Augmented Reality Based Human Machine Interface for Semiautonomous Work Machines

Tuomo Palonen

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 10.10.2016

Thesis supervisor:

Prof. Arto Visala

Thesis advisor:

M.Sc. Heikki Hyyti

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Tuomo Palonen

Title: Augmented Reality Based Human Machine Interface for Semiautonomous Work Machines

Date: 10.10.2016 Language: English Number of pages: 8 + 67

Department of Electrical Engineering and Automation

Professorship: Automation technology Code: AS-84

Supervisor: Prof. Arto Visala

Advisor: M.Sc. Heikki Hyyti

Forest machines are being automated today. However, the challenging environment and complexity of the work makes the task difficult. A forest machine operator needs easily interpretable input from the machine in order to supervise and control it. Hence, a device that would show the digital information as a part of the real environment is desired.

The goal of the thesis is to implement a real time augmented reality display for forest machines. The main task is to estimate the pose of the user's head because the virtual data should be aligned with real objects. Also, the digital content and how it is visualized has to be considered.

A machine vision camera and inertial measurements are used in the pose estimation. Visual markers are utilized to get pose estimate of the camera. And, orientation from inertial measurements is estimated using an extended Kalman filter. To get the final estimate, the orientations of the two devices are sensor fused. Furthermore, the virtual data comes mainly from an on-board lidar. A 3D point cloud and a wire frame model of a forestry crane are augmented to a live video on a PC.

The implemented system proved to work outdoors with actual hardware in real time. Although there are some identifiable errors in the pose estimate, the initial results are encouraging. Further improvements should be targeted to the accuracy of marker detection and to the development of a comprehensive sensor fusion algorithm.

Keywords: augmented reality, pose estimation, forestry, calibration, sensor fusion

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Tuomo Palonen

Työn nimi: Lisätyn todellisuuden käyttöliittymä puoliautonomisiin työkoneisiin

Päivämäärä: 10.10.2016 Kieli: Englanti Sivumäärä: 8 + 67

Sähkötekniikan ja automaation laitos

Professuuri: Automaatiotekniikka Koodi: AS-84

Työn valvoja: Prof. Arto Visala

Työn ohjaaja: DI Heikki Hyyti

Haastava ympäristö ja monimutkaiset työtehtävät tekevät metsäkoneiden toimintojen automatisoimisesta vaikeaa. Olisikin toivottavaa, että metsäkoneen-kuljettaja pystyisi tulkitsemaan koneelta tulevaa tietoa helposti ja nopeasti. Ratkaisuksi ehdotetaan järjestelmää, joka sulauttaa digitaalisen tiedon osaksi käyttöympäristöä. Tämä mahdollistaisi puoliautonomisen työkoneen sujuvamman valvomisen ja ohjaamisen.

Tämän työn tavoitteena on toteuttaa lisätyn todellisuuden näyttö metsäkoneisiin. Tärkeimpänä tehtävänä on estimoida käyttäjän pään sijainti ja asento, sillä digitaalisen datan pitäisi limittyä todellisuuden kanssa. Lisäksi on pohdittava virtuaalisen tiedon sisältö, ja kuinka se esitetään käyttäjälle.

Asennon ja paikan mittaamiseen käytetään päähän kiinnitettyä konenäkökameraa ja inertiamittausyksikköä. Kameralla tunnistetaan työkoneen hyttiin sijoitettuja tunnistemerkkejä, joilla sekä kameran paikkaa että asentoa voidaan estimoida. Asentoestimaattia korjataan vielä inertiamittauksilla anturifuusiota hyödyntäen. Virtuaalinen tieto näytölle tulee pääasiassa laserkeilaimelta ja se lisätään tietokoneen ruudulla näkyvään videoon kolmiulotteisena pistepilvenä. Myös metsäkoneen puomi ja työkalu esitetään virtuaalisena mallina.

Toteutettu järjestelmä osoittautui toimimaan oikealla laitteistolla ulkoilmassa tehdyssä kokeessa. Alustavat tulokset ovat rohkaisevia, mutta myös paikan ja asennon virheitä havaittiin ja identifioitiin. Tulevaisuuden kehityskohteita ovat tunnisteiden paikan tarkempi mittaaminen ja kokonaisvaltaisemman anturifuusion kehittäminen.

Avainsanat: lisätty todellisuus, paikan ja asennon estimointi, metsätalous, kalibrointi, anturifuusio

Preface

It has been interesting to work with such a timely topic. Many people have taken their first contact to augmented reality by playing mobile games during this project. The future possibilities, especially in the industry, are undeniable. I would like to thank my supervisor professor Arto Visala for giving me the opportunity to work with this subject as a member of the Autonomous Systems research group. Thank you for creating such a great atmosphere at the work place and always stating the importance of this thesis.

I want to thank my advisor Heikki Hyyti on his great impact for my work. I could count on that whatever problems I faced, you had a solution. It is safe to say that without your support, the end result would have lacked some quality.

Also, I would like to thank Ville Toiviainen especially for helping me to produce the PCB. Ville had always time to answer my questions despite of his own thesis work.

I want to express my gratitude to my family for giving me such a happy home. You have always been there for me when I have needed support, and you have given me the best possible tools to continue my life. I would also like to thank all my friends, especially those who have shared my journey from the early childhood to this date. Lastly, my greatest gratitude goes to my love Elina. You have looked after me and brought joy to my life throughout this whole time at university. Thank you so much.

Espoo, 10.10.2016

Tuomo Palonen

Contents

Symbols and abbreviations

Symbols

In general, bold capital letters represent matrices (e.g. $\boldsymbol{R}$) and plain capital letters are the elements of matrices (e.g. R_{12}). Small bolded letters are used as vectors and plain small letters represent scalars (e.g. $\boldsymbol{x} = (x, y, z)$).

$\boldsymbol{R}$	rotation matrix
$\boldsymbol{K}$	intrinsic matrix
$\boldsymbol{T}$	transformation matrix
$\boldsymbol{I}$	identity matrix
$\boldsymbol{M}$	projection matrix
$\boldsymbol{A}$	windowed second order matrix
$\boldsymbol{D}$	distortion model
k_i	distortion parameter
$\boldsymbol{t} = (t_x, t_y, t_z)$	translation
$\boldsymbol{x}_W = (x_W, y_W, z_W)$	point in world frame
$\boldsymbol{x}_C = (x_C, y_C, z_C)$	point in camera frame
$\boldsymbol{x}_I = (x_I, y_I)$	point in image frame
(ψ, θ, φ)	Euler angles (yaw, pitch, roll)
ϕ	scalar angle in axis-angle representation
f	focal length
(o_x, o_y)	principal point of camera
$\boldsymbol{c}$	camera center in world frame
$\nabla \boldsymbol{I}(\boldsymbol{p_i})$	image gradient
$\overline{\boldsymbol{q}}$	unit quaternion
t	time
T	temperature
$\boldsymbol{C_3}$	last row of direction cosine matrix
$\boldsymbol{Q}_k$	process noise covariance
$\boldsymbol{R}_k$	measurement noise covariance
$\boldsymbol{x}_k$	state vector
$\boldsymbol{u}_k$	control vector
b_i^{ω}	gyroscope bias in i:th axis
${}^{b}\boldsymbol{a}_k$	non-gravitational acceleration
$\sigma_{C_3}^2$	variance of DCM-state prediction
$(\sigma_b^{\omega})^2$	variance of bias state prediction
σ_f^2	variance of acceleration measurement noise
σ_a^2	scale factor for non-gravitational acceleration
f_{meas}	accelerometer measurement
ω_{meas}	gyroscope measurement
$p_{gain/bias}^{f_i/\omega_i}(T)$	temperature dependent calibration model

Abbreviations

2D	two-dimensional
3D	three-dimensional
AR	augmented reality
BRIEF	binary robust independent elementary features
CAD	computer-aided design
CCD	charge-coupled device
CMOS	complementary metal oxide semiconductor
CPU	central processing unit
DCM	direction cosine matrix
DLT	direct linear transformation
DoG	difference-of-Gaussian
EKF	extended Kalman filter
EPnP	efficient perspective-n-point
FAST	features from accelerated segment test
GPS	global positioning system
ID	identification
IMU	inertial measurement unit
k-d tree	k-dimensional tree
lidar	light detection and ranging
LED	light-emitting diode
MEMS	microelectromechanical systems
OI	orthogonal iteration algorithm
PC	personal computer
PCB	printed circuit board
PnP	perspective-n-point
PTAM	parallel tracking and mapping
RANSAC	random sample consensus
RGB	red, green and blue color model
RMSE	root mean square error
SIFT	scale invariant feature transform
SLAM	simultaneous localization and mapping
SURF	speeded up robust features
UI	user interface
USB	universal serial bus
VR	virtual reality

1 Introduction

Forestry is a great field of industry, especially in Finland. Of all the exports in Finland in 2014, forest products took 20.1 % and were worth of 11.2 billion Euros [1]. The total worth of world's exports was €338 billion in 2014 [2]. In the same year, Finland was the third biggest exporter of forest products in Europe after Germany and Sweden and sixth biggest in the world. Forestry is a multidisciplinary work. It is not just about logging and processing trees, but also actively controlling the growth and health of forests.

The logging of trees is mainly done with machines today. Modern forest harvesters are able to fell, delimb (remove branches) and cut trees to logs in site semiautonomously. The logs are then collected and delivered to roadside by transporting vehicles called forwarders. Automating different processes has proved to increase productivity and quality. Still, forestry machines are mostly manually controlled but there is an ongoing research to make them more autonomous [3].

One work in the literature was concentrated on the autonomous path tracking of a forwarder [4]. There have been also researches for silvicultural tasks like automatic detection of young spruces [5], and autonomous control of a boom which has a freely hanging tool attached to it [6]. With these two methods it could be possible to clean the weeds around the spruces without manual control. Automating some phases of the work can be valuable, but it is a long way before fully autonomous systems are in sight.

As stated in [7], particularly harvesters are difficult to fully automate because there would be very complex tasks to a computer like which tree to be cut next and in what direction. The dynamic environment outside in rough surface makes it also difficult for sensors and algorithms to produce accurate and sufficient information. This may lead to unexpected events that the computer should handle. When going towards more autonomous forest machines, the operator could benefit from knowing what the machine has learnt and is planning to do.

The problem is that conventional sensory output information on flat screens is challenging and slow to interpret. The target should be that the forest machine operator can understand the machine with just one look. Thus, there could be a need for a system that shows the digital information as a part of the real world. And, this is exactly what augmented reality (AR) does.

In AR the digital data is projected to the user's visual field of view so that the virtual content feels as a part of the real world. There is evidence that AR can improve the performance of the users in traditional industrial tasks like assembly work [8] and maintenance [9]. So, there are reasons to believe that it could be beneficial also in the field of forestry.

The goal of this thesis is to design and implement a prototype of an augmented reality display for forest machines. In this work the prototype should be able to work in a real life test with real hardware, and give a realistic visualization. The virtual data is targeted to be shown in real time and it will come mainly from an

onboard laser scanner. The prototype should also be a useful platform for further development of augmented reality related algorithms.

However, the implemented system is not required to be close to a final product. Hence, the system is not developed for any specific head-mounted display in mind. Also, the interaction between the user and the machine is outside the scope of this work since it is such a large problem in itself.

The main issue of this thesis is to track the location and orientation of the head/display, also called as pose estimation. Pose estimation is needed because the virtual points existing in 3D world have to be accurately projected on to the displaying device. The main research problem of this thesis is *how to measure the pose of an augmented reality display in a forest machine*. More specifically, the research questions are:

- What are suitable pose estimation methods in a forest machine?
- Which are the main error sources impacting on the pose estimation quality?
- How to visualize the virtual data for a forest machine operator?

The structure of the paper is following. Chapter 2 introduces the reader to the field of augmented reality. It discusses the topics that are currently the most researched in AR and gives a solid background for further studies. Chapter 3 surveys the literature on the vision-based pose estimation. It also mentions some methods found from the literature with more details. Chapter 4 studies inertial measurement units (IMU) and how they are used in attitude estimation. Chapter 5 describes the design of the implemented prototype and explains the reasons behind the choices. The evaluation of the finished prototype including the test setup is discussed in Chapter 6. Finally, Chapter 7 concludes the thesis by answering to the research questions. It also discusses about the quality of the implemented prototype and gives proposals for future development.

2 Augmented reality

Augmented reality (AR) integrates digital data into real world in real time. The digital information can be merged to live video or it can be augmented to user's perception of the environment. The field of augmented reality is relatively young but it contains clearly identifiable branches. The main topics currently are tracking, interaction techniques, calibration and registration, applications and displaying techniques [10]. This section discusses the properties and challenges regarding these main subjects after presenting briefly the history and definition of augmented reality.

2.1 Background

Although the augmented reality technology has started to get more attention only recently, the roots of it go back to 1960s. Professor Sutherland at Harvard university was first to introduce a head-mounted display which was able to add virtual data on the visual field of view (Figure 1) [11]. It took a while before the term "augmented reality" was introduced. It was done by professor Caudell, a researcher at Boeing Computer Services, who developed a head-mounted display for factory workers in the aerospace industry in 1992 [12]. Augmented reality was seen as a technology that augments the line of sight of the user with virtual information.

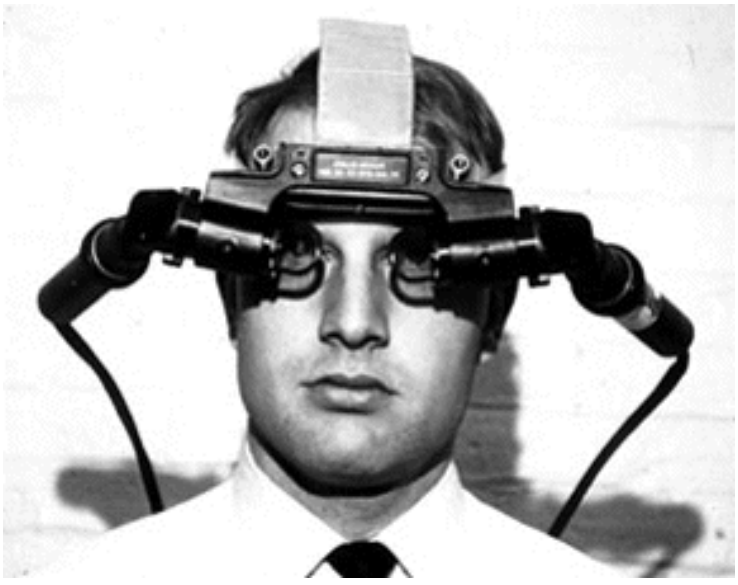

Figure 1: Optics of the first head-mounted display in 1968 [11].

The rapid growth of AR research in the 1990s, initiated Ronald Azuma to publish a comprehensive survey [13]. By his well-known definition, augmented reality system has the following characteristics:

- Combining real and virtual
- Being interactive in real time
- Registering (aligning) real and virtual objects in 3D

This definition allows other sensory information than just visual to be accounted as augmented reality. A system could also augment sound, haptic feedback and smell to the user's reality. Furthermore, the definition does not limit the technology to be a head-mounted display.

A related technology to AR is virtual reality (VR), which is a completely computer-simulated reality and thus differs from AR. Augmented reality could be seen in the middle of real environment and virtual environment [14]. The two technologies have mostly similar challenges but differ in some requirements.

