at the final point before crashing. It can initially be noted that the unconnected fiducial projection factor for this scenario was pretty unstable, as it kept crashing instantly once multiple cameras were used. The performance measures calculated for this method are listed in Table 7.8. However, the results for the detections by multiple cameras are not very useful because the crashing significantly increases the error. The results for the single cameras show that the method is reasonable, even though it is worse than the unconnected fiducial pose factor.

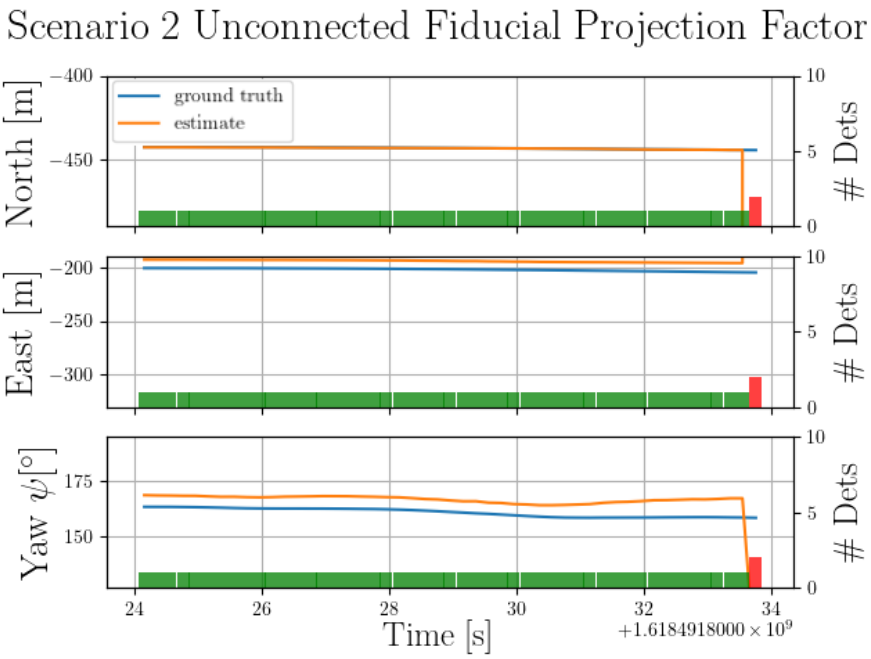


Figure 7.14: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

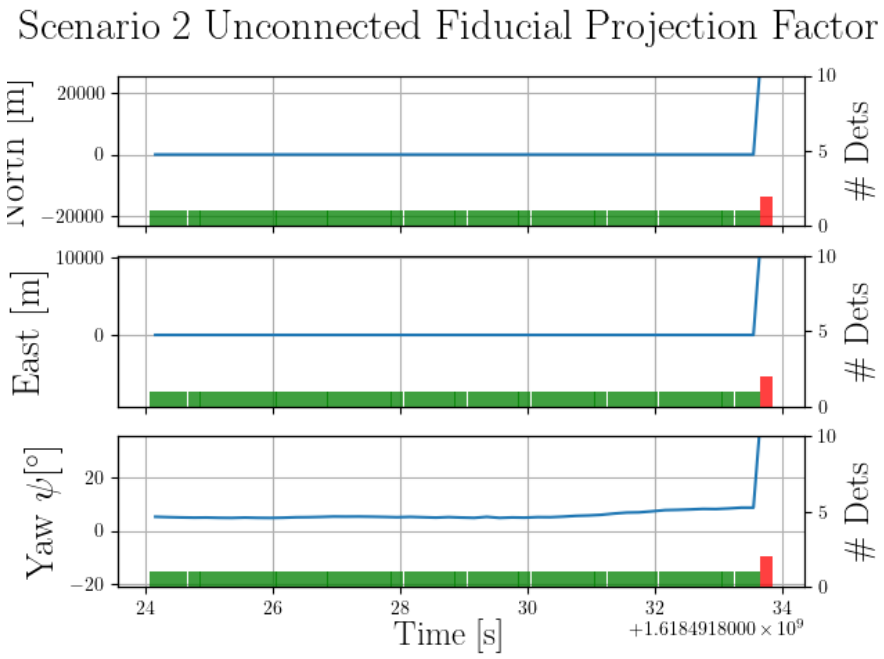


Figure 7.15: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.14. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.8: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	13	14	11	11	13	14
2. axis [°]	13	55	55	55	10	12
3. axis [°]	12	72	72	72	5.9	8.7
1. axis [m]	8.2×10^3	5.7×10^4	5.7×10^4	5.7×10^4	2.8	3.2
2. axis [m]	3.3×10^3	2.3×10^4	2.3×10^4	2.3×10^4	7.5	8.0
3. axis [m]	5.1×10^3	3.6×10^4	3.6×10^4	3.6×10^4	7.0	7.3

7.2.4 Connected Fiducial Projection Factor

The estimated and ground truth 2D pose of milliAmpere is shown in Figure 7.16, and the RSE is shown in Figure 7.17. When connecting the graph, the method was able to get past the area which caused the crash for the unconnected fiducial projection factor method, though as can be observed from the figures, there's still some larger errors early in the scenario. The performance measures are listed in Table 7.9. They were generally worse than both the connected and unconnected fiducial pose methods tested on this scenario, and were similar when comparing the measures over all detections to those over multiple cameras and single camera detection situations.

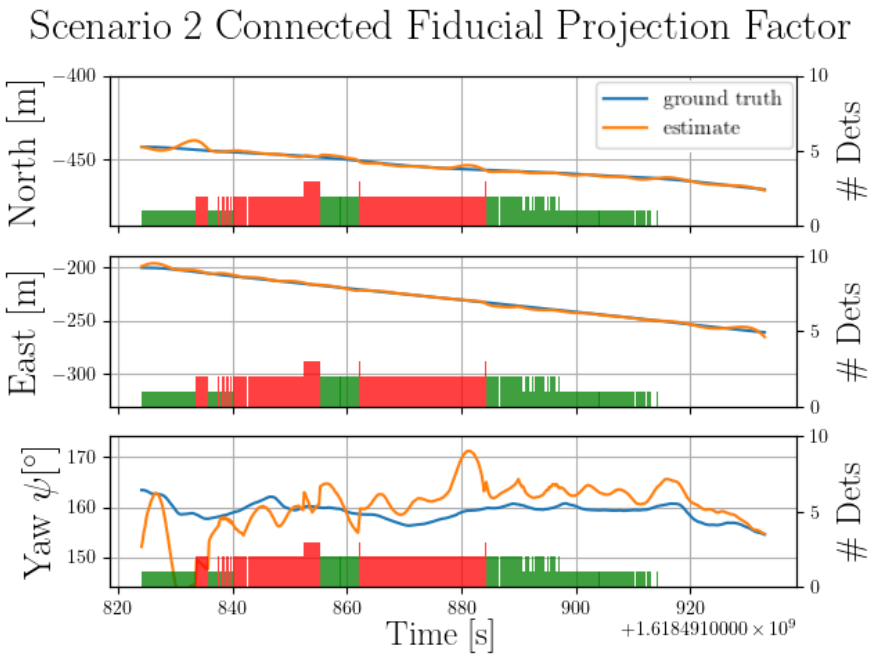


Figure 7.16: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

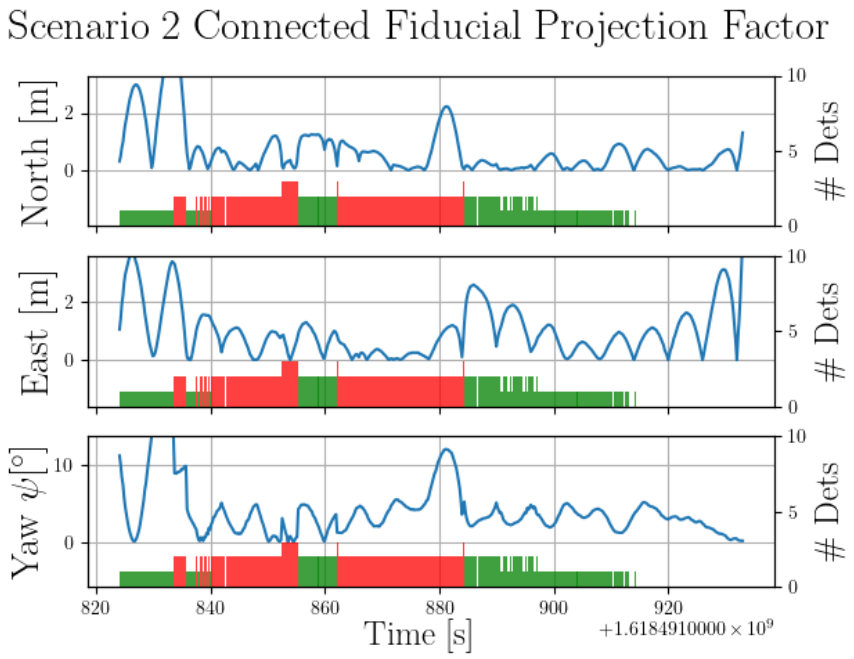


Figure 7.17: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.16. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.9: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	14	22	9.0	18	17	22
2. axis [°]	20	37	25	37	15	22
3. axis [°]	5.5	16	5.3	12	5.7	16
1. axis [m]	1.2	4.7	1.1	4.6	1.3	4.7
2. axis [m]	1.3	3.7	0.84	3.3	1.5	3.7
3. axis [m]	2.2	4.5	2.2	4.5	2.1	4.5

7.3 Scenario 3

This section presents the results of applying the fiducial SLAM method to scenario 3, described in Section 6.2.3. Scenario 3 was similar to scenario 2, milliAmpere was moving diagonally north-east up the canal facing the markers, around half way across the canal. This distance was the primary difficulty for the fiducial SLAM method to handle for this scenario, as there are few detections. The scenario can be divided into two separate time ranges, the first between 180 and 220 seconds, where primarily only one camera observed the markers at the same time, and the frequency of detection was not very consistent. The second time range would be between 260 and 290 seconds, where multiple cameras detect the markers simultaneously. Here, the frequency of the detections was also more consistent compared to the first time range.

7.3.1 Unconnected Fiducial Pose Factor

The estimated and ground truth 2D pose for milliAmpere using the unconnected fiducial pose factor method is shown in Figure 7.18, with the RSE shown in Figure 7.19. As can be observed, the estimate was not very consistent, and included several error spikes for some of the detections, particularly for the first time range. Error spikes were still observed for the second time range, where the markers were detected by multiple cameras, though the error values were lower here than the first time range. In the plots one can also observe some offsets, especially for the last part of the second time range, where the error stabilise somewhat around a constant offset value. The performance measures are listed in Table 7.10. As observed above, the cases where multiple cameras were used for detection performed the best. The second axis, east, had the highest errors over all, with some of the errors in the first time range being over 50 m.

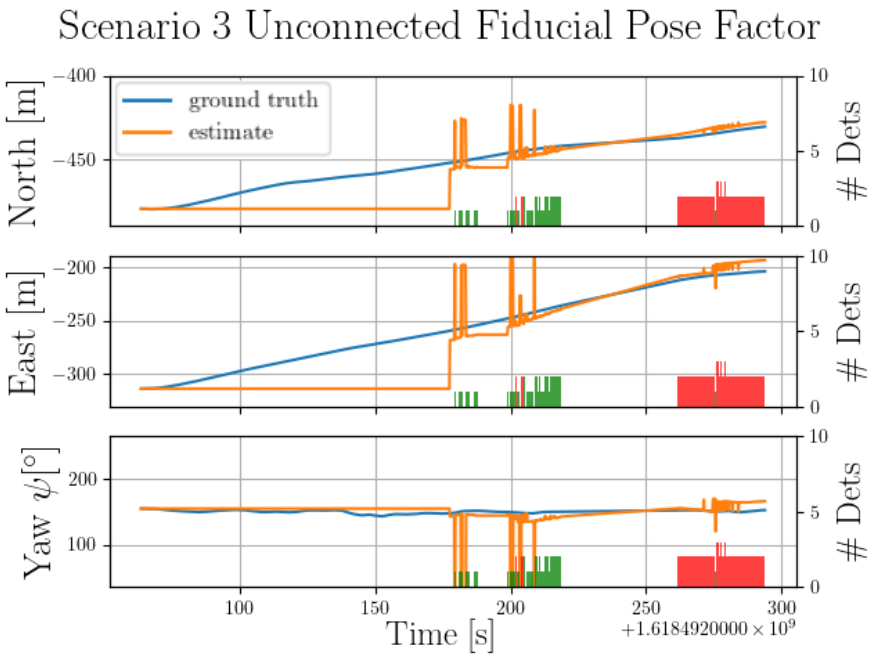


Figure 7.18: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.10: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	9.4	16	11	16	7.0	15
2. axis [°]	3.3	16	2.7	7.3	4.0	16
3. axis [°]	21	100	11	44	30	100
1. axis [m]	2.5	13	2.4	13	2.6	11
2. axis [m]	14	69	8.3	30	20	69
3. axis [m]	4.8	9.1	4.8	7.4	4.8	9.1

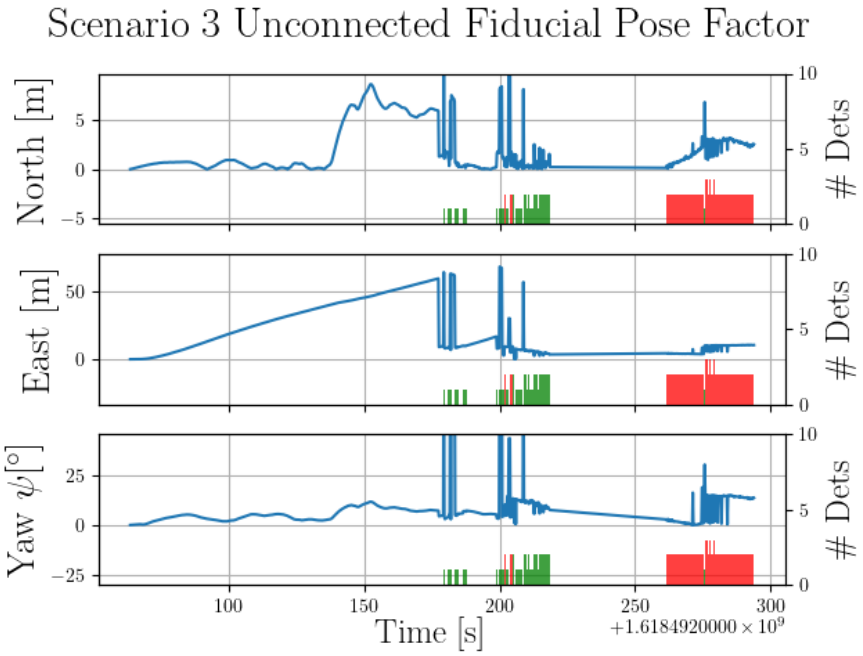


Figure 7.19: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.18. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

7.3.2 Connected Fiducial Pose Factor

The estimated and ground truth 2D pose for milliAmpere can be seen in Figure 7.20, and the RSE between these in Figure 7.21. The error spikes noted for the previous method introduced instability to this method, though the errors were smaller, especially for the the east axis. In Table 7.11, the performance measures are listed. These are overall improved from the unconnected fiducial pose method, especially for the third axis rotation and second axis translation.

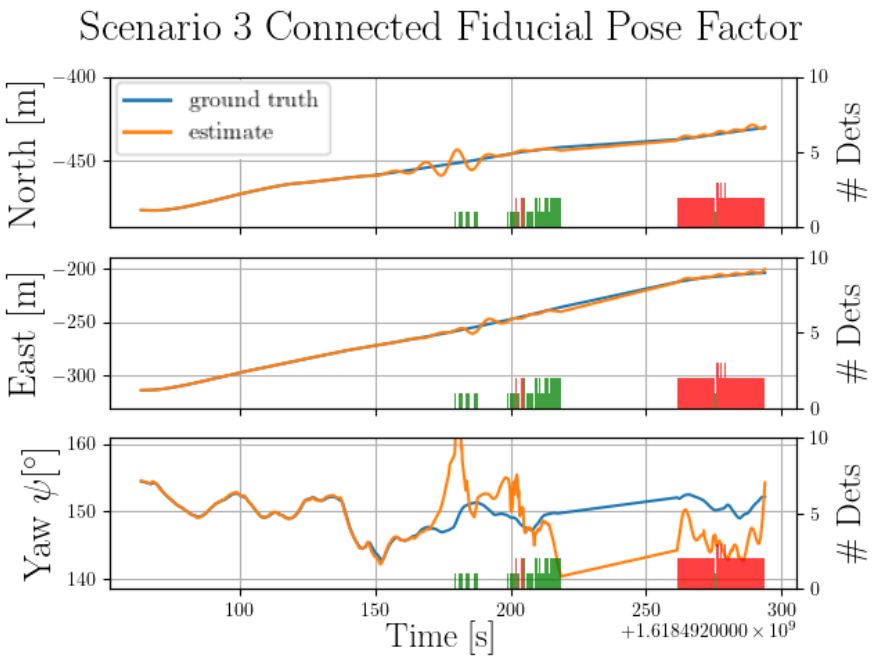


Figure 7.20: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

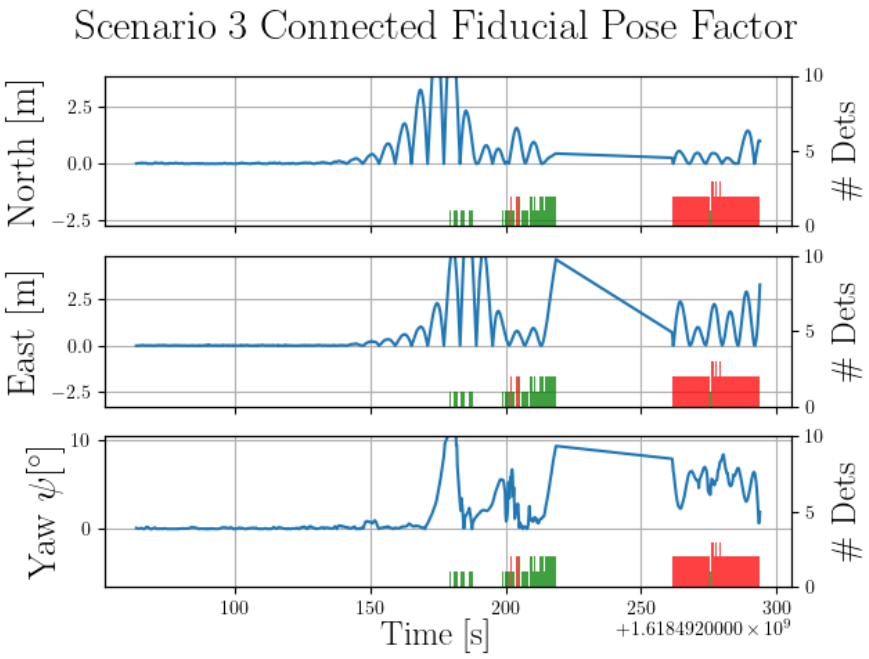


Figure 7.21: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.20. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.11: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	5.3	14	6.0	11	3.8	14
2. axis [°]	4.9	9.9	5.8	9.9	3.1	8.2
3. axis [°]	5.3	14	5.5	8.4	4.9	14
1. axis [m]	1.1	6.1	0.6	1.6	1.6	6.1
2. axis [m]	1.9	7.2	1.4	3.3	2.5	7.2
3. axis [m]	0.92	6.5	1.1	6.5	0.46	0.84

7.3.3 Unconnected Fiducial Projection Factor

The estimated and ground truth 2D pose are shown in Figure 7.22, and the RSE is shown in Figure 7.23. Similarly to testing the unconnected fiducial projection factor method for scenario 2 in Section 7.2.3, this method crashed. In this case, once the first fiducial marker was introduced, the method was unable to estimate the pose. The detection which caused the crash is also the same which caused a 50 m RSE for the East axis in Section 7.3.1, indicating that this is an unreliable measurement. The performance measures are listed in Table 7.12, tough only contain the details for the detection which caused the method to crash.