Rendering is more demanding in VR than in augmented reality as the virtual images have to be more realistic since they completely replace the real world [13]. However, tracking the pose of an object must be more accurate in AR than in virtual environments because the errors are easier to detect [15]. Next, the importance of accurate pose estimation is discussed regarding augmented reality.

2.2 Pose estimation

The pose of an object consists of its position and orientation in three-dimensional space. The position is described as a translation vector $\boldsymbol{t}$ from the center of the reference coordinate system origin. On the other hand, orientation can be described as three consecutive rotations about the axes of the reference frame, from which a rotation matrix $\boldsymbol{R}$ can be formed [16]. They can be defined as

$$\boldsymbol{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \tag{1}$$

$$\boldsymbol{R} = f(\psi, \theta, \varphi), \tag{2}$$

where ψ is an angle of rotation about z-axis, θ is a rotation about y-axis and φ is a rotation about x-axis. These angles are also called as yaw, pitch and roll respectively. In Figure 2, the translation and rotation of an object is visualized.

In order to preserve the illusion of virtual and real world coexisting, the augmented virtual objects have to be aligned and synchronized with the real world seamlessly. This is also called as the registration problem. The needed accuracy is of course depended on the application.

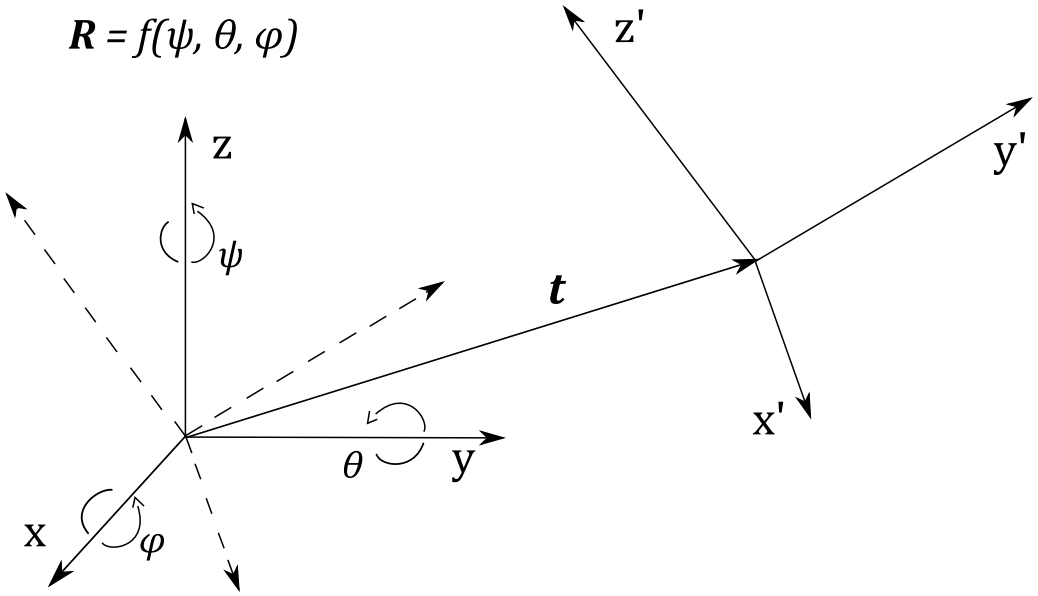

Figure 2: Pose of an object described as a rotation ***R*** and translation ***t***.

But, if you think about a tree standing 10 meters away from a spectator in a forest, and a virtual model of that tree is to be aligned. Even two degrees of estimation error of the head orientation would mean a misplacement of approximately 35 cm. Then, it would not be obvious for the user to interpret the model belonging to the right tree. This is demonstrated in Figure 3. As it turns out, one of the main error sources for registration is in fact the tracking error of the head pose [17].

According to Welch and Foxlin [18], there are identifiable characteristics that an ideal tracking system would contain. It should be cheap, fast, robust, accurate,

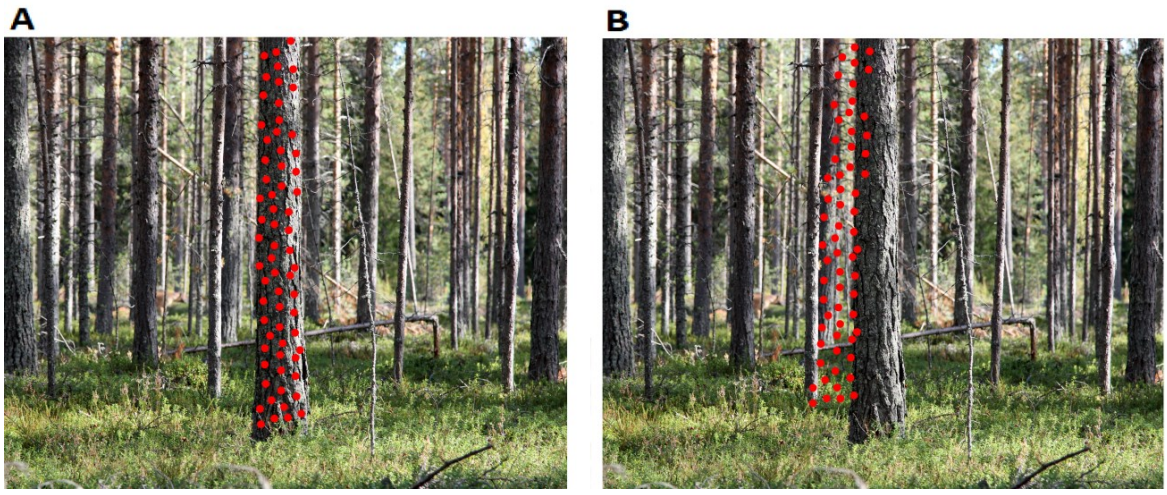

Figure 3: Illustration on the effects of inaccurate pose estimation. A) Accurate pose estimate, B) Inaccurate pose estimate.

tracking 6 degrees of freedom, immune to occlusions, tenacious, wireless, small and self-contained. Unfortunately, a tracking system that would satisfy all the needs is not available [18]. Also, there are different types of tracking errors that can decrease the performance or cause sickness to the user.

In [18], the errors were classified into two categories: static and dynamic errors. Static tracking errors involve spatial distortion, spatial jitter and stability. Spatial distortion contains repeatable errors at different poses caused by sensor scale factors. Noise can make the perceived image unintentionally shaking and this is called the spatial jitter. Stability means slow but steady changes in tracker output which may become visible over time. It may be caused by e.g. temperature changes.

Moreover, dynamic errors can be divided into latency, latency jitter and other dynamic errors. Latency is the mean delay between the motion and the corresponding data delivery. Latency jitter on the other hand is the change of latency which can cause stepping and twitching to the moving image. Other dynamic errors such as prediction algorithm overshoot fall to the last category.

Several AR tracking solutions rely on vision-based tracking [10] because it provides accurate pose estimates and camera is often already part of the system. However, vision-based tracking systems do have some limitations. When velocity increases, tracking becomes imprecise due to motion blur. Occlusion can also lead to a tracking failure as the trackable features cannot be seen. One must do trade-offs with speed and quality of the measurements, as higher frame rates and complex algorithms increase the computational burden.

Alternative solution is to use inertial sensors, which are fast and immune to occlusions. They can also be very small and capture high velocities. The problem with the inertial sensors is the drift. The errors of accelerometers and gyroscopes will build up over time as the measurements have to be integrated in order to obtain position and orientation. The drift could be compensated by the use of external sensors such as magnetometers and GPS. But, in some environments these sensors become useless. Magnetometers get interfered by metal and GPS don't work well in indoors or even in dense forest.

Other devices such as mechanical and acoustic sensors have been also utilized as a tracking solution in AR [19]. While the mechanical sensors are usually quite accurate, they work only for a limited range of motion. Acoustic sensors on the other hand can be small and lightweight, but suffer from a lack of accuracy, especially when the distance increases.

In order to compensate the shortages of any individual tracking system, hybrid systems that combine the measurements from different sources are getting more attention. Fusing inertial and vision has been a one of the greatest field of interests [20], [21], [22].

2.3 Registration

The data points of a virtual object existing in real world have to be projected onto the displaying device. To be able to do that, coordinate transformations must take place. A coordinate transformation means mapping of a vector from one coordinate system to another [23]. There may be two or more different coordinate systems depending on the application needs. Usually, there is a world coordinate system that can be considered to be the reference with respect to other coordinates.

Considering a simple augmented reality system with a single camera, the coordinate systems can be assigned as in Figure 4 on the next page. The goal is to map a point in the world coordinate system W into the image coordinates I at the right place. The camera frame C point $\boldsymbol{x}_C$ is obtained by

$$\boldsymbol{x}_C = {}^C\boldsymbol{T}_W \boldsymbol{x}_W, \tag{3}$$

where $\boldsymbol{x}_W$ is a homogeneous point in world coordinates and ${}^C\boldsymbol{T}_W$ is a transformation matrix from world frame to camera frame [23]. The transformation matrix consists of rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t}$, which were presented in (1) and (2). It can be expressed as

$$\boldsymbol{T} = [\boldsymbol{R} \mid \boldsymbol{t}] = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

The matrix $\boldsymbol{K}$ that contains the intrinsic parameters of the camera, converts the points in camera coordinates to image coordinates. Under perspective projection, the transformation is obtained from

$$\boldsymbol{x_I} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \boldsymbol{K}[\boldsymbol{I} \mid \boldsymbol{0}]\boldsymbol{x_C} = \begin{bmatrix} -f/s_x & 0 & o_x \\ 0 & -f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} [\boldsymbol{I} \mid \boldsymbol{0}] \begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix}, \tag{5}$$

$$x_I = \frac{u}{w},$$

$$y_I = \frac{v}{w},$$

where f is a focal length, s_x, s_y are pixel sizes and (o_x, o_y) is the principal point of the camera [24]. The variables x_i and y_i are image coordinates i.e. pixel coordinates and $\boldsymbol{I}$ is an identity matrix. In this simple model lens distortion and skew were assumed to be zero. The intrinsic parameters can be calibrated offline if

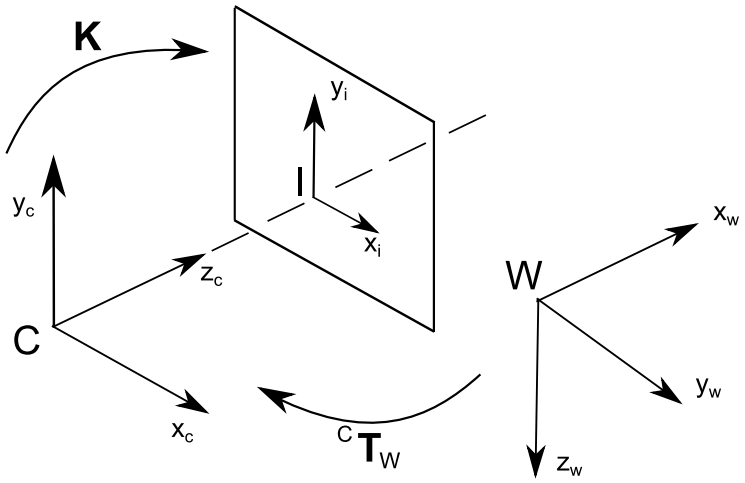

Figure 4: Coordinate systems and transformations.

the system does not require zooming capability. And, this is usually the case in AR. The camera calibration will be discussed later in Section 3.1.

In case of multiple consecutive coordinate transformations, the transformation matrices can be linked simply by multiplying them [23]. Another useful property is that the direction of transforming from one coordinate system to another can be changed by taking the inverse of the 4x4 transformation matrix: ${}^{C}T_{W}^{-1} = {}^{W}T_{C}$ [23].

The link between the registration and successful pose estimation is quite clear. The transformation of the points between the real world and the display needs the information about the pose of the head in the environment. But, it is also necessary to obtain the position of the real world objects that are to be augmented. For example, lidar can be used to get point cloud information of the real world [25] which can be then augmented to the display through coordinate transformations. It is also possible to match a known model of an object in a camera image to find out the pose of the real world object. The selection of the method is solely application specific.

A very important part of augmented reality is the technology that allows the user to actually see the virtual reality along with the real world. Different displaying techniques are discussed in the following part.

2.4 Displaying techniques

Displaying devices are needed in every AR application. They can be categorized based on the visualization technique or the displaying position [26]. There exists a video see-through, optical see-through and a projective visual presentation possibility. Also, the devices can be either head-mounted, hand-held or spatially positioned. Each type of a display has its advantages over another, and these will be described next.

Head-mounted displays feel like the most intuitive solution for many AR applications as it generally enables the user to move freely in the environment and

it is hands-free. It has been stated recently that there are still development to be made to get the head-worn displays to be light weighted and untethered [26], [27]. It is essential that the user can wear the device comfortably. Battery life and heating are some of the issues that the developers have to consider.

Since the hand-held devices like smart phones and tablets are already widely used, it is realistic that they might become mass-market displays for AR applications [26]. Other features that support this is their mobility and minimal intrusiveness [10]. Nevertheless, these displays require the user to hold the device, which will limit the applications they are suitable.

The technique, which will not require a user to wear or hold any devices, is the spatially positioned display. This category of displays augments images straight into the user's environment [26]. Virtual objects can be integrated to displaying screens like television or projected onto some real world objects. Applications that do not demand mobility, like presentations, are ideal for statically positioned displays. Next, the visual display techniques are compared.

Video see-through devices augment the virtual objects into the video feed the user monitors. One of the advantages of this solution is the capability to match the viewing delays of real and virtual [13]. This is not the case for optical see-through devices, as they allow the user to see the real world with overlaid graphics. Optical see-through is achieved by holographic optical elements, half-silvered mirrors or other related technologies [10].

According to [13], one advantage over optical see-through is the fact that contrast issues are easier to deal with video see-through displays. The brightness of the virtual objects should be matched with the real world. And, the eyes can cover larger range of contrast of the real world than the video camera is able to do. This makes the task for optical see-through devices difficult. It is also easier to remove real world objects virtually with video see-through technology as the real world is digitalized [26]. Furthermore, video see-through is able to include depth cues by occlusion in a simpler manner [13].

As stated in [13], there are also features that advocate the use of optical see-trough displays. The resolution of the real world is degraded with video see-through displays. Optical see-through allows also a safer operation in a power cut-off situation, because the user can still see the real world. Moreover, video see-through devices can suffer from eye-offset. The camera is usually located slightly aside of the eyes, and if this is not compensated, the viewer has to do some effort to accommodate. Furthermore, optical see-through displays do not have a delayed view of the real world.

One optical technique that is under development can handle a few problems of the conventional optical see-through displays [26]. It is called a virtual retinal display. It can draw images directly onto retina with a low-power laser. This allows the user to get a wider field of view and solve contrast issues.

Another kind of visual technique is to project the virtual elements onto real world objects. According to [26], it has both positive and negative aspects. Advantage of it is that the user is not required to wear any device. For that reason, it allows the viewer's eyes to be accommodated during focusing. It can also offer a large field of view for the user. However, projective displays are only suitable for

indoor use as the projected images are not bright and have a low contrast. They generally also have a lack of occlusion capability, but this can be improved by head-mounted projectors [28]. The target surfaces can be coated with retroreflective material which will reflect the projected light towards the light source close to the user's eyes. Then, the virtual images can only be seen if the line of sight is free.