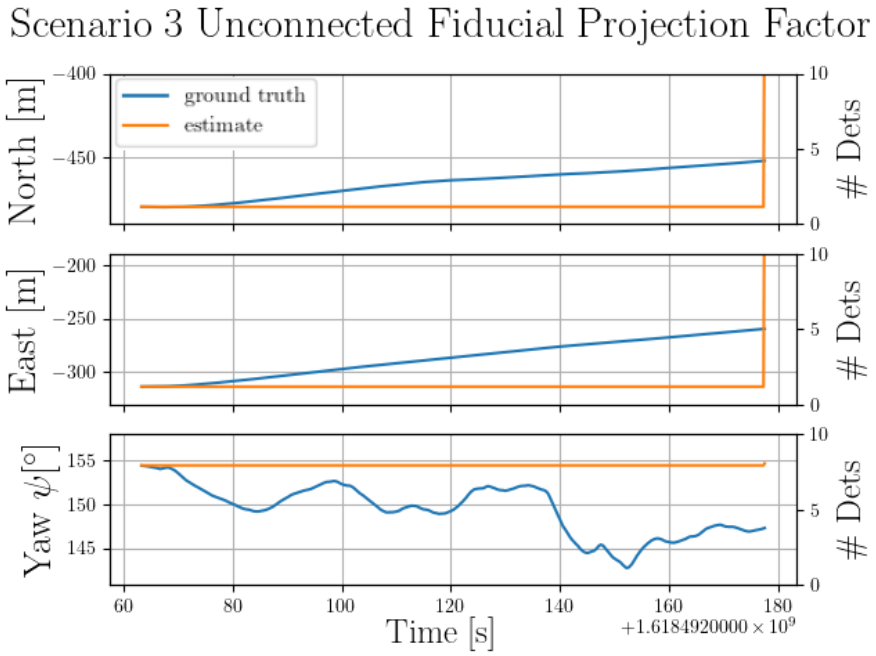


Figure 7.22: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

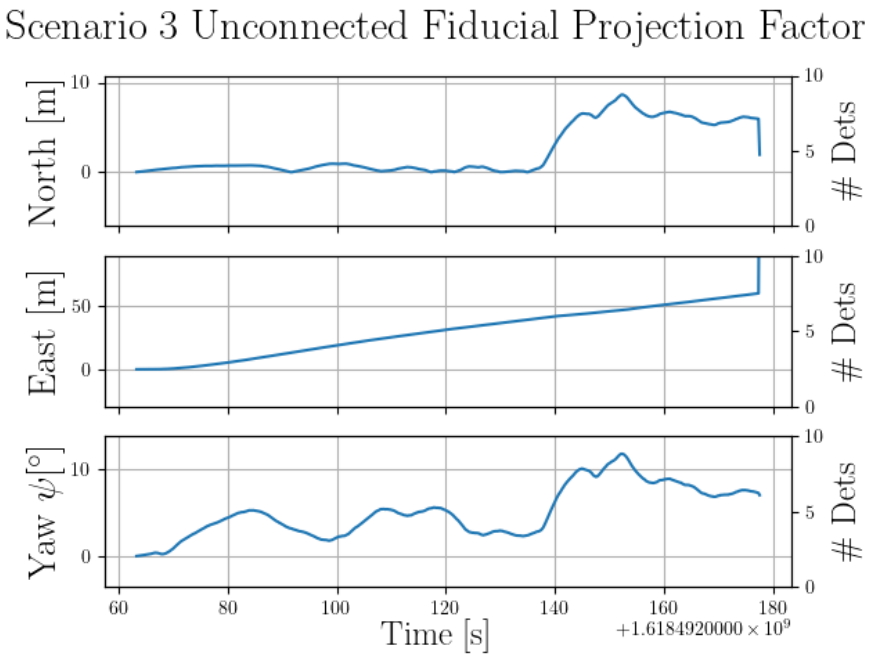


Figure 7.23: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.22. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.12: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	28	28	0.0	0.0	28	28
2. axis [°]	8.1	8.1	0.0	0.0	8.1	8.1
3. axis [°]	7.0	7.0	0.0	0.0	7.0	7.0
1. axis [m]	1.9	1.9	0.0	0.0	1.9	1.9
2. axis [m]	200	200	0.0	0.0	200	200
3. axis [m]	110	110	0.0	0.0	110	110

7.3.4 Connected Fiducial Projection Factor

The estimated and ground truth pose for milliAmpere are shown in Figure 7.24, and the RSE is shown in Figure 7.25. This method was able to complete most of the first time range, though crashed at the end. It can be observed that there was some instability in the middle of the first time range, where the input went from infrequent to frequent detections. The method crashed for a pose where multiple cameras detected the markers, which is especially noticeable in the listed performance measures, see Table 7.13, where both the performance for all estimates and specifically for the estimates using multiple cameras were very large due to the crash. However, the performance where a single camera was utilised for estimation was very small, especially for the translation, which outperforms the connected fiducial pose factor method for the North and East axes.

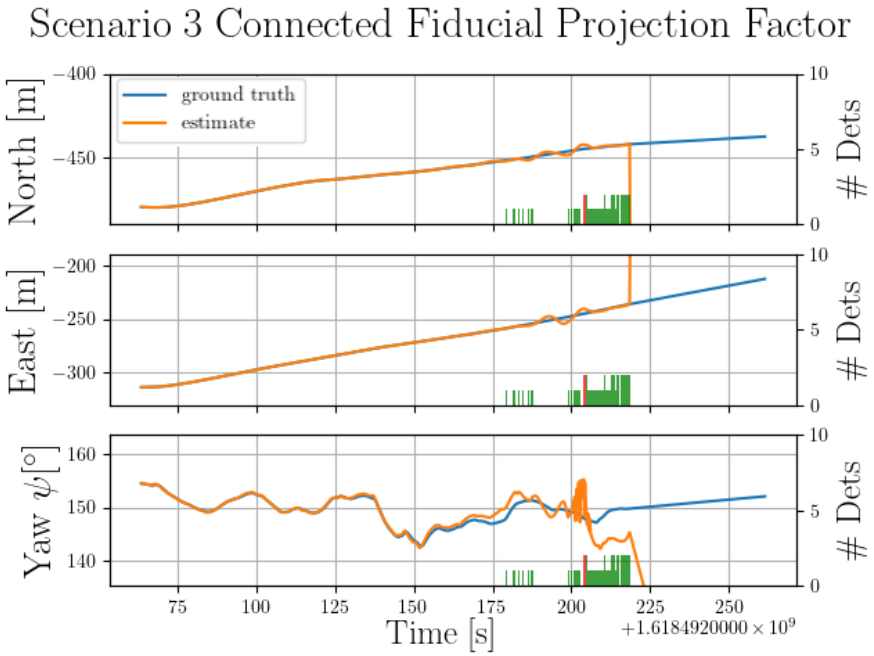


Figure 7.24: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

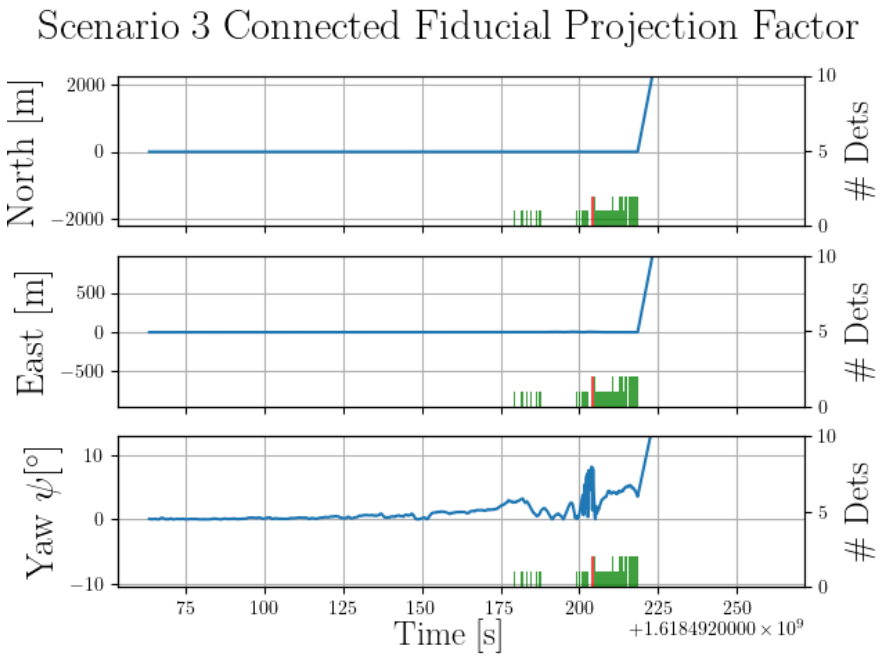


Figure 7.25: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.24. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.13: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	16	150	61	150	8.4	17
2. axis [°]	21	40	26	40	20	25
3. axis [°]	10	100	42	100	3.8	7.6
1. axis [m]	2×10^3	2.1×10^4	8.5×10^3	2.1×10^4	0.45	1.1
2. axis [m]	84	8.9×10^3	3.6×10^3	8.9×10^3	1.6	5.2
3. axis [m]	74	7.8×10^3	3.2×10^3	7.8×10^3	0.87	2.4

7.4 Scenario 4

There are no results shown for this scenario, as no detections were added to the back-end. As noted in Section 6.2.4, in this scenario, milliAmpere is too far away from the AprilTag markers. However, this is still a significant result, as it shows both that the detector is reliable and that the conditional removal steps are working as intended and does not add false or unreliable fiducial markers into the back-end. Furthermore, this also implies that the markers were close to the correct size for the operating area, as they can be detected in scenario 3, where milliAmpere is operating around the centre of the canal.

7.5 Scenario 5

Scenario 5 was created as a larger dataset for general SLAM methods, where a significant portion of the trajectory takes the ferry near the fiducial markers. The orientation of the ferry is also not as controlled as in the other scenarios, which means that several of the cameras of the 360 EO-IR Sensor Rig must be used to detect all the possible fiducial markers in this dataset. Most of the detections were in the time range between 625 and 670 seconds, though some infrequent detections appeared until around 700 seconds. The scenario itself was described in Section 6.2.5.

7.5.1 Unconnected Fiducial Pose Factor

The estimated and ground truth for the 2D pose can be seen in Figure 7.26, and the RSE between these can be seen in Figure 7.27. The performance measures are listed in Table 7.14. These show that the method obtained decent results, especially for the estimates using multiple cameras, though the other were most likely influenced by some of the poorer detections at the end of the primary detection time range for this scenario.

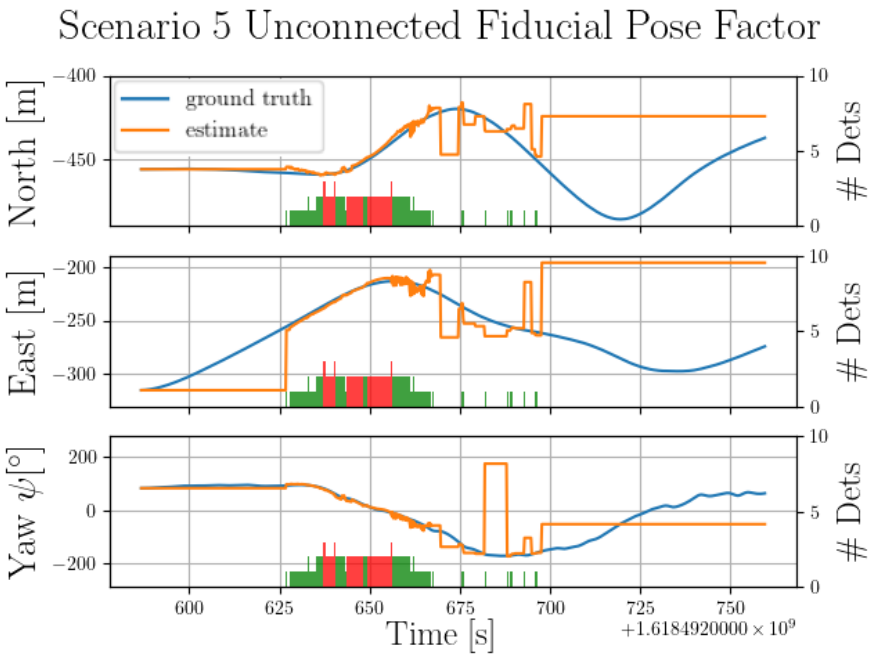


Figure 7.26: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.14: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	8.2	28	6.0	12	9.2	28
2. axis [°]	9.0	21	5.9	13	10	21
3. axis [°]	12	100	4.3	9.4	15	100
1. axis [m]	6.3	56	1.6	3.0	7.7	56
2. axis [m]	5.7	45	2.0	4.9	6.9	45
3. axis [m]	6.1	18	4.1	6.5	6.9	18

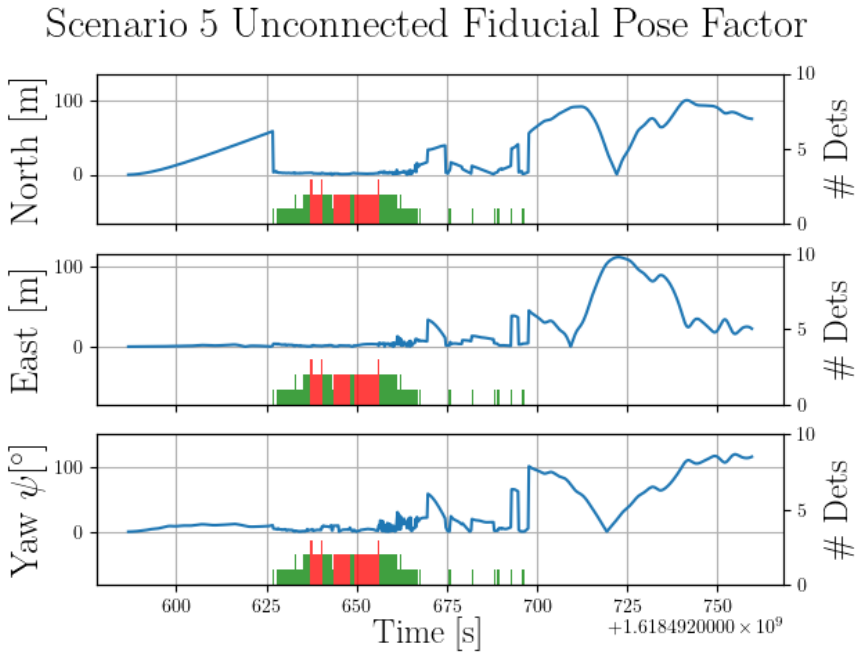


Figure 7.27: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.26. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

7.5.2 Connected Fiducial Pose Factor

The estimated and ground truth for the 2D pose is shown in Figure 7.28, and the RSE between these is shown in Figure 7.29. No major error spikes can be observed for this method, unlike in the unconnected fiducial pose factor method. The method was not always stable, as the error increased especially around the start and end of the primary time range for this scenario. The performance measures are listed in Table 7.15, which are all improved from the unconnected fiducial pose factor method.