2.5 Interaction and user interface

Many of the AR applications need not only to register virtual objects with real objects but also to provide interaction capability [26]. Moving, selecting and rotating virtual objects are common user actions in AR applications. Additionally, user interface could allow drawing trajectories or paths, and giving other inputs such as numerical values or text. These are all included in basic desktop UIs as well. However, in AR this all happens in the 3D world. While the mouse and keyboard were created specifically for flat computer displays, the developers of AR are finding intuitive ways to control the virtual content superimposed in the environment.

There are a lot of different AR specific interaction techniques which could be divided under the following classes:

- spatial and tangible interaction
- voice and aural interaction
- human gestures and gaze

Spatial and tangible interaction involves manipulation of real objects in a three-dimensional space [29]. A pointer is an example of this category, because it allows the user to point onto the virtual object of interest and select it. Tangible interaction techniques are well-accepted as the physical objects give a hint of how they should be handled. Additionally, the familiar properties and physical constraints of tangible objects provide a natural way of interaction [10].

For instance, Kawashima et al. [30] used a real paddle to control virtual furniture on a book. The objects could be picked on to the paddle and viewed from different angles by rotating the paddle. A piece of furniture could be removed by hitting it with the paddle.

Voice commands present an alternative way of interfacing the machine. This technique is suitable for linking a command to a specific function. For example in modern mobile phones one can found already applications, such as Apple's Siri [31]. Unfortunately, speech recognition is not the most fastest and accurate method of interacting, but it is all the time improving. Noisy environments can however still be problematic. The use of audible interfaces with other interaction methods can provide a very intuitive control in AR applications.

Human motions can also be tracked, thus allowing the user's body to be an interaction tool. Hands are the most used body part for interaction since working

with them is very natural for people. So, there have been many methods developed for hand tracking including finger tracking [32] and the use of specific gloves [33].

Another potential interaction technique is the human eye. By following the motion of the pupils with tiny cameras, the gaze of the user can be calculated. The gaze could provide similar pointing capability like the computer mouse. Esteves et al. [34] used eye-tracking to interact with a smart watch. The user initiated tasks by following a specific moving target on the display for small amount of time.

All in all, it always depends on the application which interaction technique or combination of them should be used. However, there are certain designing factors in AR interfaces independent of the chosen technique [35]. The learning of the interaction technique should not be too demanding. It is encouraged to utilize skills that most people have gained in their everyday life. The use of the interface should also be ergonomic to avoid fatigue or discomfort. And, users prefer systems that can provide feedback [36], [37]. For example, if the user has selected a virtual object, an audio sound indicating a correct selection could be provided.

It has been noticed that users of AR tend to focus their concentration on the application and its display. This might lead to dangerous situations if a user does not observe the change of state in the environment or notice an obstacle nearby. For example, in the outdoor use a car might come unexpectedly to the user's path.

According to [37], the effect becomes especially bad with video see-through displays because the surroundings outside the field of view are not visible. Motion-based user interfaces can also be a health risk, since it is possible that the user hits a solid object nearby while concentrating on the application. Thus, it is important to design the system safety in mind. Users should be instructed about the possible risks and how to avoid them.

This chapter described elementary challenges and aspects of augmented reality. Following part of the text will go more deeply in the topic of vision-based tracking which is the most active research area by far regarding pose estimation in AR. In this paper the main focus is on marker-based estimation.

3 Vision-based pose estimation

The idea of vision-based pose estimation is to extract a pose of an object using computer vision algorithms and captured camera images. There are many different configurations and algorithms to do this. In this chapter feature-based methods with a single camera are considered. The chapter starts with discussion about camera calibration, an essential part of several pose estimation methods. Then, it is presented how to utilize markers in pose estimation. Finally, a short introduction to an alternative option to markers is given, called natural-feature based methods.

3.1 Calibration of camera

Camera calibration is needed for metric measurements in a 3D world from 2D images. The purpose of calibration is to obtain the intrinsic and/or the extrinsic parameters. Intrinsic parameters describe the relationship between camera coordinates and image coordinates. Extrinsic parameters on the other hand describe the relationship between world coordinates and camera coordinates. The idea of calibration is to use correspondences between 3D locations of known points and their projections in the image plane. Usually the 3D points are structured so that they are easy to detect from the image, e.g. corners of a checkerboard [38].

Camera calibration is not a new problem and different kinds of methods have been proposed [39]. Mainly, these can be divided into two categories [40]. One of them is an iterative method that minimizes a geometric error, an error between known image points and projected image points. Another method is a closed-form solution where the parameter values are directly obtained through a non-iterative algorithm. A typical procedure is first to apply a closed-form algorithm and then use the parameters acquired from it as an initial guess for the iterative search.

Direct Linear Transformation (DLT) [41] is a well-known closed-form solution. It solves the parameters of the projection matrix $\boldsymbol{M}$ which derivation can be found in Appendix A. The projection matrix can be constructed from three parts:

$$\boldsymbol{M} = \boldsymbol{K}[\boldsymbol{R} \mid \boldsymbol{t}] = [\boldsymbol{KR} \mid -\boldsymbol{KRc}], \tag{6}$$

Intrinsic matrix $\boldsymbol{K}$ and rotation matrix $\boldsymbol{R}$ can be extracted from the matrix $\boldsymbol{KR}$ by RQ-decomposition which is analogous to QR-decomposition [42]. Vector $\boldsymbol{c}$ describes the center of the camera in the world. It is the point where $\boldsymbol{Mc} = \boldsymbol{0}$.

The standard DLT does not take into account lens distortions and it is also affected by noise [40]. In the general case, the points cannot lie in the same plane, because it is a degenerate solution providing a non-unique solution.

To refine the parameters from the close-formed solution, there exist several iterative minimization methods. Besides the more conventional gradient descent and Gauss-Newton optimization methods, one could also mention an often used Levenberg-Marquardt algorithm.

Furthermore, distortion correction has to be used, because the projection model (typically a perspective projection) is always only an approximation [40]. Radial distortion appears when the light starts to refract more and more further from the center. For example, straight lines in world can appear “barrel”-like in the image. Tangential distortion is another typical model. It exists when the camera lens is not exactly parallel with the camera sensor.

Perspective projection (pinhole model), where all the light rays travel straight meeting at a projection point, is not the only camera model available. For wide-angle/fish-eye lenses, the pinhole model is not any more suitable. For instance, one should consider using an equidistance projection model in those cases [43].

3.2 Marker-based pose estimation

A vision-based tracking determines the pose from the images acquired by a camera. Independent from the method, the system needs to recognize features from the images in order to do pose calculations. If the environment can be modified, it can be wise to add objects with known dimensions that can be easily detect from the scene. These are usually called markers, or fiducials.

There are advantages of using markers compared to other tracking methods such as model-based and natural feature-based systems [37]. Challenging environments can be problematic especially for natural feature-based methods. Surroundings with repetitive textures, such as white walls, moving objects (e.g. trees in the wind) or reflective surfaces can hamper the detection of features.

As stated in [37], there is other support for using markers as well. They provide a correct scale, because the measures of the markers are known. Additionally, they can be used as a meaningful coordinate system reference, as they are usually planted on walls, floors or similar rigid structures.

The most utilized fiducials in AR are coarsely square-shaped black and white markers, and infrared markers. They have been widely used because machine vision algorithms can extract them easily from the surrounding image scenery. There are also options for colorful markers and circular markers but they are not a concern in this thesis.

3.2.1 Black and white markers

Black and white markers have usually thick black borders on a white surface [44]. The black borders are there for the detection of the marker and its four corners. Four coplanar but non-collinear points are enough to calculate the pose between the marker and camera uniquely [45]. This is the main reason why square-shaped markers are popular.

The detection starts by binarizing an image using e.g. adaptive thresholding or canny detector in order to recognize edges [44]. Then regions of black pixels are detected and all but rectangle ones are rejected. The rectangle contours are segmented into four lines and the corners are found from the intersections. The

corner locations can be refined using e.g. Harris-Stephens method [46]. It is important to get the corners detected accurately because it has a significant effect for the pose calculation.

When the marker corners are detected, they should be identified. Even if we have only one marker, the algorithm has to recover the order of the four corner points based on the inner image pattern of the marker. Square-shaped markers have two distinctive identification techniques: template matching and 2D barcode reading [37]. Template markers (Figure 5B) handle this problem by having a simple image inside the borders and matching a template to it. A marker template is a sampled image of the marker having a specific grid size.

The detected marker is transformed to its canonical form by applying a perspective transformation [37]. It is then divided into cells forming the same grid pattern as the template. The marker and all templates are compared in four possible orientations using a suitable similarity method. The method can be for example a sum of squared differences or a normalized cross-correlation.

But, template markers have some problems. Large number of markers will make the identification step slow. It is also difficult to create lot of markers that are dissimilar enough in four orientations. 2D barcode markers (Figure 5A) are replacing template markers today, because they produce less false positives [47]. Each cell of a barcode marker contains only black or white pixels. Thus, they can be interpreted as a binary code.

The binary code represents an identification number but it can also have additional information if needed. The additional bits can be used e.g. to identify the orientation of the marker. They can also provide error detection and correction possibility using for example a Hamming code [48]. However, the physical size of the cells decreases, when the amount of bits grows. And, the maximum distance the markers are still detected also decreases. Decisions regarding to this trade-off have to be thus made.

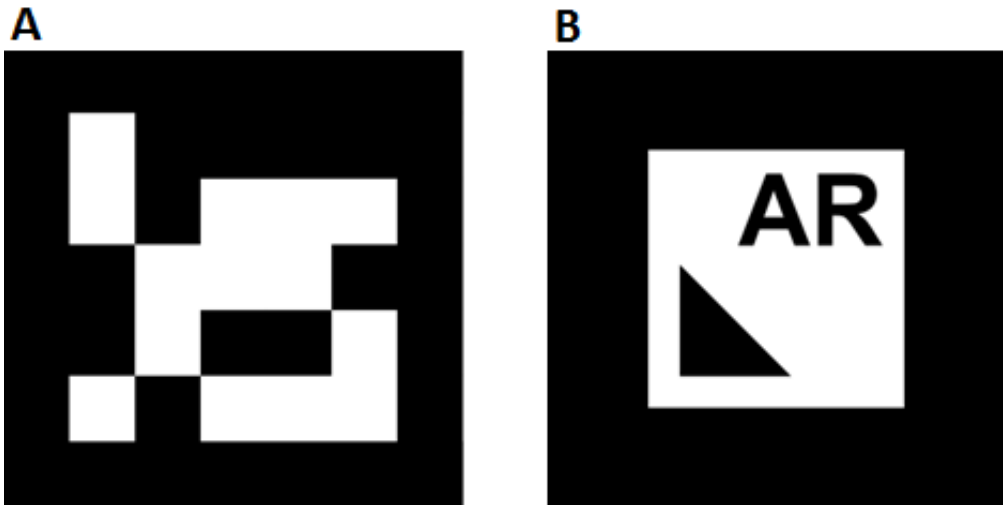

Figure 5: There are both 2D barcode and template markers.
A) ArUco 2D barcode marker [50]. B) A template marker [49].

Developers of AR applications today can utilize freely available software libraries which offer some pre-built functionality. The first and probably the most well-known library is ARToolKit [49] which uses black and white square marker for detection and pose estimation. Other libraries utilizing such fiducials are e.g. ArUco [50] and VTT's ALVAR [51]. It is up to the developer to select the best suitable tool, when weighing options like the used platform, environment and ease of use.

Lot of algorithms has been proposed to calculate the pose from 2D markers. The same principles apply that were already told in the camera calibration section 3.1. This time however, the intrinsic parameters are already known.

Lu et al. introduced Orthogonal Iteration Algorithm (OI) [52] that is more efficient and no less accurate than Levenberg-Marquardt method [53]. Lepetit et al. [54] proposed a method called EPnP which is a non-iterative algorithm faster than OI but slightly less accurate. However, the initial solution from EPnP can be further refined by some iterative optimization method in order to obtain the same accuracy as OI.

Ababsa et al. [44] presented a real-time 2D fiducials tracking system that utilizes the OI algorithm in the pose estimation. Their tests were made indoors and the locations of the markers were known. They superimposed the markers with wireframe models of a cube or a pyramid and estimated the reprojection error to be between 0.7 and 1.2 pixels. The pose accuracy is reduced, when only one fiducial is detected from the camera image.

3.2.2 Infrared markers

As previously mentioned, also infrared-based markers are much utilized in pose-estimation with a camera. An infrared pass-band filter can be used on a camera lens, but most of the cameras detect the LEDs even without it. Another possibility is to have the LEDs around the camera lens and use retroreflective material, which will reflect the infrared light back to the light source.

Once the LEDs show up bright from the image background, the center of the individual LED blobs have to be calculated. Thresholding can be used to find a blob of pixels that represent an individual LED. After that the center is calculated using e.g. intensity-based weighted average of pixel coordinates [55], [56].

In order to solve the perspective correspondence problem, the image points of the target should be somehow identified. If the amount of markers is not too large, it is possible to test all the possible combinations of image and target points, and pick the one with the minimum reprojection error. In practice, it would work only in very restricted cases without modifications. Another possibility is to blink the LEDs in specific temporal order to obtain their identity. This means that the camera has to take image sequences of the LEDs before they can be identified.

For LED-based head pose estimation, Meers et al. [55] suggested a method where three infrared LEDs are mounted on a pair of glasses. The LEDs are placed in a specific geometry (isosceles triangle) with known measures. The constraints given by the geometry are utilized in order to project the image points into 3D

world from the camera coordinates. Then, the gaze of the 3D LED model can be projected on a 'virtual screen' relative to the camera. Accuracy of the system was experimented to be within 0.5 degrees when the user is no further than one meter from the camera. This kind of setup means that the head has to face the camera and all the LEDs have to be in sight.

Another pose estimation method using infrared LED markers was implemented by Faessler et al. [56]. Their algorithm requires at least four markers to be placed on a target object (quadrotor) in an arbitrary but non-symmetric manner. The exact position of the markers is measured using an external motion capture system. Other prefaces included camera calibration and equipping the camera with an infrared-pass filter. Once the image positions of the LEDs are detected, the pose estimation algorithm starts.

A brute-force and a predictive method are used to calculate the initial pose which is further refined by an iterative pose optimization. From three LED correspondences, one can calculate four pose candidates (P3P algorithm). The rest of the LEDs are used to define the right pose by finding the one with minimum reprojection error. The brute force method has to do this for every combination of three detected markers in the image matched with every permutation of three LEDs of the target object. The pose that has a reprojection error smaller than a threshold value is selected.

The computational burden of the brute force algorithm becomes high, especially when the number of LEDs increases. Thus, the algorithm tries first to find the right correspondences with a predictive phase. The pose prediction uses current and previous pose and a constant velocity model. Then, the LEDs of the target object are projected into the camera image and the nearest LED is used as a correspondence pair. Finally, the correspondences are checked by computing the pose from every combination of three LEDs and checking if there are enough matches under the reprojection error threshold. According to their experiments, the mean orientation was less than one degree and the mean position error was less than one centimeter from one meter distance of the camera.

One should keep in mind when comparing the performance of different methods that the experiments differ on how they are executed and in which conditions. But, the results do give some sense of the current state in pose estimation accuracy.

3.3 ArUco Library in marker detection

Marker detection algorithms used in this work are based on ArUco module [57] inside the machine vision library OpenCV [58]. As mentioned in Section 3.2.1, there are some other augmented reality libraries which provide mostly similar algorithms for marker detection. ArUco was chosen from them because it is implemented using functions in OpenCV which has become very popular among machine vision people. Thus, it can provide a large inventory of other machine vision algorithms, and it is also easy to merge to other projects using the same library.