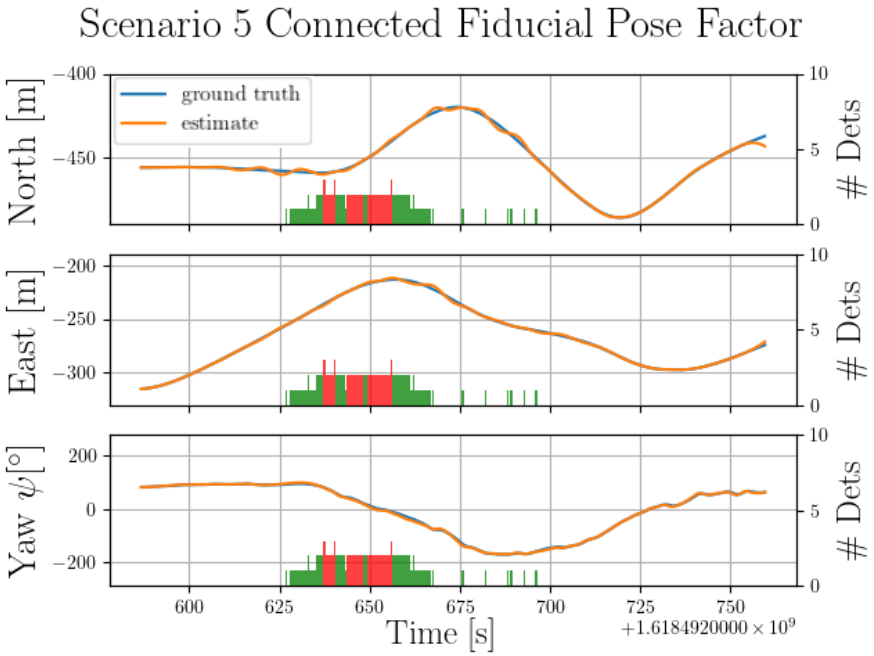


Figure 7.28: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

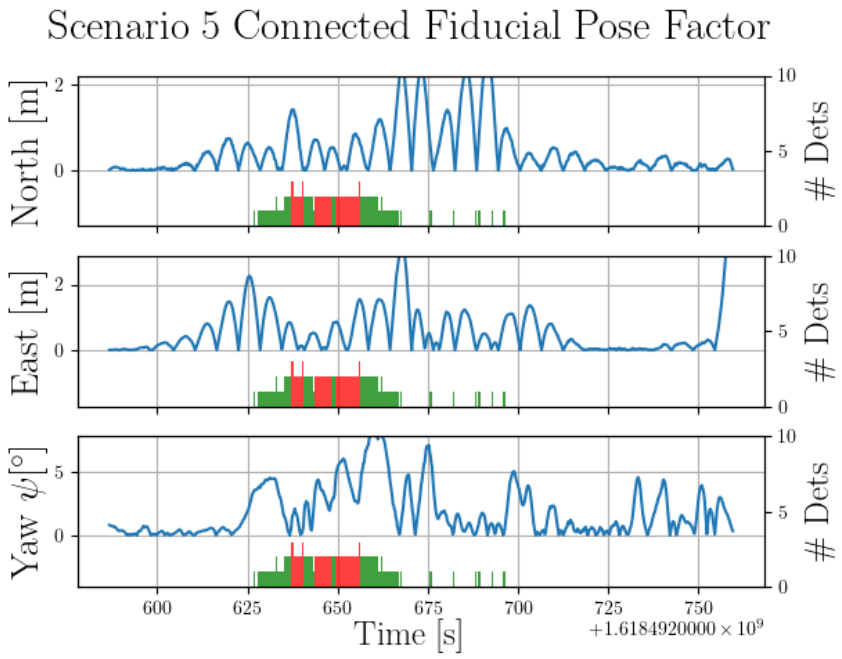


Figure 7.29: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.28. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.15: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	9.4	14	10	14	8.9	14
2. axis [°]	3.7	7.8	5.0	7.8	2.7	7.2
3. axis [°]	4.3	9.0	3.7	6.0	4.7	9.0
1. axis [m]	0.75	2.4	0.64	1.4	0.81	2.4
2. axis [m]	0.97	3.1	0.61	1.6	1.1	3.1
3. axis [m]	0.61	1.3	0.81	1.3	0.47	1.2

7.5.3 Unconnected Fiducial Projection Factor

The unconnected fiducial projection factor method crashed again for this scenario. However, the estimated and ground truth 2D pose is shown in Figure 7.30, the RSE is shown in Figure 7.31, and the performance measures are listed in Table 7.16.

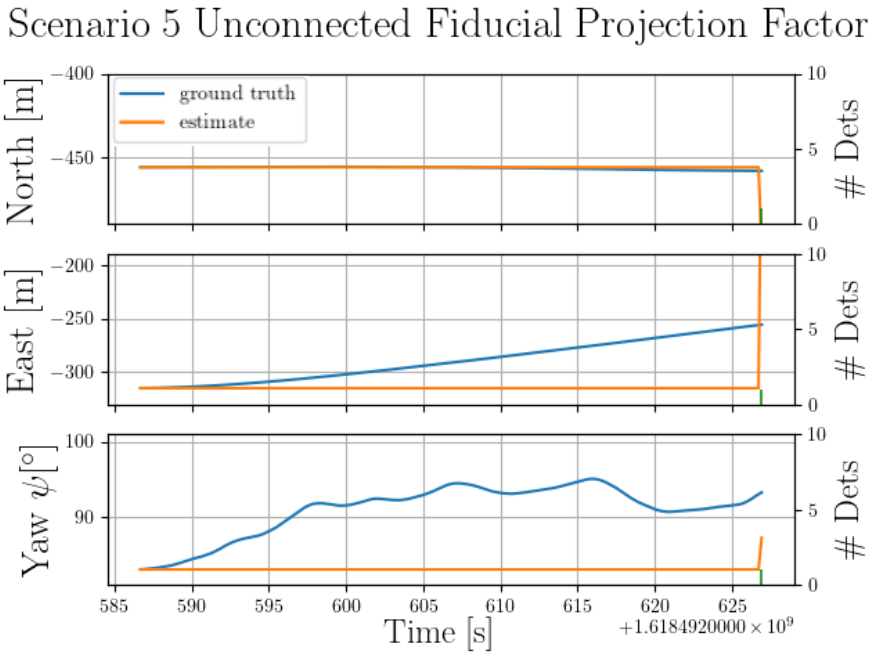


Figure 7.30: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

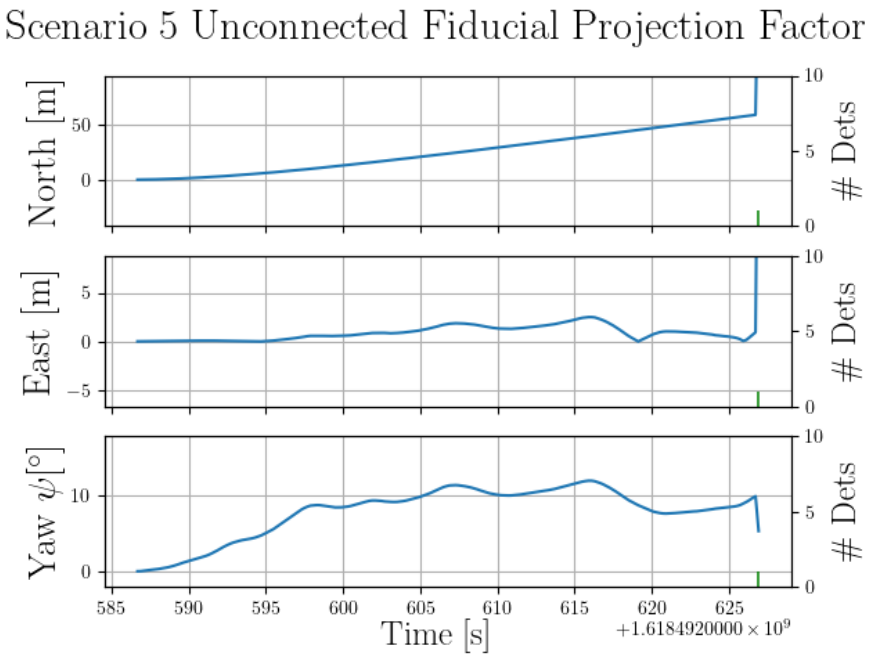


Figure 7.31: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.30. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.16: Measures of the overall performance of the fiducial SLAM method using RMSE (**Mean**) and maximum RSE (**Max**). **All**: Estimates where any camera detects any fiducial marker. **Multi**: Estimates where multiple cameras detected any fiducial marker. **Single**: Estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	2.0	2.0	0.0	0.0	2.0	2.0
2. axis [°]	27	27	0.0	0.0	27	27
3. axis [°]	5.3	5.3	0.0	0.0	5.3	5.3
1. axis [m]	220	220	0.0	0.0	220	220
2. axis [m]	36	36	0.0	0.0	36	36
3. axis [m]	97	97	0.0	0.0	97	97

7.5.4 Connected Fiducial Projection Factor

The estimated and ground truth 2D pose can be observed in Figure 7.32, and the RSE can be seen in Figure 7.33. Similar error spikes to those observed for the connected fiducial pose factor method can be seen in these figures. These error spikes seem similar to those observed for the connected pose factor method, though the error appeared greater for the connected projection factor method. The performance measures are listed in Table 7.17, which were slightly outperformed by the connected fiducial pose factor. Especially the maximum RSE values were worse compared to the connected fiducial pose factor method.

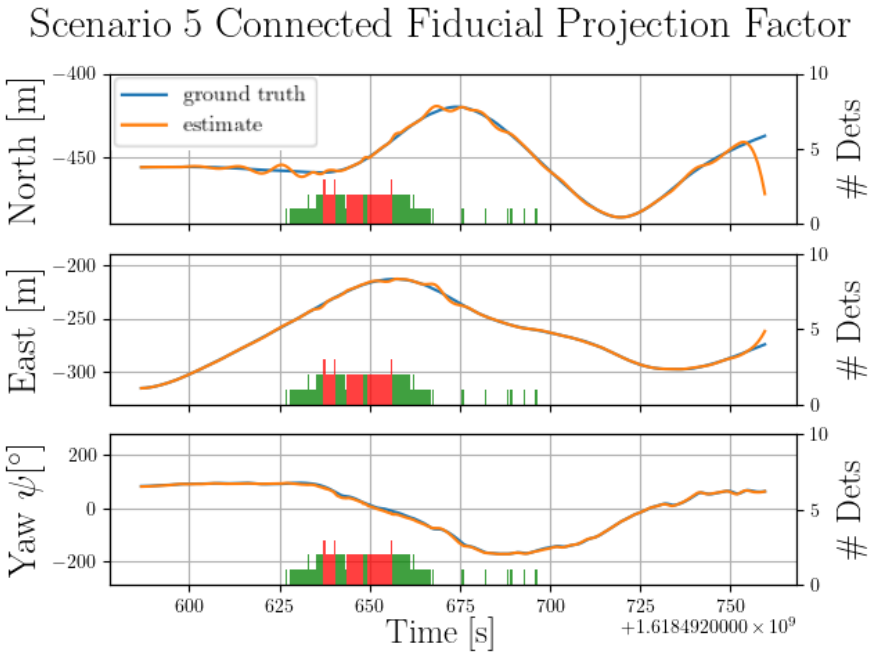


Figure 7.32: Left axis. Orange line: The North, East and Yaw estimate by the Fiducial SLAM method. Blue line: Ground truth. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

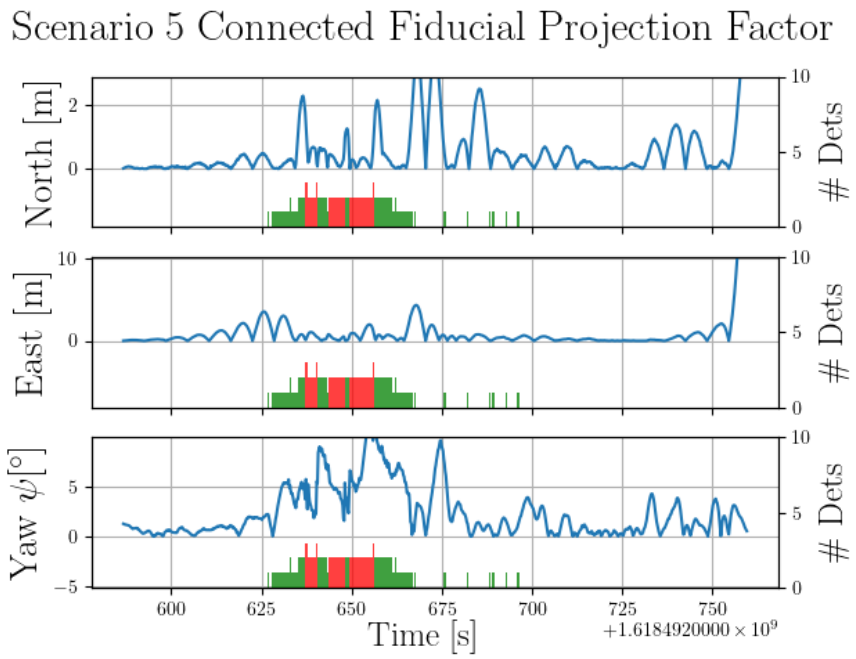


Figure 7.33: Left axis, blue line: The North, East and Yaw error, difference between the estimated and ground truth trajectories shown in Figure 7.32. Right axis, red and green bars: The number of detections for each time step. Green means only one camera observes any marker, red means multiple cameras observe any marker. Time is given in Unix epoch time.

Table 7.17: Measures of the overall performance of the fiducial SLAM method. **All**: (**Mean**) values when looking at estimates where any camera detects any fiducial marker. **Multi**: RMSE (**Mean**) and max RSE (**Max**) values when looking at estimates where multiple cameras detected any fiducial marker. **Single**: RMSE (**Mean**) and max RSE (**Max**) values when looking at estimates where only a single camera detected any fiducial marker. Given for all three rotational and translational axes for the error pose, given in the NED coordinate system.

	All		**Multi**		**Single**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**
1. axis [°]	15	30	17	30	13	24
2. axis [°]	18	27	19	27	17	24
3. axis [°]	5.9	12	6.4	12	5.7	12
1. axis [m]	0.84	3.6	0.44	1.4	0.99	3.6
2. axis [m]	1.2	4.3	0.92	2.0	1.4	4.3
3. axis [m]	1.7	3.3	1.9	3.3	1.5	3.0

Chapter 8

Discussion

Two different methods for adding fiducial markers to a factor graph were implemented and tested in this thesis, namely the fiducial pose factor and the fiducial projection factor. The fiducial projection factor might be the most logical to implement when first attempting to use fiducial markers in factor graphs, as it uses the detected corner pixels directly in the factor graph. This method allows the measurement uncertainty to be defined as pixel uncertainty, which directly connects to the sensor and allows an intuitive approach to tuning the factor. This factor was nontrivial to implement, as multiple pixels and world points would have to be connected for each measurement, and the fiducial marker geometry had to be implemented and constrained by the factor.

The factor was also more unstable than the fiducial pose factor, as the unconnected fiducial projection factor method crashed early for both scenario 2, 3 and 5. The crashes seemed to coincide with a large error in the initial estimate. As there are more variables and constraints involved in this method compared to the pose fiducial factor, the large initial error caused problems for this factor. While the large initial error was linked to the crashes in scenario 3 and 5, scenario 2 started with the fiducial markers in view, and was able to estimate the pose initially. However, the method crashed once multiple cameras were used for the estimation.

As can be observed from multiple of the scenarios and methods, certain detections introduced offsets in the error plots. This indicates that while the method can accurately estimate the pose of the ferry, the transform from body frame to the camera frames were offset, at least for some of the cameras. Because these transforms were measured by hand, and not estimated using optimisation methods, it was very likely that they were inaccurate, and the assumption that they were constant was wrong.

Compared to the fiducial projection factor, the fiducial pose factor was both more robust and simpler to implement. It relied upon a known and proven IPPE

PnP method to add the detected fiducial marker to the factor graph as a pose. The methods testing this factor also never ended up crashing the program. This was most likely because it is simpler for the iSAM2 backend to handle one well-defined pose factor compared to multiple pixel and world coordinate corners, and even in Section 7.2.1 where the method fails, a pose is estimated even though the measured pose doesn't make sense. While the IPPE PnP method might handle badly defined corners, or simply not return a pose, the measured corner pixels of the fiducial projection factor are inserted directly into the back-end with none of the elements which makes the IPPE PnP method robust.

This method had the disadvantage of defining a constant Gaussian measurement noise for all measured poses by the IPPE PnP method. This is not an accurate model for this measurement, as the detected corner points are projected onto the image. The projection involves normalising the point by the depth distance from the camera, then discretising the point to a pixel. Because of this process, the uncertainty of the pose measurement should scale with distance to the fiducial marker. As this was not implemented, this method was less reliable at a distance compared to the fiducial projection factor, which used pixel uncertainty for each detected corner.

Similarly to the fiducial projection factor methods, error offsets were observed for the fiducial pose factor methods, which again suggests that the camera transforms were incorrect. As noted above for the fiducial pose factor, these methods were more robust primarily because the pose representation allowed the offsets to not influence the estimation as much.

The use of multiple sensors, as tested in the connected fiducial pose and projection methods, resulted in better overall performance. As the motion between the estimated poses was constrained by motion models, the errors were significantly smaller, especially when combined with infrequent GNSS measurements. However, this also caused some instability, especially as the incorrect camera transforms meant that the fiducial marker detection measurements would be added to the optimiser incorrectly relative the vessel centre. As described above, these incorrect offsets caused the measurements to be inconsistent between the cameras, leading to the instabilities observed.

The 360 EO-IR Sensor Rig used to record the image data in the experiments had several problems, which impacted the results to some degree. It obtained 360° vision by using five individual cameras, all connected to a separate embedded system which used GNSS signals to trigger the cameras at the same time and timestamp them accurately. However, the cable necessary for this was plugged in to the onboard INS to provide a ground truth for the datasets. Therefore, a software synchronised method was used, which can result in time delays of up to 0.1 seconds, which can also cause offset measurements,

which may worsen some of the effects discussed above.

The camera calibration parameters were used by both the fiducial pose factor and the fiducial projection factor. These parameters are unique for all cameras, and were therefore estimated prior to the experiments in a lab setting using a rigid, A4-sized calibration checkerboard. Preferably, these parameters should be estimated in the same conditions as the data gathered, e.g. right before the data was gathered and at a similar distance to the objects being interacted with. As the lab environment used for estimating the parameters and the harbour environments used to gather data were very different, the calibration parameters used in this thesis were not as accurate as they should have been. However, the calibration process was difficult to execute for a few reasons. To handle to the distances between milliAmpere and the fiducial markers, the calibration should be done at the same distance, however this would mean a larger calibration checkerboard would have to be used, which would be more difficult to handle. Because all five cameras used were mounted on the roof of milliAmpere, it would also be difficult to move and rotate either the checkerboard or milliAmpere to fully cover the cameras' fields of view.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

In this thesis, a Fiducial SLAM method was developed based on the iSAM2 framework. Two factors, the fiducial pose factor and fiducial projection factor, were implemented and tested to use detected fiducial markers to connect the current system pose to the GNSS position of the markers. Testing on real data showed that both could be accurate methods for adding external global position data to a SLAM system. The fiducial pose factor, using the IPPE PnP method to estimate the transformation between the camera frame and the fiducial marker frame, was shown to generally be more accurate and stable than the fiducial projection factor. It was further shown that including other sensors, such as constraining the relative motion of the system using preintegration and slow and accurate GNSS measurements further improved the method.

The datasets where gathered using NTNU's autonomous ferry milliAmpere and used to test the accuracy of the Fiducial SLAM method. These datasets were primarily generated with a focus on the AprilTag fiducial markers placed near the canal, although scenario 5 represented a more general scenario. These datasets can be interesting for other methods, especially methods for ASVs operating in urban environments, and should be robust to the challenges one may face in these environments.

9.2 Future Work

There are many different ways of improving upon and taking the work done in this thesis further. One important way would be to improve the camera transforms used in this thesis, which can be done in several ways. The estimation of these transforms is a difficult task, especially as the cameras are mounted facing different directions. Another possibility is to estimate the camera pose online as part of the optimisation problem. Then this transform is assumed un-

known, which can be modelled in the factor graph.

Another interesting way of improving this work, would be to introduce other sensor information to the factor graph. As can be seen by introducing IMU and GNSS measurements in the factor graph in this work and by Skjellaug [59], the introduction of other measurements to a factor graph is both easy and improves the accuracy of the method. It would therefore be interesting to combine other sensors and methods to the same iSAM2 back-end, e.g. including the laser odometry described by Skjellaug et al. [60] or a VO method. The datasets gathered are very useful to test other methods which should be robust to the challenges faced for urban ASVs.

One possible advantage of using artificial landmarks in a multi-sensor SLAM system, is that they could be detected by multiple sensors, such as both the lidar and the cameras. The detection of fiducial markers using lidar has been shown to work by Huang et al. [33], which would be interesting to explore. Merging detections by camera and lidar could be done using factor graphs, and connecting these factors to common landmarks should increase the accuracy of the method further.