Marker detection mostly follows the usual path which was introduced in Section 3.2.1. In this section, more specific information regarding to this work's detection phase is going to be described.

First of all, ArUco uses 2D barcode markers instead of template markers because they provide less false positives and their identification is faster when the amount of markers increases. Marker detection begins by transforming the image into binary form so that the important features like borders of the markers become highlighted. Three algorithms were compared including simple thresholding, adaptive thresholding and canny detector. The best of these proved in practice to be adaptive thresholding. Unlike the simple thresholding which uses a global threshold value, adaptive thresholding only calculates a threshold value for small regions of the image. Thus, when there are different lighting conditions in the image, adaptive thresholding can handle them better. However, one must be careful when choosing the region size.

Then, contours are found from the binary image. Contours are a curve of continuous points having similar intensity along the boundary. Some of the contours are rejected before marker identification. First, it is tested if the contours are rectangles by minimizing vertices using Douglas-Peucker algorithm [59]. If a contour is left with 4 vertices, the contour is thought to represent a rectangle. After that, too big and too small marker candidates are eliminated. Also, if marker candidates are too close to each other only the one with larger perimeter is held.

The utilized markers have 25 bits of which 10 provide the identification number from 0…1023. Other 15 bits are used for determining the orientation of the marker. The bit rows are formed by using a modified version of Hamming coding [48] where only the first five bits are used. One row consists of two data bits and their corresponding three parity bits like shown in Figure 6. The first parity bit is

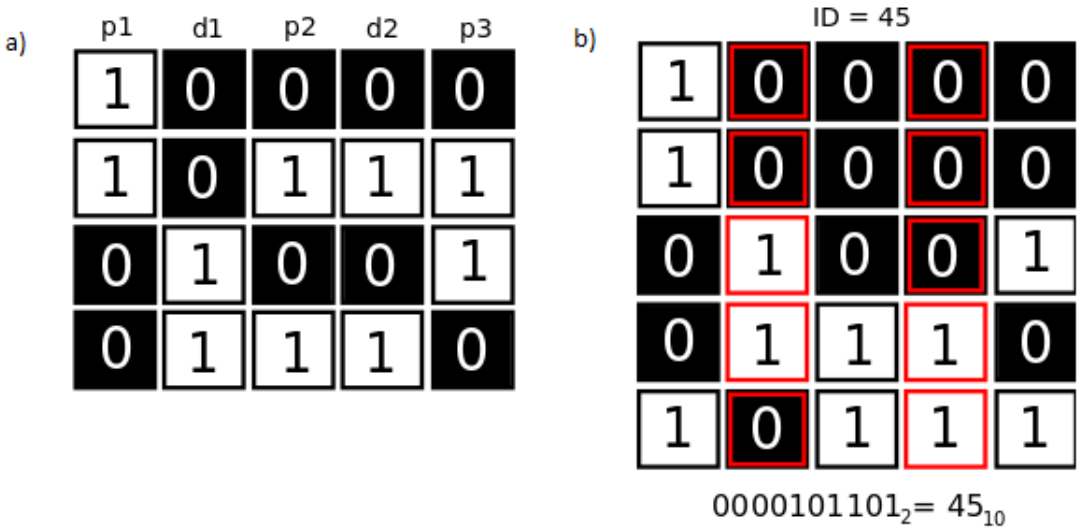

Figure 6: Coding of an ArUco marker. a) Four valid codes: p = parity bit, d = data bit. b) An example marker where each row consists of one of the four possible codes.

inversed, in contrary to the Hamming code, in order to avoid creating completely black markers.

The orientation is found by rotating the marker in the four possible orientations and checking if all the marker's rows match to valid codes in one of them. While this identification scheme seemed to work very well in practice, there are also more advanced and efficient methods available, e.g. [47]. And, it was noticed that identification number 1023 will provide a symmetric figure and its orientation cannot be unambiguously determined.

After identification, the corners of the markers are refined in sub-pixel accuracy. The algorithm is based on an assumption that a vector from any edge point $\boldsymbol{p}$ in the neighborhood of the true corner point $\boldsymbol{q}$ is orthogonal with the intensity gradient at $\boldsymbol{p}$. Thus, a dot product is used because it also gives a zero value when point $\boldsymbol{p}$ is on a flat region (e.g. middle of the black border) having a zero intensity gradient. However, in real life there is never an ideal sharp corner in camera image so the goal is to find an estimated corner point that minimizes the quadratic sum of dot products in the neighborhood. A spatial weighting function $w(\boldsymbol{p})$ is also added to give more weight to pixels near the center of the neighborhood. The algorithm can be formed as follows:

$$\arg\min_{\widehat{\boldsymbol{q}}\in\mathbb{R}^2} \sum_{\boldsymbol{p}_i\in\mathcal{N}} \left(w(\boldsymbol{p}_i)\nabla \boldsymbol{I}(\boldsymbol{p}_i)^T(\widehat{\boldsymbol{q}}-\boldsymbol{p}_i)\right)^2, \tag{7}$$

where $\nabla \boldsymbol{I}(\boldsymbol{p}_i)$ is an image gradient at point $\boldsymbol{p}_i$ in the neighborhood $\mathcal{N}$. By taking the derivative with respect to $\widehat{\boldsymbol{q}}$ and setting it to zero, a linear equation can be found:

$$\boldsymbol{A}\widehat{\boldsymbol{q}} = \boldsymbol{b}\,, \tag{8}$$

where

$$\boldsymbol{A} = \sum_{\boldsymbol{p}_i\in\mathcal{N}} w(\boldsymbol{p}_i)\nabla \boldsymbol{I}(\boldsymbol{p}_i)\nabla \boldsymbol{I}(\boldsymbol{p}_i)^T$$

$$\boldsymbol{b} = \sum_{\boldsymbol{p}_i\in\mathcal{N}} w(\boldsymbol{p}_i)\nabla \boldsymbol{I}(\boldsymbol{p}_i)\nabla \boldsymbol{I}(\boldsymbol{p}_i)^T\,\boldsymbol{p}_i$$

A sub-pixel corner location estimate is obtained from the solution. The above algorithm has its bases on the theory by Förstner et al. [60]. The matrix $\boldsymbol{A}$ is so called windowed second moment matrix which can be found in many image processing algorithms e.g. in Harris corner detection.

3.4 Natural feature-based pose estimation

There is also option to get the 2D image and 3D world point correspondences without the help of fiducials. This is accomplished by keypoint matching [24]. Keypoints are distinctive features than can be extracted from the images. Each keypoint is assigned with a descriptor that encapsulates the characteristics of it and its neighborhood. Finally, the keypoints are matched against reference points whose positions in the reference frame and whose descriptors are known. When the correspondences are found, the pose estimation is in principle the same as with markers.

Feature point extraction is a complex operation because the keypoints should be invariant to scale, viewpoint and illumination changes [61]. The well-known SIFT (Scale Invariant Feature Transform) [62] detects the keypoints using a difference-of-Gaussian (DoG) function in scale-space and finding the extrema. DoG is a close approximation and more efficient to the scale-normalized Laplacian of Gaussian which has shown to provide the most stable keypoints compared to a range of other algorithms. SURF [63] and FAST [64] are two popular alternatives to keypoint extraction which are developed for computational efficiency in mind.

For each keypoint, a descriptor is usually calculated which is used for matching it against those in the database. The descriptor characterizes the keypoint and its neighborhood so that it is distinctive and invariant to viewpoint and illumination changes [61].

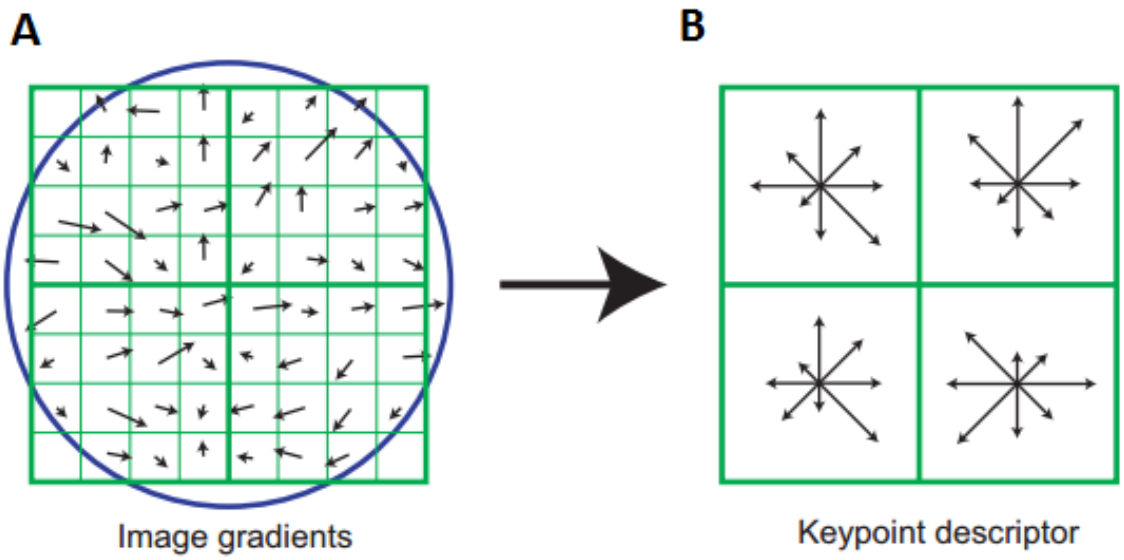

Figure 7: SIFT uses orientation histograms as keypoint descriptors [62]. A) Gradients are calculated within a keypoint neighborhood. B) There are four orientation histograms with 8 bins that are concatenated to build the descriptor.

For example, the descriptor in SIFT is build up from several steps. A scale and orientation is calculated for the keypoints in order to correct neighborhood of the keypoint in the image. The orientation is acquired by taking a histogram of gradient orientations in the sampled neighborhood and selecting the bins with highest peaks. Thus, multiple keypoints with different orientations might exist in the same location. Next step is to divide the corrected neighborhood in sub-regions and calculate the weighted gradient magnitude and orientation for each sample point. These vectors are then used to form orientation histograms which can be seen in Figure 7. The size of the descriptor consists of the number of orientation histograms and the number of bins they contain. In addition, there are also two intensity-based descriptor methods called BRIEF [65] and ORB [66] which are computationally more efficient than SIFT or SURF.

The idea of the keypoint matching is to compare the online extracted descriptor to the database and find the closest match. In AR applications, nearest-neighbor method is usually adopted. It is not however intuitive to compare two descriptors but several methods have been proposed [24]. For instance, the described SIFT uses Euclidean distance for the nearest neighbor search inside a k-d tree [63].

Despite the fact that using natural features in augmented reality is still a very new subject, there are interesting papers in the literature. Wagner et al. [67] showed that their natural feature based method can obtain a real-time performance even in mobile phones. They represent a modified SIFT where the feature extraction is done by FAST algorithm. They used 3x3 sized sub-regions of pixels with four bins in each of the orientation histograms, totaling a 36 element descriptors. The keypoint matching in k-d tree is changed to another tree representation they called as Spill Forest which is faster than the original. Keypoint outliers are first removed by a preprocessing phase and RANSAC [68]. The initial pose calculated by homography is further refined by Gauss-Newton minimization with M-estimator cost function.

The database of comparison features is built from known planar textured targets. Since FAST does not estimate a scale of a keypoint, Wagner et al. provide an image pyramid of the feature descriptors from all meaningful scale levels. They used different video sets to test the performance and calculated a reprojection error around one pixel. They estimated that this system could run 15 frames per second in a typical AR application. But, the CPUs of the phones today are much better than they were in 2010 when the paper was released. So, even better results are to be expected.

Additionally, Wagner et al. included a template tracker, also called as patch tracker, which reduces the computation time drastically and makes the system more robust. In template tracking the features in the reference image are projected to the input image according to the estimated pose, and they are searched inside a specific window size. The estimated pose is calculated from a linear motion model that takes into account the current and previous pose. It gives usually a good prediction because the time between two consecutive frames is quite small. However, a template tracker cannot survive by itself because it needs at least an initial pose estimate. Also, if the tracking fails, e.g. due to fast movement, it needs a

reinitialization. For that reason, it is quite common to use a combination of a natural feature based detection and a template tracker.

Natural features are also utilized in the active research area of visual SLAM (simultaneous localization and mapping) [69]. In visual SLAM, both the camera pose and the map are estimated simultaneously. Thus, no prepared models of the environment or the objects are needed. Needless to say, visual SLAM is a very complex problem, but the use of natural features is an essential part of it. The SLAM process itself is outside of the scope of this thesis.

The recognized work by Klein et al. [70] called Parallel Tracking and Mapping (PTAM) uses FAST algorithm to detect corners on four-level image pyramid. They also use a patch tracking as their main feature detection procedure. The system enables a real-time augmented reality in an unknown environment. However, it is adequate for small-workspaces only.

A recent paper published in 2015 by Mur-Artal et al. [71] proposes the most reliable and complete solution for monocular SLAM. It is based on the same ideas than PTAM but adds some new algorithms to further refine the system. From the perspective of natural feature matching, it changes the patch tracking scheme to ORB descriptors. The ORB features are used for all tasks, which includes tracking, mapping, relocalization and loop closing. Contrary to e.g. SIFT or SURF, ORB is fast enough to be used in a real-time visual SLAM application.

In their paper, they tested a few state-of-the-art visual SLAM systems using the TUM RGB-D benchmark database. It contains several video sequences of different setups. They reported an absolute trajectory root mean square error of their proposed system to be between $0.3 - 3.45$ cm. However, the benchmark database sequences that were considered to be unsuitable for monocular systems were discarded.

This emphasizes the problems of using natural feature based systems. It is known that they do not tolerate well in case of large changes in viewpoints, untextured environments or rapid movements, at least for now. Also, the scale is unknown with visual SLAM and there is no meaningful coordinate system in case virtual points from an external sensor needed to be projected on the image.

In this chapter monocular vision-based pose estimation methods were described. The main concentration was on marker detection but also the utilization of natural features was introduced. The following chapter will present how to measure the orientation of an object using inertial sensors.

4 Attitude estimation using inertial measurements

Inertial Measurement Unit (IMU) is a device containing a three-axis accelerometer and a three-axis gyroscope. It can be used to obtain a body's orientation in three-dimensional space respect to the Earth coordinate system, i.e. attitude. It has been traditionally used in inertial navigation of aircraft, ship and spacecraft. The development of the Microelectromechanical systems (MEMS) technology enabled smaller and less expensive IMUs to be available. Although low-cost MEMS IMUs are noisier and not as accurate as the traditional mechanical IMUs, the help of sensor fusion and calibration techniques have make them sufficient in many systems. Today, they can be found in mobile phones, drones and virtual/augmented reality applications. In this chapter, an introduction to the problem of attitude estimation using a MEMS IMU is given.

4.1 MEMS IMU

The advantage of using a MEMS IMU is that it can give accurate measurements even in fast motions. It is also immune to occlusion and it has high measurement rate. These are all complementary characteristics compared to the visual pose estimation methods. Hence, the combination of visual and inertial measurements is utilized in this thesis for attitude estimation.