In this thesis, the detected fiducial markers were added directly to the back-end with some preprocessing done to remove obvious noise. A more robust front-end could be developed, where the measurements must be consistent between each other and the previous system pose, e.g. based on the RANdom SAmple Consensus (RANSAC) algorithm, see Szeliski [65, p. 491]. This could also possibly be done by using the IPPE PnP method in the front-end, to estimate the current pose and give an initial estimate to the back-end, which could use the fidcial projection factor. Another possible way to ensure that the measurements are consistent could be to optimise the current subgraph before sending it to the back-end, using an optimisation method such as Levenberg-Marquardt, and removing improbable measurements from the graph sent to the back-end. It would also most likely improve the system to introduce a similar front-end to the UcoSLAM method by Muñoz-Salinas and Medina-Carnicer [46], and using natural features together with the fiducial markers, and only use certain *keyframes* for the back-end optimisation. A more thorough system like this could ensure that the larger error spikes observed especially for the unconnected implementations of the Fiducial SLAM method.

Exploring different fiducial markers would also be interesting, as there are many different systems and variations in existence. The ArUcO and AprilTag fiducial markers have been discussed in this thesis, though many other markers and detectors have been implemented, such as TopoTag by Yu et al. [70] and ChromaTag by DeGol et al. [16] are interesting fiducial marker systems. There are also AprilTag families which could be interesting to explore, some of which

were noted in Section 3.6.2. For autonomous ferries, the tagCustom38h12 family is especially interesting, as it can be defined recursively. This means that from a distance, the large marker will be detected and can cover a large area further away from the docking area of the ferry. Once the ferry arrives near the docking area, the camera might be positioned such that the marker is too large for the camera. However, as this marker can be defined recursively, another marker can be placed in its centre, so that near the docking area, a smaller marker can be used to assist the docking process. Autonomous docking is an important and difficult research topic, and fiducial markers can be very useful tools for this.

It would also be interesting to introduce some form for online camera calibration method to the system, or use the datasets described here for this purpose. The use of the AprilTag fiducial markers for camera calibration was done by Richardson et al. [56], though in this case multiple markers placed on the same board, similar to a standard calibration checkerboard, was used. However, as the fiducial marker size is known, calibration methods could still be explored using the markers online. In the GTSAM library, the camera calibration can be added as an unknown variable and optimised for, so that the camera calibration could be a part of the general optimisation problem for a large multi-sensor SLAM method.

Multiple other factor graph libraries were described in Section 3.8.1. Exploring the use of some of these would be interesting, especially to compare the speed and accuracy of these methods compared to the GTSAM library. Because ASVs usually are not as limited by computing power as e.g. drones, the libraries could be explored in order to find the most suitable and accurate library.

Bibliography

[1] Mohammad Aqel, Mohammad Hamiruce Marhaban, M Iqbal Saripan and Napsiah Ismail. 'Review of visual odometry: types, approaches, challenges, and applications'. In: *SpringerPlus* 5 (Dec. 2016).

[2] Autoferry. *Autonomous all-electric passenger ferries for urban water transport (Autoferry)*. URL: https://www.ntnu.edu/autoferry (visited on 10/05/2021).

[3] Yaakov Bar-Shalom, X. Rong Li and Thiagalingam Kirubarajan. *Estimation with Applications To Tracking and Navigation*. John Wiley and Sons, Inc., 2001.

[4] Keshav Bimbraw. 'Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology'. In: *ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics, Proceedings* 1 (Jan. 2015), pp. 191–198.

[5] Jean R. S. Blair and Barry Peyton. 'An Introduction to Chordal Graphs and Clique Trees'. In: *Graph Theory and Sparse Matrix Computation*. Ed. by Alan George, John R. Gilbert and Joseph W. H. Liu. New York, NY: Springer New York, 1993, pp. 1–29.

[6] G. Bradski. 'The OpenCV Library'. In: *Dr. Dobb's Journal of Software Tools* (2000).

[7] Edmund Førland Brekke. *Fundamentals of Sensor Fusion*. 2020. URL: http://folk.ntnu.no/edmundfo/msc2020-2021/sf2020.pdf (visited on 22/08/2020).

[8] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid and J.J. Leonard. 'Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age'. In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[9] Ching-Yao Chan. 'Advancements, prospects, and impacts of automated driving systems'. In: *International Journal of Transportation Science and Technology* 6.3 (2017). Safer Road Infrastructure and Operation Management, pp. 208–216.

[10] Han-Pang Chiu, Xun S. Zhou, Luca Carlone, Frank Dellaert, Supun Samarasekera and Rakesh Kumar. 'Constrained optimal selection for multi-sensor robot navigation using plug-and-play factor graphs'. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 663–670.

[11] Toby Collins and Adrien Bartoli. 'Infinitesimal Plane-Based Pose Estimation'. In: 109.3 (Sept. 2014), pp. 252–286. ISSN: 0920-5691.

[12] Contributors and Packages. *Caesar.jl*. 2020. URL: https://github.com/JuliaRobotics/Caesar.jl.

[13] P. Corke. *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised, Extended And Updated Edition*. Springer Tracts in Advanced Robotics. Springer International Publishing, 2017. ISBN: 9783319544137. URL: https://books.google.no/books?id=d4EkDwAAQBAJ.

[14] Nicholas Dalhaug. 'Lidar-based Localization for Autonomous Ferry'. In: (June 2019). Master thesis Cybernetics and Robotics, NTNU. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2625739.

[15] Timothy A. Davis, John R. Gilbert, Stefan I. Larimore and Esmond G. Ng. 'A column approximate minimum degree ordering algorithm'. In: *ACM Transactions on Mathematical Software* 30.3 (2004), pp. 353–376.

[16] Joseph DeGol, Timothy Bretl and Derek Hoiem. *ChromaTag: A Colored Marker and Fast Detection Algorithm*. 2017. arXiv: 1708.02982 [cs.CV].

[17] F. Dellaert and M. Kaess. 'Factor Graphs for Robot Perception'. In: *Foundations and Trends in Robotics, FNT* 6.1-2 (Aug. 2017), pp. 1–139.

[18] Frank Dellaert. *Factor graphs and GTSAM: A hands-on introduction*. 2012.

[19] Egil Eide. *MilliAmpère - Norges første førerløse ferje*. URL: https://www.sdir.no/globalassets/sjofartsdirektoratet/fartoy-og-sjofolk---dokumenter/ulykker-og-sikkerhet/sjosikkerhetskonferansen-2018/sjosikkerhetskonferansen_2018_foredrag_1_25_egil_eide.pdf (visited on 18/06/2018).

[20] J. Engel, T. Schöps and D. Cremers. 'LSD-SLAM: Large-Scale Direct Monocular SLAM'. In: *European Conference on Computer Vision (ECCV)*. Sept. 2014.

[21] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. 'Efficient Graph-Based Image Segmentation'. In: *Int. J. Comput. Vision* 59.2 (Sept. 2004), pp. 167–181. ISSN: 0920-5691.

[22] D. Fernandez and A. Price. 'Visual odometry for an outdoor mobile robot'. In: *IEEE Conference on Robotics, Automation and Mechatronics, 2004.* Vol. 2. 2004, 816–821 vol.2.

[23] Christian Forster, Luca Carlone, Frank Dellaert and Davide Scaramuzza. 'On-Manifold Preintegration for Real-Time Visual–Inertial Odometry'. In: *IEEE Transactions on Robotics* 33.1 (Feb. 2017), pp. 1–21.

[24] Thor I. Fossen. *Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, 2011.

[25] Hallvard Laupsa Fosso. 'Visual SLAM and Localization for Autonomous Ferry'. Master thesis Cybernetics and Robotics, NTNU. June 2020.

[26] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas and M.J. Marín-Jiménez. 'Automatic generation and detection of highly reliable fiducial markers under occlusion'. In: *Pattern Recognition* 47.6 (2014), pp. 2280–2292.

[27] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas and R. Medina-Carnicer. 'Generation of fiducial marker dictionaries using mixed integer linear programming'. In: *Pattern Recognition* 51 (2016), pp. 481–491.

[28] Martin Eek Gerhardsen. 'Fiducial SLAM for Autonomous Ferry'. Project thesis Cybernetics and Robotics, NTNU. Dec. 2020.

[29] Abigail S. Golden and Roberta E. Weisbrod. 'Trends, Causal Analysis, and Recommendations from 14 Years of Ferry Accidents'. In: *Journal of Public Transportation* 19.1 (Mar. 2016), pp. 17–27.

[30] Trym Vegard Haavardsholm. 'A Handbook in Visual SLAM'. 2020.

[31] Inger Berge Hagen. *Full scale protorype launch*. 2021. URL: https://www.ntnu.edu/web/autoferry/news/-/blogs/full-scale-prototype-launch-?_com_liferay_blogs_web_portlet_BlogsPortlet_redirect=https%3A%2F%2Fwww.ntnu.edu%2Fweb%2Fautoferry%2Fnews%3Fp_p_id%3Dcom_liferay_blogs_web_portlet_BlogsPortlet%26p_p_lifecycle%3D0%26p_p_state%3Dnormal%26p_p_mode%3Dview%26_com_liferay_blogs_web_portlet_BlogsPortlet_cur%3D1%26_com_liferay_blogs_web_portlet_BlogsPortlet_delta%3D20 (visited on 12/05/2021).

[32] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. New York, NY, USA: Cambridge University Press, 2003. ISBN: 0521540518.

[33] Jiunn-Kai Huang, Shoutian Wang, Maani Ghaffari and Jessy W. Grizzle. 'LiDARTag: A Real-Time Fiducial Tag System for Point Clouds'. In: *IEEE Robotics and Automation Letters* 6.3 (July 2021), pp. 4875–4882.