A MEMS IMU measures proper accelerations via accelerometers and angular velocities via gyroscopes in 3D. The components are produced by micromachining [72]. Most of the gyroscopes utilize the idea of Coriolis acceleration which is an apparent acceleration caused by a rotating reference frame. The structure of the gyroscope consists of vibrating mechanical elements, and there are actually no rotating parts. The vibrations are sensed by capacitive, piezoresistive or piezoelectric detectors. Resolution, drift, zero-rate output (bias) and scale factor (gain) are typically used to compare the performance of the gyroscopes.

On the other hand, accelerometers use in general a proof mass which is suspended by beams anchored to a fixed frame. The mathematical model of an accelerometer can be modeled by a second-order mass-spring-damper system. Two common transduction techniques are capacitive and piezoresistive. Accelerometer measures proper accelerations that are accelerations relative to a free fall. This means that if the accelerometer is at rest on the surface of Earth, it will measure an acceleration of $1g \approx 9.81\ m/s^2$ away from the center of the Earth. There are similar specifications for accelerometers than there are for gyroscopes. The most important ones are resolution, offset (bias), sensitivity (gain), operation range and bandwidth.

Both accelerometers and gyroscopes suffer from bias and gain errors. They also tend to shift in time mainly due to temperature changes. And, even if the environment would be in a fixed-temperature, the device itself will heat. To minimize the harmful bias and gain errors, calibration is needed. Next, the main ideas in MEMS IMU calibration are presented.

4.2 Calibration of IMU

Calibration is needed because there are systematic errors in the output of the sensors. Bias and gain model is often used to compensate the measured raw sensor reading to estimate the actual value. Sometimes the misalignments of the three orthogonal sensors are also estimated [73], [74]. Calibration is done by comparing the raw measurements to some known reference values and minimizing the differences between them.

In case of an accelerometer, this reference value is often the Earth's gravity. For uniaxial accelerometer, the two unknown parameters, bias and gain, can be acquired by putting the device in two known angles in respect to Earth and solving the linear equations [75]. This also implies that for tri-axial accelerometer a minimum of six equations must be provided.

There are different methods used in the literature to obtain the reference orientations. A work by Renk et al. [73] uses a slowly rotating robot arm to measure several thousands of orientations for IMU. Then, an iterative minimization algorithm is used in least-squares manner to obtain the unknown parameters. In turn, Kim et al. [74] utilized an optical tracking system where LEDs attached on the IMU are detected to get the orientation.

In the methods above, specific hardware is needed. It is however possible to use the mathematical model of gains and biases with the assumption that the length of the tri-axial accelerometer vector corresponds to the gravity vector [76]. Then, estimates of the gains and biases can be calculated by putting the IMU in six different, but unknown, stationary orientations and iteratively minimizing the error between the gravity vector and the length of the estimated acceleration vector.

Gyroscopes are calibrated usually by attaching the device onto a rotating platform with known angular velocity. The platforms vary from rotating tables [77] to robotic arms [78]. However, methods needing no equipment have been presented also for gyroscopes. Fong et al. [79] utilizes an already calibrated tri-axial accelerometer sensor together with any kind of orientation algorithm which uses the integrals of gyroscope measurements. Measurements from the IMU are taken during a motion sequence consisting of arbitrary rotations and stationary pauses.

It has been stated that the most significant errors in inertial navigation are bias errors, and that an angular random walk is the main noise term [80]. It is safe to assume that this applies also in attitude estimation. Also, the temperature variations on biases and gains are significant for MEMS IMUs and they should be taken into account [80].

One solution to tackle the drifting gyroscope biases is to estimate them online. In the work by Hyyti et al. [78] an online gyroscope bias estimation is applied for attitude estimation using a variable measurement covariance of accelerations in an extended Kalman filter. They also use a linear temperature model in the calibration procedure. They showed experimentally that their proposed attitude estimation method outperforms comparable state-of-the-art algorithms when the gyroscope measurements contain bias. This is also the reason why the method was utilized in

this thesis. More information about the method in question is presented in Section 4.3.2. The calibration process is also described later in Section 5.3.4.

4.3 Attitude estimation

Attitude describes a three-dimensional orientation of a rigid-body. Before going into details about attitude estimation, it is very beneficial to understand that there are several different ways to present a body's orientation. One should also be aware of possible problems regarding them because otherwise even system failures could occur.

4.3.1 Representations of attitude

Remember how the attitude was parametrized in (2) in Section 2.2 by consecutive rotations about z-, y- and x-axis by the angle of ψ, θ, φ (yaw, pitch, roll). We now derive the relation between the three parameters and the rotation matrix, also called as direction cosine matrix (DCM):

$$\boldsymbol{R} = f(\psi, \theta, \varphi) = \begin{bmatrix} \theta_c\psi_c & -\varphi_c\psi_s + \varphi_s\theta_s\psi_c & \varphi_s\psi_s + \varphi_c\theta_s\psi_c \\ \theta_c\psi_s & \varphi_c\psi_c + \varphi_s\theta_s\psi_s & -\varphi_s\psi_c + \varphi_c\theta_s\psi_s \\ -\theta_s & \varphi_s\theta_c & \varphi_c\theta_c \end{bmatrix}. \tag{9}$$

In (9), the subscript "c" refers to cosine and "s" to sine. Conversely, the angles of rotation can be acquired from the rotation matrix:

$$\begin{aligned} \psi &= atan2(R_{21}, R_{11}) , \\ \theta &= \arcsin(-R_{31}) , \\ \varphi &= atan2(R_{32}, R_{33}) , \end{aligned} \tag{10}$$

where $atan2$ is the inverse tangent function recovering the right quadrant of the computed angle.

Rotation matrix is an elementary way to represent an attitude and it is computationally stable. It is effectively the same as a proper orthogonal matrix [16]. Orthogonality gives six constraints on the nine parameters of the matrix. Hence, rotations can have at most three degrees of freedom and they can be described by at most three independent parameters.

When reading any publications, one should be aware that there are two practices for the transformation of coordinates [16]. A passive description changes the coordinate system in which the object is presented. On the other hand, active transformation rotates the object in the described coordinate system. The difference shows up just minimally in the calculations as the rotation matrices are the transposes from one description to the other.

As already mentioned, rotation matrix can be constructed by three consecutive rotations about the axes [16]. These rotation angles are called commonly as Euler angles. Thus far, one of the asymmetric representations (z-y-x) has been shown

but there exists actually twelve sets of Euler angles, six symmetric (e.g. z-x-z) and six asymmetric. So, the order in which the rotations are made has an effect on the calculation between the rotation matrix and Euler angles.

Although it is interesting to characterize a unique rotation by just three parameters, the practice suffers from singularities. It has been shown that it is actually impossible to parametrize a rotation globally by just three parameters without singularities [81]. In some applications, this representation is however sufficient if the rotations are restricted to never reach them.

But, there exist an attitude representation that uses only four parameters and it is globally non-singular [82]. The parametrization is called as a quaternion. It has its basis on the Euler's theorem which states that any attitude can be described by a rotation about a single axis. The axis-angle representation can be formed as

$$\boldsymbol{\phi} = \phi \boldsymbol{e}\,, \tag{11}$$

where $\boldsymbol{e}$ is a three-dimensional unit vector

$$\boldsymbol{e} = \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} \tag{12}$$

and ϕ is a scalar angle. Furthermore, the unit quaternion $\bar{q}$ can be derived from (11) and (12) as

$$\bar{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{q} \\ q_4 \end{bmatrix} = \begin{bmatrix} \boldsymbol{e} \sin\left(\frac{\phi}{2}\right) \\ \cos\left(\frac{\phi}{2}\right) \end{bmatrix}. \tag{13}$$

Also, the quaternion has to obey a unit length constraint:

$$|\bar{\boldsymbol{q}}|^2 = |\boldsymbol{q}|^2 + q_4^2 = 1\,. \tag{14}$$

Finally, the relation between the rotation matrix and the quaternion can be formulated as

$$\boldsymbol{R}(\bar{\boldsymbol{q}}) = (q_4^2 - |\boldsymbol{q}|^2)\boldsymbol{I}_{3x3} - 2q_4[\boldsymbol{q}\times] + 2\boldsymbol{q}\boldsymbol{q}^T\,, \tag{15}$$

where $\boldsymbol{I}_{3x3}$ is an identity matrix and $[\boldsymbol{q}\times]$ is the cross-product matrix [82]

$$[\boldsymbol{q}\times] = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix}. \tag{16}$$

Some other useful parametrization are the Rodrigues parameters and the Modified Rodrigues parameters. More information about them is represented e.g. in [16]. The attitude estimation algorithm in this work uses a DCM representation with z-y-x convention (9) and it is described in the next section.

4.3.2 Methods in attitude estimation

Low-cost MEMS IMUs are affected by significant amount of noise. Other considerable challenges are the bias and gain errors that drift over time, and are affected by temperature changes. Hence, an attitude estimation algorithm that fuses accelerometer and gyroscope measurements is needed. The algorithm should also correct the gyroscope bias online.

According to Hyyti et al. [78] there are only few accurate algorithms that can estimate gyroscope biases online with only triaxial gyroscopes and accelerometers. As only the work by Hyyti et al. was freely available, it was selected for the work. The method uses a DCM-based adaptive extended Kalman filter (EKF) for the attitude estimation.

The estimate of the attitude is achieved by integrating the angular velocities of the gyroscopes. Because it will eventually drift, it is fused with accelerometer readings of the gravity. According to [78], the attitude is estimated only partially because the heading is acquired directly from the bias free gyroscope measurements and it will drift over time. The gyroscope neither knows anything about the heading in the global/navigation frame. Section 5.3.5 will describe how the heading estimate is corrected in this work.

As stated in [78], there are six states in the implemented EKF. Three of them are the last row elements of the DCM which describe the direction of gravity. Other three states are the gyroscope biases of each axis. Discrete nonlinear state-transition function is presented in (17). It is discretized by the Euler method. The state vector $\boldsymbol{x}_k = [C_{31}\ C_{32}\ C_{33}\ b_x^\omega\ \ b_z^\omega\ \ b_z^\omega]^T$ is updated due to a rotation caused by angular velocity measurements and an additional rotation due to biases. Angular velocities are given as an input $\boldsymbol{u}_k$ for the system.

$$f_k(\boldsymbol{x}_k, \boldsymbol{u}_k) = \begin{bmatrix} \boldsymbol{I_3} & -\Delta t[\boldsymbol{C}_3 \times]_k \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_3 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} \Delta t[\boldsymbol{C}_3 \times]_k \\ \boldsymbol{0}_{3\times3} \end{bmatrix} \boldsymbol{u}_k \tag{17}$$

The matrix $[\boldsymbol{C}_3 \times]$ is defined as

$$\begin{bmatrix} 0 & -{}^W_B C_{33} & {}^W_B C_{32} \\ {}^W_B C_{33} & 0 & -{}^W_B C_{31} \\ -{}^W_B C_{32} & {}^W_B C_{31} & 0 \end{bmatrix}, \tag{18}$$

$$W = world\ frame,$$
$$B = body\text{-}fixed\ frame,$$

where ${}^W_B C_{ij}$ are elements from the DCM. The matrix $[\boldsymbol{C}_3 \times]$ is derived from the fundamental dynamic model of rotation where the DCM is multiplied with the angular velocity tensor. But, this time only the last row is updated. The measurements are acquired from the accelerometers which should give the direction of gravity if there are no external accelerations. Naturally, the

assumption is not true while the head moves, but it is taken into account in measurement covariance shown later in this section.

Process noise is assumed to be zero mean Gaussian white noise with a covariance $\boldsymbol{Q}_k$. It is assumed that there is no cross-correlation between states and the covariance matrix is formulated as

$$\boldsymbol{Q}_k = \Delta t^2 \begin{bmatrix} \sigma_{C_3}^2 \boldsymbol{I}_3 & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & (\sigma_b^{\omega})^2 \boldsymbol{I}_3 \end{bmatrix}, \tag{19}$$

where $\sigma_{C_3}^2$ is the variance of the DCM state prediction. The value is approximated as the variance of the gyroscope measurements since it is the main factor affecting it. The other parameter $(\sigma_b^{\omega})^2$ is the variance of the bias state prediction and it is just set to a small value because the gyroscope bias is assumed to change slowly.

As already mentioned, the measurement covariance $\boldsymbol{R}_k$ includes the effect of external accelerations. It has the following form:

$$\boldsymbol{R}_k = \left(\left\| {}^b\boldsymbol{a}_k \right\| \sigma_a^2 + \sigma_f^2\right)\boldsymbol{I}_3, \tag{20}$$

where $\left\| {}^b\boldsymbol{a}_k \right\|$ describes the magnitude of the estimated non-gravitational acceleration. These accelerations are derived from the relation between the predicted direction of gravity and the measurements of the accelerometer. The parameter σ_f^2 represents the variance of the acceleration measurement noise. The parameter σ_a^2 is on the other hand used as a scaling factor for the non-gravitational acceleration magnitude. The varying measurement covariance matrix $\boldsymbol{R}_k$ implies that when the estimated magnitude of external forces increases, the trust is more on the predicted state. The fundamentals of the bias estimator can be also found from the relation between the predicted direction of gravity updated by angular velocities and from the measurements of the accelerometers.

This chapter presented the use of MEMS IMU in attitude estimation. It was noticed that a proper calibration and sensor fusing the gyroscope and accelerometer measurements are essential for an accurate estimation. This chapter also ends the theory part of this thesis. Next chapter will focus on the practical work describing the implementation of the prototype AR system.

5 Implementation of the prototype

The target of the research work is to help a forest machine operator to easily interpret the input from a machine. Hence, an augmented reality application, which shows virtual information as a part of the real scene, is implemented for a forest machine research platform [83]. Both the hardware and software solutions are described in this chapter. It is not intended to be only a list of what have been done but also the reasons behind the choices are presented. This chapter will begin with a top level overview of the system and then going to the more detailed information.

5.1 System overview

The system augments video camera images by the data from a lidar and from the pose measurements of the forestry crane. For head/camera pose calculations it uses a RGB camera and a MEMS IMU. This information is synchronized and fused together. The result is a pose estimate which is used to project the virtual data in world frame on to the coordinates in the camera image. The overview of the system can be seen on the next page in Figure 8.

The pose estimate of the calibrated camera is obtained by detecting 2D barcode markers inside the tractor cabin. The markers are distributed around the cabin and the locations of the marker corners are calculated beforehand with an automatic calibration procedure described later in Section 5.3.2.

As found in Chapter 3, the reasons for using black and white markers in pose estimation are following. First of all, it makes sense to utilize the same camera that captures the live video in pose estimation as well. Markers have also been successfully used in earlier research projects in the field of augmented reality. In addition, 2D barcode markers are low-cost, easy to set up and they give a scale. Lastly, ArUco library provides some freely available algorithms to utilize in this work (see Section 3.3).

As stated in Chapter 4, inertial measurement unit is used alongside the camera because it has complementary characteristics. IMUs are not affected by occlusions or lighting conditions and they give accurate results in fast movements. IMU measurements are collected by a microcontroller and they are sent to the main computer. The data consists of accelerations and angular velocities measured by the 3D accelerometer and 3D gyroscope respectively. The attitude can be then calculated by the algorithm presented in Section 4.3.2.

In order to combine the camera and IMU orientation, they should match timewise. This is accomplished by taking timestamps and estimating a delay between IMU and camera. More information of the synchronization is provided later in Section 5.3.7. The orientation is fused with the camera orientation after the IMU coordinates are rotated to match the camera coordinates.

The lidar is constructed by using a 2D laser scanner. It is attached to the boom of the tractor so that it scans 180 degrees from ground to sky. The point cloud data

given by the lidar is constantly updated. The boom has two markers in the joints which are visible in the lidar data. Together with the instrumented boom, the location of the joints can be calculated [83]. Furthermore, the tool attached to the boom is instrumented with IMUs and its location can be extracted [84]. All this information is rendered to the camera image through coordinate transformations.

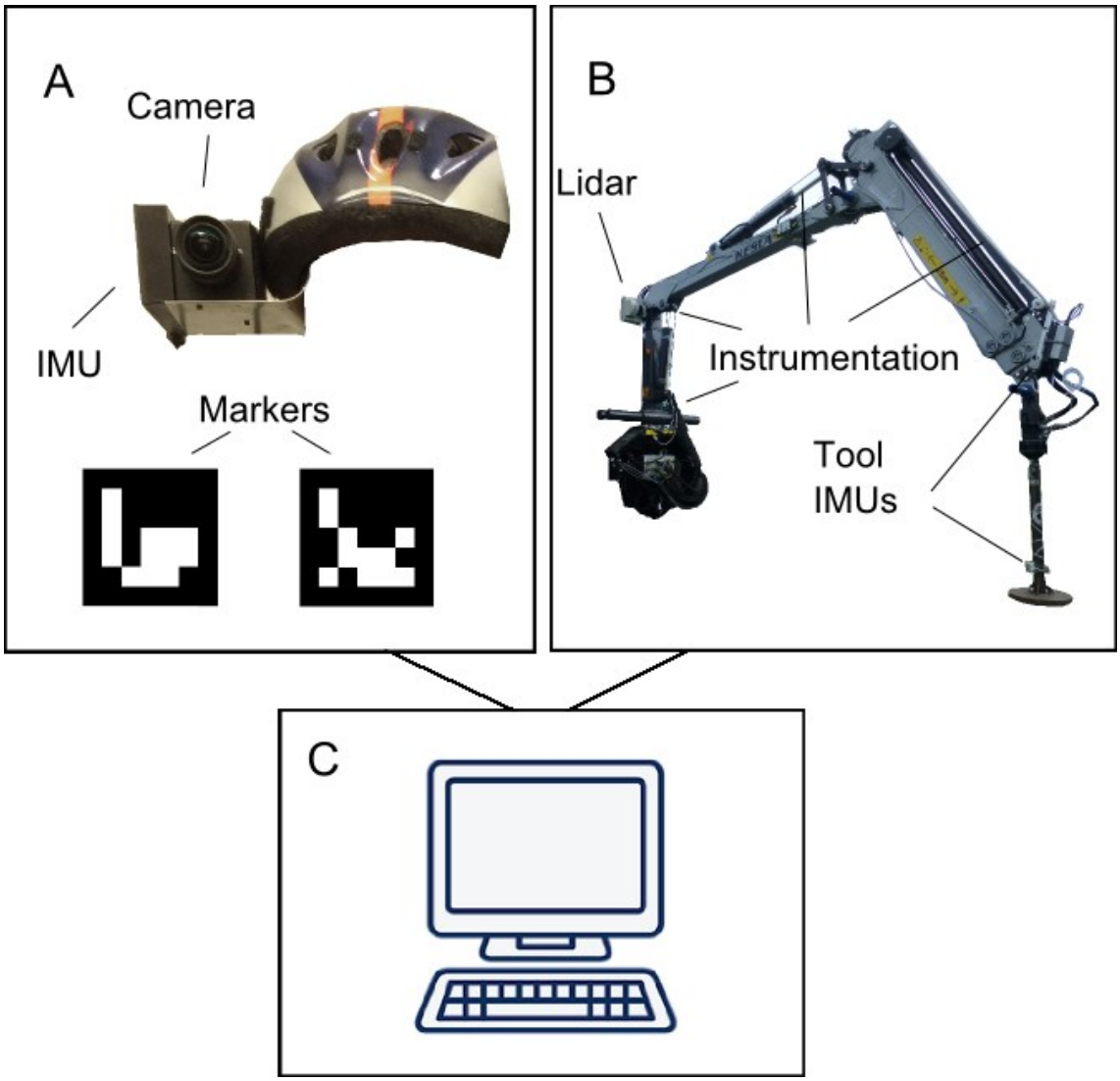

Figure 8: System overview. A) Camera/IMU device is attached to a helmet and it sends pose measurements. B) Forestry crane has a lidar on the basis of the boom. The crane is also instrumented to get positions of its joints [83] and the tool's position is acquired by IMUs [84]. The virtual data comes from all these sensors. C) A PC is used to collect and process all the incoming data and show augmented reality on the display.

5.2 Hardware

In the heart of the system's hardware are a machine vision camera together with markers, IMU with a microcontroller, and a PC. These components are needed to measure the pose of the camera and render the live view with virtual data. The data chosen to be augmented consists of a 3D point cloud and a wireframe of the crane. The point cloud data comes from the measurements of a lidar (2D laser scanner attached to the boom, see Figure 8). The wireframe model on the other hand is formed from the instrumentation of the forestry crane and from two IMUs relative to the tool. Next, the hardware is described in more detail. Also, the rationalism behind the choices is embraced.

5.2.1 Camera

The machine vision camera used in this work is a DFK 41AU02.AS from the Imaging Source [85]. It has a CCD (charge-coupled device) sensor that transforms the amount of light hitting to pixels into a digital value. The camera produces a resolution of 1280x960 and pushes 15 frames per second of Bayer8 format. And, it has a USB 2.0 connection.

Bayer8 format gives raw intensity data of a Bayer filtered image and each pixel has a value from 0-255 (8-bits). Bayer filtering [86] is used in many digital cameras to get color images. It is a method where a pixel has a color filter on it (red, green or blue) thus passing only specific range of wavelengths into the pixel. The filters are put in a pattern where green filters consist of 50 % of the sensor area and red and green filters cover 25 % each. Demosaicing the filtered values into a color image can be done e.g. by interpolating the values in the neighborhood.

The chosen camera had been used before in another machine vision project and thus there was already available software for frame grabbing and transforming images from raw format to color images in a Linux environment. The connection to a PC was also effortless because of the USB-format and Video4Linux2 driver.

The selected camera was easily mountable because it had a cubic shaped body with screw holes. One feature missed was an external triggering possibility which would have helped in the synchronization of the IMU measurements and camera images. But, the synchronization could be also done in software side so it was not a rigorous specification.

Apart from camera, also the camera lens had to be chosen. Because the demonstration should be as realistic as possible, the lens was chosen to be a fish-eye lens. More specifically, Tamron 13FM22IR was selected [87]. It has a 118.6° horizontal angle of view and 90.0° vertical angle of view. A positive effect of a wide-angle lens is that more markers are visible in a single image. On the contrary, the projection errors are larger than in conventional lenses [43].

5.2.2 IMU and microcontroller

The chosen inertial measurement unit was MPU-6050 from InvenSense [88] on a breakout board acquired from SparkFun. The IMU consists of 3-axis gyroscope and a 3-axis accelerometer and has a size only of 4x4x0.9 mm. It has three 16-bit analog-to-digital converters for both gyroscopes and accelerometers. User-programmable gyroscopes ranges are from $\pm$250°/sec to $\pm$2000°/sec. The measurement ranges for accelerometers are from $\pm$2g to $\pm$16g. In this work $\pm$500°/sec and $\pm$8g was used. The communication between IMU and application processor is performed using I^2C (Inter-Integrated Circuit) at 400 kHz. Moreover, there is an onboard temperature sensor.

MPU-6050 is a widely used IMU among hobbyist and for that reason there are already software libraries available [89] which eases the set up. It is a low-cost unit and cannot compete with more costly ones [78]. It was still evaluated to be accurate enough for this work's purposes. The onboard temperature sensor was one requirement because it enables temperature-based calibration to be made. This is important since the chip itself heats in use and the temperature of the environment can also change.

The highest sample rate possible is 1 kHz with this IMU device. However, a sample rate of 500 Hz was used because taking more measurements increase the computational burden and getting a measurement every 2 ms was thought to be quite enough.

The microcontroller of this work was selected to be Teensy 3.1 from PJRC [90]. It has a 32 bit ARM Cortex-M4 72 MHz processor. Communication with a PC is carried out with USB connection. Naturally, Teensy 3.1 also contains an interface for I^2C communication which is necessary to read the data from the MPU-6050 unit.

Teensy 3.1 is compatible with Arduino software & libraries [91] which were utilized in this work to get the raw measurements read. The board has also pins that helped the early testing to be executed on a solderless breadboard. After the code and wiring had proved to be working, the IMU and the microcontroller were soldered on a printed circuit board. Following section will describe how the circuit was designed and implemented. It also presents the final camera/IMU attachments with clarifying figures.

5.2.3 Printed circuit board

Printed circuit board (PCB) is a platform that electronically connects components and it is manufactured by etching a laminated copper sheet. The components are soldered on the PCBs pads or plated through holes and they are connected by copper traces. The design of the circuit board can be send to manufacturers or if necessary materials are on hand, they can be homemade. The latter option was used in this work.

The design of the PCB was done by using CadSoft's EAGLE 7.5.0 [92]. It has both the schematic editor for documentation purposes and a layout editor for actual

board design. Design of the circuit was fairly simple as can be seen in the schematic in Figure 9. Only other components besides IMU and microcontroller were two 4.7 $k\Omega$ resistors and one 15 nF capacitator.

The layout design was thriven to have as small footprint as possible since it would be attached to the side of the camera. It also needed screw holes to be added to the design which are the circles marked in black. The plated through holes are marked in green and copper traces in blue. Only the areas in white color were etched away leaving the insulated material (soldermask) visible. The layout design can be seen in Figure 10 A.

After the layout was ready, it was time to implement the board. There are several sources in the web that have instructions on how to make a homemade PCB. In this work the pattern that was etched from the PCB was acquired by exposing a photosensitive laminated board to ultraviolet light. Before the exposure, the layout was printed on a transparent sheet which was put on the board. The image was developed using sodium hydroxide and after that it was rinsed in pure water. Then, the board was sunk in to the acid which was ferric

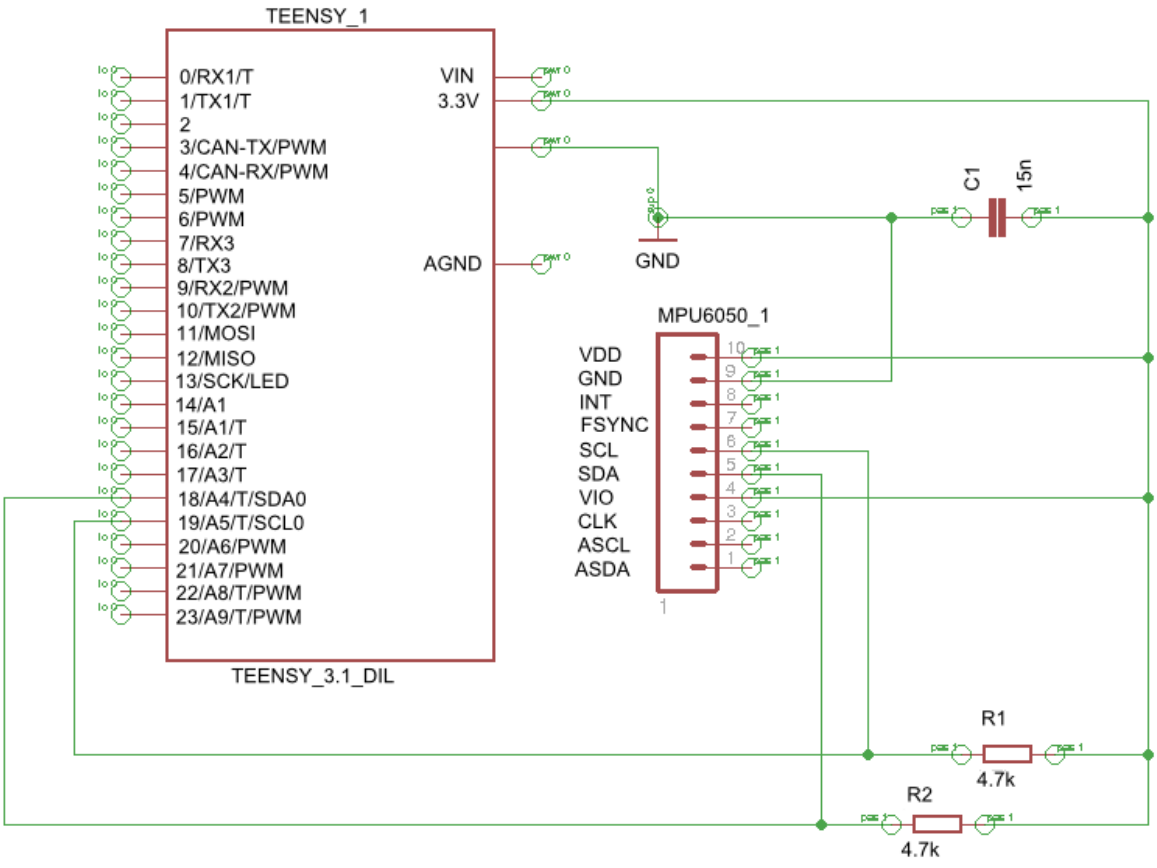

Figure 9: Schematic of the PCB.

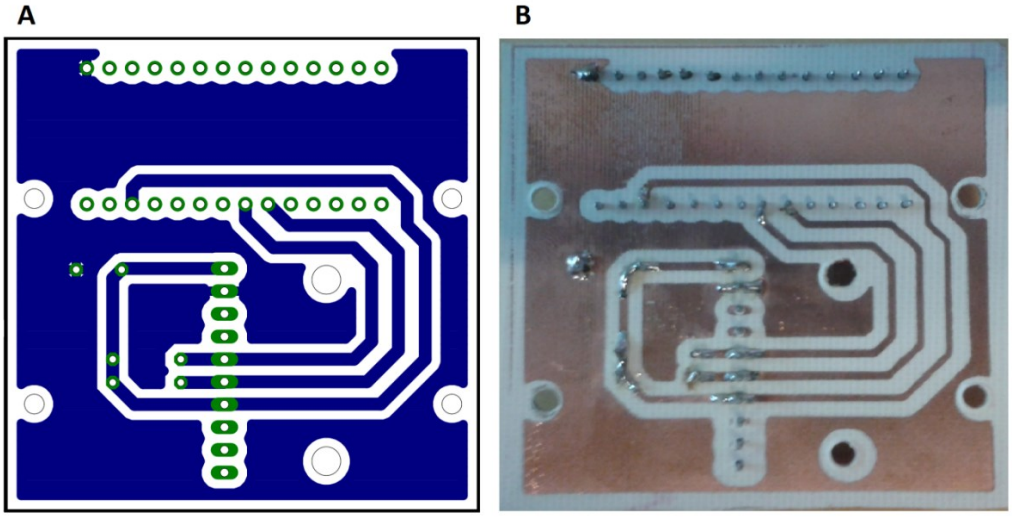

Figure 10: A) Layout of the PCB. B) Final product.

chloride. After a while the board was taken out and rinsed again in water. Finally the holes were drilled and the components were soldered on to place. The final circuit board can be seen in Figure 10 B.

The PCB was attached to the side of the camera rigidly since the IMU and camera should measure the same orientation. The finished product with PCB housing is presented in Figure 11.

Figure 11: Camera and the self-made IMU-microcontroller board.