[34] J. D. Hunter. 'Matplotlib: A 2D graphics environment'. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[35] Viorela Ila, Lukas Polok, Marek Solony and Pavel Svoboda. *Highly Efficient Compact Pose SLAM with SLAM++*. 2016. arXiv: 1608.03037 [cs.RO].

[36] Eric Jones, Travis Oliphant, Pearu Peterson et al. *SciPy: Open source scientific tools for Python*. 2001. URL: http://www.scipy.org/.

[37] M. Kaess, A. Ranganathan and F. Dellaert. 'iSAM: Incremental Smoothing and Mapping'. In: *IEEE Trans. on Robotics (TRO)* 24.6 (Dec. 2008), pp. 1365–1378.

[38] Michael Kaess, Viorela Ila, Richard Roberts and Frank Dellaert. 'The Bayes Tree: An Algorithmic Foundation for Probabilistic Robot Mapping'. In: vol. 68. Jan. 2010, pp. 157–173. ISBN: 978-3-642-17451-3.

[39] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard and Frank Dellaert. 'iSAM2: Incremental Smoothing and Mapping with Fluid Relinearization and Incremental Variable Reordering'. In: vol. 31. July 2011, pp. 3281–3288.

[40] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard and Frank Dellaert. 'iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree'. In: *International Journal of Robotic Research - IJRR* 31 (May 2012), pp. 216–235.

[41] C. F. F. Karney. *GeographicLib, Version 1.51*. 2019. URL: `https://geographiclib.sourceforge.io/1.51` (visited on 14/05/2021).

[42] H. Kato and M. Billinghurst. 'Marker tracking and HMD calibration for a video-based augmented reality conferencing system'. In: *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*. 1999, pp. 85–94.

[43] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige and Wolfram Burgard. 'G2o: A general framework for graph optimization'. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3607–3613.

[44] Todd Lupton and Salah Sukkarieh. 'Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions'. In: *IEEE Transactions on Robotics* 28.1 (2012), pp. 61–76.

[45] Justin E. Manley. 'Unmanned surface vehicles, 15 years of development'. In: *OCEANS 2008*. 2008, pp. 1–4.

[46] Rafael Muñoz-Salinas and R. Medina-Carnicer. 'UcoSLAM: Simultaneous Localization and Mapping by Fusion of KeyPoints and Squared Planar Markers'. In: *Pattern Recognition* (2020), p. 107193.

[47] Raúl Mur-Artal, J. M. M. Montiel and Juan D. Tardós. 'ORB-SLAM: a Versatile and Accurate Monocular SLAM System'. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[48] Marius Strand Ødven. 'Lidar-Based SLAM for Autonomous Ferry'. In: (Jan. 2019). Master thesis Cybernetics and Robotics, NTNU. URL: `https://folk.ntnu.no/edmundfo/msc2019-2020/MasterFinalReducedMarius.pdf`.

[49] Lauro Ojeda and Johann Borenstein. 'Methods for the Reduction of Odometry Errors in Over-Constrained Mobile Robots'. In: *Auton. Robots* 16 (May 2004), pp. 273–286.

[50] Edwin Olson. *AprilTags Visual Fiducial System*. 2010. URL: https://april.eecs.umich.edu/software/apriltag (visited on 14/05/2021).

[51] Edwin Olson. 'AprilTag: A robust and flexible visual fiducial system'. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[52] OpenStreetMap contributors. *Planet dump retrieved from https://planet.osm.org*. https://www.openstreetmap.org. 2017.

[53] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[54] Bernd Pfrommer and Kostas Daniilidis. *TagSLAM: Robust SLAM with Fiducial Markers*. 2019. arXiv: 1910.00679 [cs.RO].

[55] J. Rekimoto. 'Matrix: a realtime object identification and registration method for augmented reality'. In: *Proceedings. 3rd Asia Pacific Computer Human Interaction (Cat. No.98EX110)*. 1998, pp. 63–68.

[56] Andrew Richardson, Johannes Strom and Edwin Olson. 'AprilCal: Assisted and repeatable camera calibration'. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2013.

[57] Francisco J. Romero-Ramirez, Rafael Muñoz-Salinas and Rafael Medina-Carnicer. 'Speeded up detection of squared fiducial markers'. In: *Image and Vision Computing* 76 (2018), pp. 38–47.

[58] Marcelo J. Segura, Fernando A. Auat Cheein, Juan M. Toibero, Vicente Mut and Ricardo Carelli. 'Ultra Wide-Band Localization and SLAM: A Comparative Study for Mobile Robot Navigation'. In: *Sensors* 11.2 (2011), pp. 2035–2055.

[59] Even Skjellaug. 'Feature-Based Lidar SLAM for Autonomous Surface Vehicles Operating in Urban Environments'. In: (June 2020). Master thesis Cybernetics and Robotics, NTNU. URL: https://folk.ntnu.no/edmundfo/msc2020-2021/Skjellaug_SLAM.pdf.

[60] Even Skjellaug, Edmund Brekke and Annette Stahl. 'Feature-Based Laser Odometry for Autonomous Surface Vehicles utilizing the Point Cloud Library'. In: July 2020, pp. 1–8.

[61] Unni Skoglund. *Førerløse ferger kan erstatte gangbruer*. URL: https://gemini.no/2018/06/forerlose-ferger-kan-erstatte-gangbruer/ (visited on 18/06/2018).

[62] Joan Solà, Jeremie Deray and Dinesh Atchuthan. *A micro Lie theory for state estimation in robotics*. 2018. URL: http://arxiv.org/abs/1812.01537.

[63] Michael Strohmeier, Thomas Walter, Julian Rothe and Sergio Montenegro. 'Ultra-Wideband Based Pose Estimation for Small Unmanned Aerial Vehicles'. In: *IEEE Access* 6 (2018), pp. 57526–57535.

[64] Namjoon Suh. 'Review on Parameter Estimation in HMRF'. In: (2017). arXiv: 1711.07561 [stat.ML].

[65] Richard Szeliski. *Computer vision algorithms and applications*. 2011.

[66] Kystlaget Trondhjem. *Fløtmann*. URL: https://www.kystlaget-trh.no/flotmann/ (visited on 10/05/2021).

[67] Kevin Walchko. 'Low cost inertial navigation: learning to integrate noise and find your way'. PhD thesis. Jan. 2002.

[68] John Wang and Edwin Olson. 'AprilTag 2: Efficient and robust fiducial detection'. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2016, pp. 4193–4198.

[69] Xin-She Yang. *2 - Mathematical foundations*. Ed. by Xin-She Yang. Academic Press, 2019, pp. 19–43. ISBN: 978-0-12-817216-2.

[70] Guoxing Yu, Yongtao Hu and Jingwen Dai. 'TopoTag: A Robust and Scalable Topological Fiducial Marker System'. In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2020).

Appendix A

Additional Material

A.1 Sensor Transformations

The relevant sensor transforms relative the *vessel center* body frame (here defined as $\mathcal{V}$) on milliAmpere are listed here. The transform to the GNSS antenna is defined as Equation (A.1). The transform to the IMU is defined as Equation (A.2). The transforms to each EO camera on the 360 EO-IR Sensor Rig is defined as Equations (A.3) to (A.7).

$$\mathbf{T}_{\mathcal{V}}^{\text{GPS}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.975 \\ 0 & 0 & 0 & -2.33 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.1}$$

$$\mathbf{T}_{\mathcal{V}}^{\text{IMU}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.20 \\ 0 & 0 & 0 & -1.65 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

$$\mathbf{T}_{\mathcal{V}}^{\text{EO\_F}} = \begin{bmatrix} 0 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2.68 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

$$\mathbf{T}_{\mathcal{V}}^{\text{EO\_FR}} = \begin{bmatrix} 0.309016994 & -0.951056516 & 0 & 0.0618034 \\ 0.951056516 & 0.309016994 & 0 & 0.1902113 \\ 0 & 0 & 0 & -2.68 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

$$\mathbf{T}_{\mathcal{V}}^{\text{EO\_RR}} = \begin{bmatrix} -0.809016994 & -0.587785252 & 0 & -0.1618034 \\ 0.587785252 & -0.809016994 & 0 & 0.11755705 \\ 0 & 0 & 0 & -2.68 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.5}$$

$$\mathbf{T}_{\mathcal{V}}^{\text{EO\_RL}} = \begin{bmatrix} -0.809016994 & 0.587785252 & 0 & -0.1618034 \\ -0.587785252 & -0.809016994 & 0 & -0.11755705 \\ 0 & 0 & 0 & -2.68 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.6}$$

$$\mathbf{T}_{\mathcal{V}}^{\text{EO\_FL}} = \begin{bmatrix} 0.309016994 & 0.951056516 & 0 & 0.0618034 \\ -0.951056516 & 0.309016994 & 0 & -0.1902113 \\ 0 & 0 & 0 & -2.68 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.7}$$

A.2 Global Fiducial Marker Transformations

In this section, the NED poses of the AprilTag fiducial markers used in the experiments conducted in this thesis are listed.

$$\mathbf{T}_{\text{NED}}^{\text{Tag0}} = \begin{bmatrix} -0.408789768 & 0.070084281 & 0.909933579 & -465.33 \\ -0.912191686 & -0.062225018 & -0.405011575 & -220.43 \\ 0.028235688 & -0.995598433 & 0.089367249 & -9.92 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.8}$$

$$\mathbf{T}_{\text{NED}}^{\text{Tag1}} = \begin{bmatrix} -0.408274781 & 0.020445789 & 0.912630085 & -459.44 \\ -0.910122951 & -0.086464549 & -0.405216109 & -208.11 \\ 0.070625186 & -0.996045105 & 0.053909477 & -10.08 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.9}$$