5.2.4 Instrumentation of the work machine

The augmented reality setup was implemented to a forest machine research platform [83]. It consists of an agricultural tractor and a forestry crane. The tractor is a slightly modified Valtra T132 and the hydraulic forest crane is Kesla 305T. This research platform is ISO 11783 compliant and it enables to control the forestry crane using joysticks installed in the tractor.

The instrumentation of the crane [83] is however more interesting in the scope of this work, because the virtual data that is rendered comes from the sensors on the crane. The boom has four degrees of freedom (Figure 12): slew, lift, transfer and extension. A magnetic strip encoder measures the slew angle. Positions of the lift and transfer cylinders are measured with Posichron PST25 magnetostrictive sensors and a draw wire sensor measures the length of the extension.

A 2D laser scanner is attached on the top of the basis of the crane vertically so that it scans the environment from ground to sky. As the crane moves, the scanner gets new information which is used to constantly update a 3D point cloud. The lidar is a SICK LMS221 with a 180° scanning angle [93]. It is also used to deliver positional data from the last two joints of the crane, alongside the earlier mentioned sensors. This is accomplished by having two markers on the side of the

Figure 12: Tractor and forestry crane used in this work [83]:
a) laser scanner, b) markers, c) IMUs.

joints that the laser rays can hit. The implementation of the lidar is done in the research group and it is outside the scope of this work.

Lastly, the freely hanging tool is also instrumented to provide knowledge of the pose estimate. The tool in this work is just a swaying mass for testing purposes. The implementation by Kalmari et al. [84] is utilized in this work, and it uses two IMUs and Kalman filtering to obtain the sway estimates. One of the IMUs is attached on the tip of the boom and the other is fixed on the tool itself.

5.3 Software

While it is important to have the necessary hardware, the magic happens under the hood. Major part of this work concentrated on the software side. Main question was how to measure the pose of an augmented reality display in a forest machine. This chapter will give one approach to deal such a problem. Goal of this chapter is to describe the chosen methods and how they were implemented.

5.3.1 Camera Calibration

The introduction for camera calibration was already given in Section 3.1 so this section will only focus on how the calibration was done in this work. The tool that was mainly used was a Camera Calibration Toolbox for Matlab by Jean-Yves Bouguet [94]. It is a handy tool with a graphical user interface and it is based on solid theoretical background. Only modification for that toolbox was an automatic detection of corner locations which was developed earlier in the research group. The algorithm works even when there is lot of camera distortion.

The toolbox uses a work by Zhang [95] as its basis. The calibration procedure only requires known points in a single plane perceived from at least two different orientations. The algorithm's closed form solution relies on the homography between the model and the image plane. It also uses two constraints on the intrinsic parameters acquired from the knowledge that the rotation axes are orthonormal. The parameters obtained from the solution are used as an initial guess for the nonlinear minimization problem solved with the Levenberg-Marquardt algorithm [53]. The minimization function contains the error between the image points projected by the model and the known image points. The function can be formulated as

$$\sum_{i=1}^{n}\sum_{j=1}^{k}\left\|\boldsymbol{x}_{\boldsymbol{I}_{ij}}-\widehat{\boldsymbol{m}}\left(\boldsymbol{K},\boldsymbol{R}_i,\boldsymbol{t}_i,\boldsymbol{D},\boldsymbol{x}_{\boldsymbol{W}_j}\right)\right\|^2, \tag{21}$$

where $\widehat{\boldsymbol{m}}\left(\boldsymbol{K},\boldsymbol{R}_i,\boldsymbol{t}_i,\boldsymbol{D},\boldsymbol{x}_{\boldsymbol{W}_j}\right)$ is the projection of the point $\boldsymbol{x}_{\boldsymbol{W}_j}$ in image i according to the extrinsic matrix $[\boldsymbol{R}_i\boldsymbol{t}_i]$ and intrinsic matrix $\boldsymbol{K}$ followed by a distortion correction parametrized by $\boldsymbol{D}$.

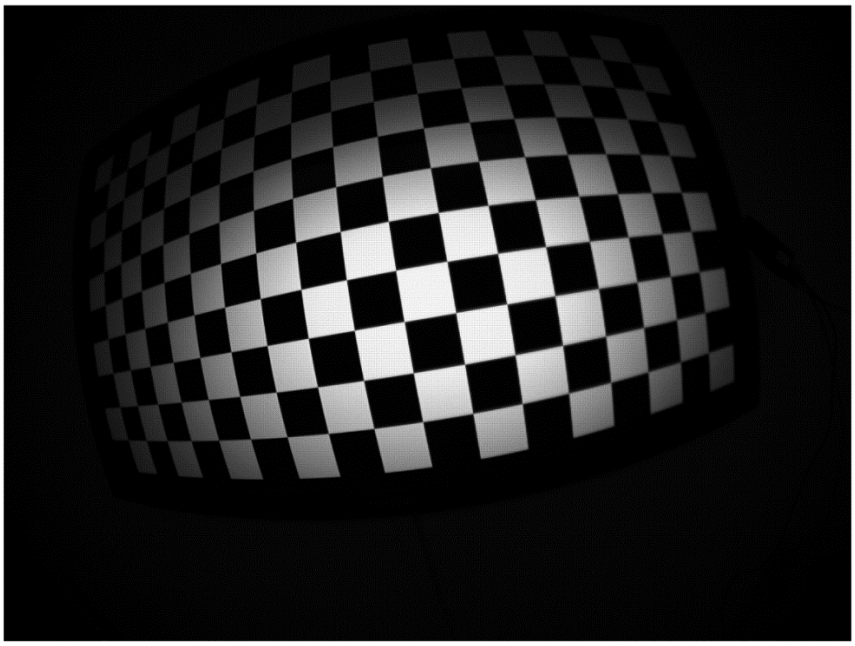

Figure 13: One of the images used in camera calibration.

The distortion model $\boldsymbol{D}$ is formulated as [43]

$$r(\theta) = k_1\theta + k_2\theta^3 + k_3\theta^5 + k_4\theta^7 + k_5\theta^9\,, \tag{22}$$

where r is the distance of an image point from the principal point, θ is the angle between the incoming ray and the principal axis and k_i is a distortion parameter.

In this work the calibration parameters were calculated from 15 planar chessboard images taken from different angles and positions. The chessboard figure is usually printed on a paper and attached to a planar board. From the experience, the paper will most often wrinkle and wave so the corner position can be inaccurate. By displaying the checkerboard figure on a flat television screen in its native resolution that kind of inaccuracy can be avoided (Figure 13).

Also, the fisheye lens had to be taken into account. Instead of using the normal pinhole model, the calibration was performed utilizing an equidistance model [43]. In the equidistance model the light rays do not travel straight like in pinhole model but curve towards the center of the camera. Fortunately, equidistance model is also implemented in the toolbox so now extra work had to be done in that part.

The accuracy of the calibration is evaluated by root mean square error (RMSE) of the reprojected corner points. RMSE of approximately 1 pixel is measured. It is evaluated to be acceptable error regarding to this work. The parameters of the intrinsic matrix are shown in (23). And, the distortion parameters of the equidistance model [43] are presented in Table 1.

$$\boldsymbol{K} = \begin{bmatrix} 487.08 \pm 2.72 & 0 & 639.5 \\ 0 & 488.15 \pm 2.74 & 479.5 \\ 0 & 0 & 1 \end{bmatrix}. \tag{23}$$

Table 1: Estimated distortion parameters.

Distortion parameters	**coefficients**
k_1	1
k_2	0.0219
k_3	-0.0021
k_4	0.0080
k_5	-0.0030

5.3.2 Calibration of markers

The pose calculation from the camera images requires that the corners of the markers are localized both in world coordinates and in image coordinates. The method of obtaining the corners in image was described in Section 3.3. Next, it is presented how the positions of the corners are found in the world frame.

One marker is not enough for this system because the operator should be able to view the outside world from any desired orientation inside the cabin of the forest machine. Also, the more points that are used in pose calculation the more error can be reduced. These are the reasons why multiple markers are used, and they should all be measured regarding to same reference coordinate system.

One possibility would be that the markers are precisely located on to some predefined and measured places. However, that method would require a cabin especially designed for this type of system. So, it was not the choice in this work. But, since we can get a pose between a single marker and the camera, it is possible to calculate the pose between two markers shown in the same image by using the rules of transformation matrices. The implemented method follows the paper by Siltanen et al. [96] with some modifications.

So, when two markers are detected, pose between them is calculated by taking the inverse of one of the transformation matrices between camera and marker:

$$^{m1}T_{m2} = {}^{c}T_{m1}^{-1}\,{}^{c}T_{m2}. \tag{24}$$

To make it a bit simpler, the origin of the reference frame is chosen to be one corner of a marker. The marker chosen is called as the main marker from here after. All the corners of the other markers should be projected on the chosen coordinate system. Thus, a transformation chain is needed from all the other markers to the main marker. And, because the errors will accumulate by each part of the transformation chain, the shortest path should be chosen.

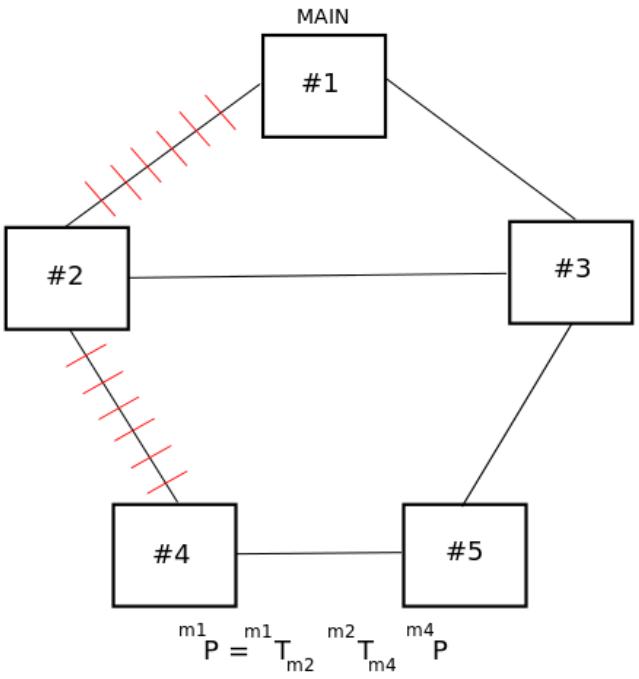

Figure 14: Illustration of the automatic corner localization procedure. One corner of marker #4 is transformed into reference coordinate system of the main marker.

An example case can be seen in Figure 14, where one corner location of marker #4 is transformed into reference frame. The lines in the figure represent that a marker pair has been detected. The shortest path is found by using Dijkstra's algorithm [97].

During the automatic marker localization process, the pose between detected marker pairs is constantly updated when new information comes available. Averaging translation part is trivial. But, since rotation matrices are characterized as orthogonal matrices with a determinant 1, the transformation matrices cannot be averaged elementwise. The translation and rotation parts are split, and the rotation matrices are turned into quaternions.

With quaternions there is a simple weighted average method

$$\overline{\boldsymbol{q}} \triangleq \left(\sum_{i=1}^{n} \omega_i \right)^{-1} \sum_{i=1}^{n} \omega_i \, q_{\boldsymbol{i}} \,, \tag{25}$$

but it has two known flaws [98]. First flaw is that the resulting quaternion is not anymore a unit quaternion. However, there exists an easy ad hoc method to fix it where the result quaternion is divided by its own norm. The second flaw comes from the fact that quaternions $\overline{\boldsymbol{q}}$ and $-\overline{\boldsymbol{q}}$ represent the same rotation. The way to solve this is to take a dot product of the two quaternions. If the dot product gives a

negative value, then the other quaternion is inversed. It should be noted that the algorithm works only if the separate quaternions are relatively close to each other. In practical implementation, the algorithm was turned into a recursive one.

Each weight for pose between marker pairs came from the marker detection algorithm. Quality of each estimated corner was evaluated and summed together. Because it would be quite expensive to calculate the residual for each corner, the evaluation was done by calculating the Euclidean distance between the original and the sub-pixel corner estimate. It seemed to give reasonable results in practice.

The calibration results can be seen in Table 2. The location of the main marker is known and a pose between the main marker and all the other markers are estimated. In this work seven markers are used for detection.

Table 2: Results from the automatic marker calibration.

Marker ID	**x (m)**	**y (m)**	**z (m)**	**roll °**	**pitch °**	**yaw °**
177 (main)	-	-	-	-	-	-
908	-0.083	-0.175	0.685	49.85	-2.67	1.87
299	0.141	-0.178	0.669	53.50	-0.95	-0.04
64	-0.523	-0.139	0.365	9.18	-1.89	84.11
760	0.710	-0.029	0.304	4.56	5.59	-86.61
838	-0.459	-0.464	0.790	8.22	1.03	98.19
341	0.455	-0.592	0.650	46.46	2.92	-87.63

5.3.3 Camera pose estimation

When all the corner positions are known in the same reference coordinate system, a pose can be calculated from any marker or combination of markers that are detected. The used algorithm is based on the one provided by the ArUco module [57]. Some modifications had to be made because the algorithm assumed a pinhole model and an equidistance model was used in this work due to the fisheye-lens. Also, the code of the original iterative algorithm was reimplemented because for some unknown reason it did not work.

For a single marker the algorithm is following. When the four corner points of the ArUco binary marker are detected from a camera image, the points are first undistorted. For a projective 2D to 2D point correspondence it is possible to calculate a 3x3 homography matrix that maps the world points into image points [45]. Extrinsic parameters can then be calculated using intrinsic parameters obtained from offline camera calibration and the homography matrix [95]. An iterative optimization algorithm (Gauss-Newton) is lastly used to refine the extrinsic parameters defining the pose between camera and world coordinates.

When more than one marker is detected in the camera image, first thing is to check whether the markers are on the same plane. If the matrix containing the corner points can be considered of rank 2 instead of rank 3, the corner points lie

approximately on the same plane [99]. It can be decided by comparing the singular values of the matrix. Then, the above mentioned homography-based method is used. If the points do not lie on the same plane, then the conventional DLT (see Appendix A) is used to give the closed-form solution further refined by the iterative algorithm.

One of the additional properties implemented on top of ArUco allowed different size of markers to be used. It is desired since the distance between the user and the markers differ inside the forest machine cabin. Before the automatic corner localization, the ID number and size of the used markers can be given.

Another practical feature considered about discarding markers that are under a threshold quality value. This is important because the pose is highly affected by the accuracy of the estimated corners. However, the selection of the threshold value is only based on experience.

5.3.4 IMU calibration

Overview of IMU calibration was already presented in Section 4.2. In this work, calibration method by Hyyti et al. [78] was utilized because it was easy to implement due to the open source code and it used a temperature based calibration method which is important for low-cost MEMS sensors.

The mathematical model for accelerometer calibration is based on the paper by Won et al. [76]. For every axis of the accelerometer gain and bias is calculated by placing the three-axis accelerometer on six different stationary orientations. The fundamental principle behind the method is that the gravity vector can be formed by summing the squared accelerometer readings of each axis. The advantage of this system lies on the fact that the orientations do not have to be known.

As already mentioned, in the proposed system the temperature was also considered. Because the accelerometer measurements seem to depend approximately linearly on the temperature (Figure 15), a linear model is used for every gain/bias parameter. The measurement model can be formulated as in [78]:

$$\begin{aligned} f_{meas,i} &= p_{gain}^{f_i}(T) f_i + p_{bias}^{f_i}(T), \\ \omega_{meas,i} &= p_{gain}^{\omega_i}(T) \omega_i + p_{bias}^{\omega_i}(T), \end{aligned} \tag{26}$$

$$p_{gain/bias}^{f_i/\omega_i}(T) = a_{gain/bias}^{f_i/\omega_i} T + b_{gain/bias}^{f_i/\omega_i} . \tag{27}$$

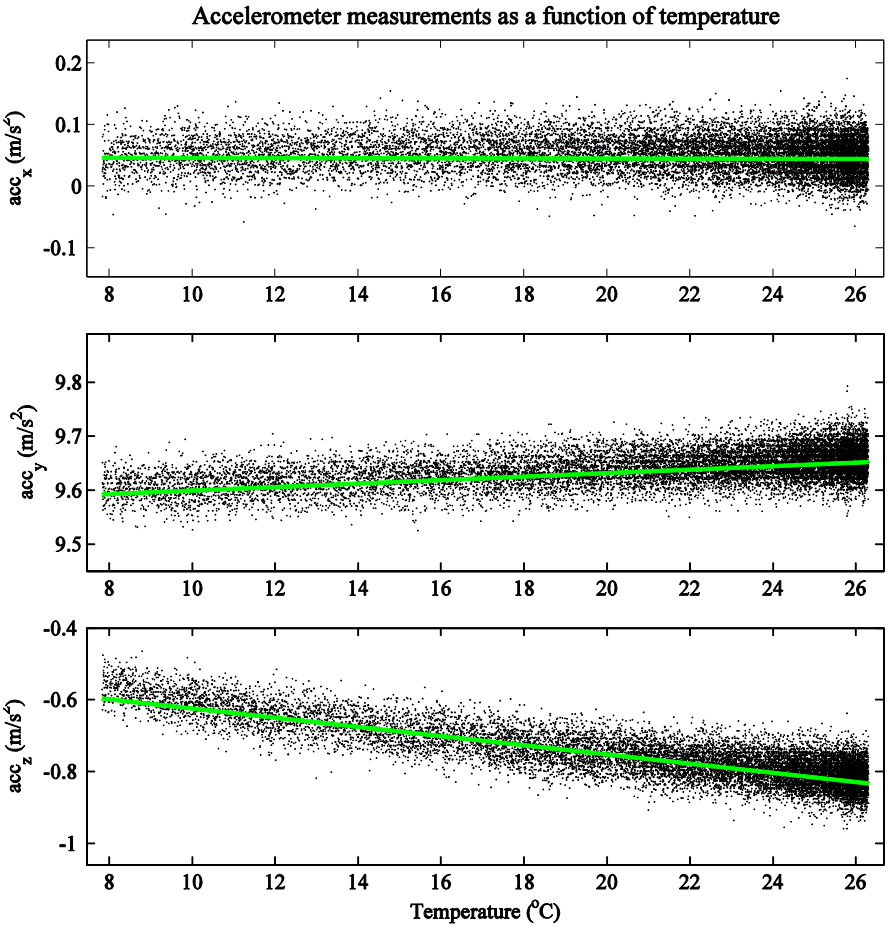

Figure 15: Temperature-based linear model fitted into accelerometer measurements.

In (26), $f_{meas,i}$ and $\omega_{meas,i}$ are the measured accelerometer and gyroscope readings in i:th axis $i \in \{x, y, z\}$ after the raw measurements f_i and ω_i are corrected by the calibration parameters. The temperature dependent linear model $p_{gain/bias}^{f_i/\omega_i}(T)$, in (27), has two parameters a and b.

The gyroscope gains can be estimated by comparing the measurements to a known rotational movement. But, as there was not a reference rotational platform easily available and the effect of gyroscope gain was thought to be negligible, the gain parameter was set to one. In the future, the availability of the camera pose estimate and calibrated accelerometer readings could be utilized as a reference. The bias of the gyroscope can be estimated in the same calibration process as the accelerometers because the gyroscope is also stationary and thus it should have a zero angular velocity.

In order to obtain the parameters of the temperature based linear model, IMU was first cooled and then let to gradually heat in all six orientations. Then, a linear line was fitted to each of the measurement sets using least squares method.

For calculation of gyroscope biases, the highest and smallest values of the line parameters were discarded and an average was taken from the rest of them. The

calibration parameters of the accelerometers were estimated in two different temperatures. Then, in order to get the parameters of the linear temperature model, a parameterized line was fitted into two points. For example, the parameters of the accelerometer's x-axis gain was calculated from (27) as

$$\begin{aligned} p_{gain}^{f_x}(T1) &= a_{gain}^{f_x} T_1 + b_{gain}^{f_x}, \\ p_{gain}^{f_x}(T2) &= a_{gain}^{f_x} T_2 + b_{gain}^{f_x}, \\ a_{gain}^{f_x} &= \frac{p_{gain}^{f_x}(T2) - p_{gain}^{f_x}(T1)}{T2 - T1}, \\ b_{gain}^{f_x} &= p_{gain}^{f_x}(T1) - a_{gain}^{f_x} T1 . \end{aligned} \tag{28}$$

The calibration parameters are listed in Table 3. By inspecting the accelerometer calibration values, it seems that the temperature change has a greater effect on the bias terms than on the gain terms.

Table 3: IMU calibration parameters.

Parameter	**x-axis**	**y-axis**	**z-axis**
$a_{bias}^{f_i}$	-0.0010182	0.0034701	-0.0040916
$b_{bias}^{f_i}$	0.2056372	-0.2403274	0.1735056
$a_{gain}^{f_i}$	0.0000659	0.0000512	-0.0000758
$b_{gain}^{f_i}$	0.9980415	1.0011017	1.0018270
$a_{bias}^{\omega_i}$	0.0210318	-0.0000472	0.0017301
$b_{bias}^{\omega_i}$	-1.1218503	0.2816288	0.6071337

5.3.5 Sensor fusion of camera and IMU

With the DCM-based IMU attitude estimation method [78] presented in Section 4.3.2, the direction of gravity can be accurately measured. However, an absolute heading cannot be found by using only MEMS gyroscopes and they tend to drift. But, they do provide accurate results in short time periods. What we would like to have is a system that combines the pose obtained from the camera images and the orientation calculated by the IMU.

While it is possible to also get position information from the IMUs by double integrating the non-gravitational accelerations, in this work that was left aside. One reason was that the non-gravitational accelerations are difficult to obtain accurately. Another issue is that accelerometers cannot sense if there is constant velocity movement.

The focus was on fusing the orientation obtained by the camera and IMU. It was thought that a similar method could be used in the fusion of heading which was done when fusing the gyroscopes and accelerometers. A separate Kalman filter was set up to not only to fuse the heading angles of camera and IMU but also the gravity component.

Six states of the filter consist of the elements in the first and last row of the DCM matrix. Because IMU and camera had different sensor rates, the prior estimate comes from the faster IMU. The gravity row vector is obtained by the method presented in Section 4.3.2. Besides, the heading row vector prior is estimated simply by updating it through the angular velocity tensor. With Kalman filter, a posteriori estimate is found by fusing the prior estimate and the measurements from the camera. In this separate Kalman filter a simple state transition is formed

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{x}_{k-1} + \boldsymbol{w}_k, \\ \boldsymbol{w}_k &\sim \mathcal{N}(0, \boldsymbol{Q}_k). \end{aligned} \tag{29}$$

The process covariance matrix $\boldsymbol{Q}_k$ is assumed to have no cross-correlation between the states. The variances of the state predictions are calculated on a period between two camera frames. They are approximated by measuring how much are the variances of the estimates given by the DCM algorithm, when the device is held stationary.

The measurement covariance matrix $\boldsymbol{R}_k$ is obtained similarly. The values are evaluated from the variances of the estimates provided by the camera pose algorithm. However, the variance is smaller when more markers are visible. So, the values are calculated for situations where one, two and three or more markers are visible. It was noticed experimentally that the estimated variances do not anymore drop significantly after more than three markers are detected.

The heading states have a high uncertainty values initially because they have no knowledge about the reference heading. This should be taken into account when setting the initial covariance matrix of the state estimates.

In order to obtain the whole rotation matrix from the gravity and heading vectors, a renormalization method is used [100]. The properties of the rotation matrix are desired to be maintained after the sensor fusion. The heading and gravity vectors $\boldsymbol{X}$ and $\boldsymbol{Z}$ should be perpendicular and thus having a zero dot product. The value from the dot product is the amount of error and it is reduced by cross coupling:

$$\boldsymbol{X} \cdot \boldsymbol{Z} = error, \tag{30}$$

$$\boldsymbol{X}_{orthogonal} = \boldsymbol{X} - \frac{error}{2}\boldsymbol{Z},$$

$$\boldsymbol{Z}_{orthogonal} = \boldsymbol{Z} - \frac{error}{2}\boldsymbol{X}.$$

The following step is to get the missing row of the rotation matrix. This is simply obtained by taking the cross-product of the two vectors. Finally, the row vectors are normalized to have a magnitude of one.

5.3.6 Coordinate systems

One important thing when dealing with rotations, are the coordinate systems. The idea in this work is to assign as few coordinate systems as possible so that numerical errors would be reduced when transforming from one frame to another. The world coordinate system *W* is selected to have the origin in the left upper corner of the main 2D barcode marker with Z-coordinate pointing up along the side border of the marker, and X-coordinate pointing in right along the upper border. The common right-hand rule is used in all coordinate systems.

There are two coordinate systems moving in the world frame: Camera coordinate system *C* and IMU's body-fixed coordinate system *B*. There is also image coordinate system *I* just like in Section 2.3 (Figure 4). Additionally, the points that are to render locate in the forestry crane coordinates *F*. It is needed to project the virtual points from the forestry crane coordinates to the camera coordinates. The chain is expressed mathematically as

$$\boldsymbol{x}_C = {}^{C}\boldsymbol{T}_W \, {}^{W}\boldsymbol{T}_F \boldsymbol{x}_F, \tag{31}$$

where ${}^{C}\boldsymbol{T}_W$ is a transformation from world frame to camera frame, ${}^{W}\boldsymbol{T}_F$ is transformation from forestry crane frame to world frame and $\boldsymbol{x}_C/\boldsymbol{x}_F$ are camera/forestry crane points.

From sensor fusion point of view, the rotation matrix inside ${}^{C}\boldsymbol{T}_W$ in (31) is the most important part. In order to use the fusion method described in the previous section, both the DCM rotation matrix from IMU and the rotation matrix ${}^{C}\boldsymbol{R}_W$ from the camera pose algorithm should be expressed in the same coordinate system. Thus, it is more convenient to transform DCM rotation matrix to match the camera coordinate system. And, it leads to the need of finding the rotation matrix between the body-fixed frame of IMU and camera frame. So, it is important that the devices are rigidly connected to each other in hardware level. In this work, the rotation matrix was constructed by multiplying the estimates from each device together at a stationary moment:

$${}^{C}\boldsymbol{R}_W \, {}^{W}\boldsymbol{R}_B = {}^{C}\boldsymbol{R}_B. \tag{32}$$

More sophisticated methods are available and they could be used to get better results, e.g. [20].

One obvious problem arises from the fact that the main marker will not be stationary in reference to the earth because the cabin of the forest machine moves. So, also the rotation of the main marker referenced to earth should be measured for example by using another IMU.

Also, the transformation between the forestry crane coordinate system and world frame has to be estimated. Approach in this work started by collecting several undistorted images where the tip of the boom is seen. To get more accurate calibration results, the boom is moved to different positions in a large area. Together with images, a pose of the camera and the locations of the boom tip in forestry crane coordinates are saved. In Matlab, the image coordinates of the boom tip are then acquired by mouse-clicking the target from each of the images. Using the camera pose and the image point, a 3D vector in world coordinates can be estimated. Then, we seek for a transformation matrix that will minimize the sum of squared distances of the forestry crane points and the 3D vectors. Matlab's numerical minimization function *fminsearch*, which uses the Nelder-Mead simplex algorithm as presented in [101], is chosen for the task.

5.3.7 Synchronization

Sensor fusion requires that the measurements from the camera and IMU are synchronized. In other words, at the time an image is captured, we should know the corresponding IMU measurement. The easiest way would be to have a common trigger for both of the devices. However, the camera did not provide an external triggering option. So, the synchronization had to be made on a software level.

IMU measurements are buffered into a container with time stamps when they are captured. A DCM estimate is also updated all the time and put in a buffer. When it is time to fuse the measurements, the DCM estimate is searched corresponding to the time the image was taken. However, the time stamps that are taken by the PC instantly when an image or IMU measurement is available still do not correspond in real time (Figure 16, next page). So, a delay between them has to be estimated.

A testing rig, where the device is moved horizontally along one of the IMU's axis, is set up. Because of that, the position of the IMU can be calculated from the accelerometer readings of that particular axis by double-integration. A ruler with measurement scale is put in front of the rig, so that the position of the camera can also be measured. In each of the camera's images, a vertical line is drawn across the center of the image. Then, the position is read from the intersection of the ruler and the line.

The small errors in the accelerometer calibration accumulate when they are double-integrated. So, the absolute distance that is calculated for IMU is slightly different than what it is for the camera. The problem is solved by normalizing the travelled distance from 0 to 1 for both sensors separately. The apparent position is thus same for the both devices as can be seen in Figure 16. But, IMU measurements seem to arrive faster to PC than camera images.

The solution is to fit third-order polynomials to the discrete measurements and find a time delay that minimizes the squared error differences between the positions. When the delay is taken into account, the position of the camera and IMU are very close to each other (Figure 17).

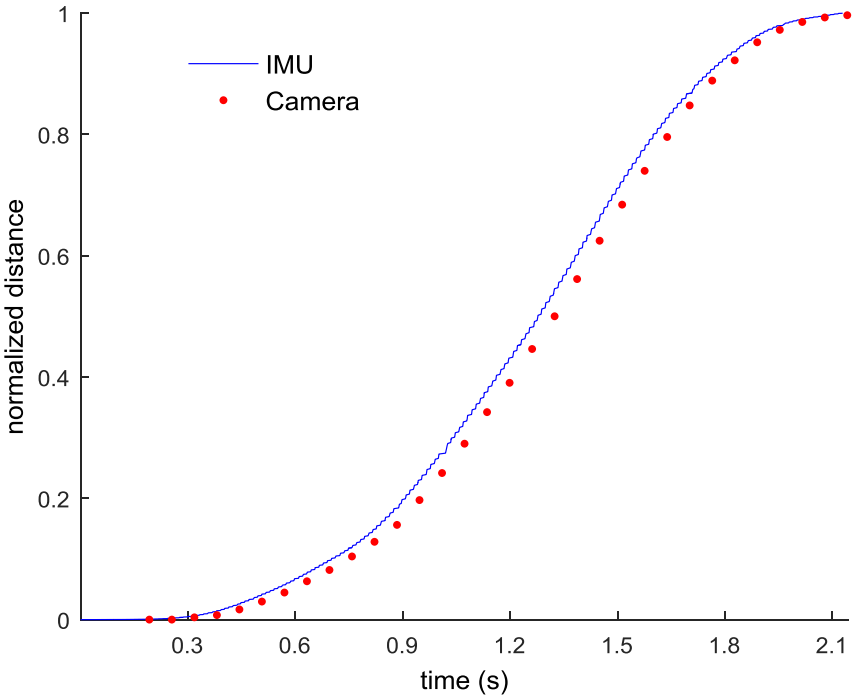

Figure 16: Camera and IMU positions without time delay correction.

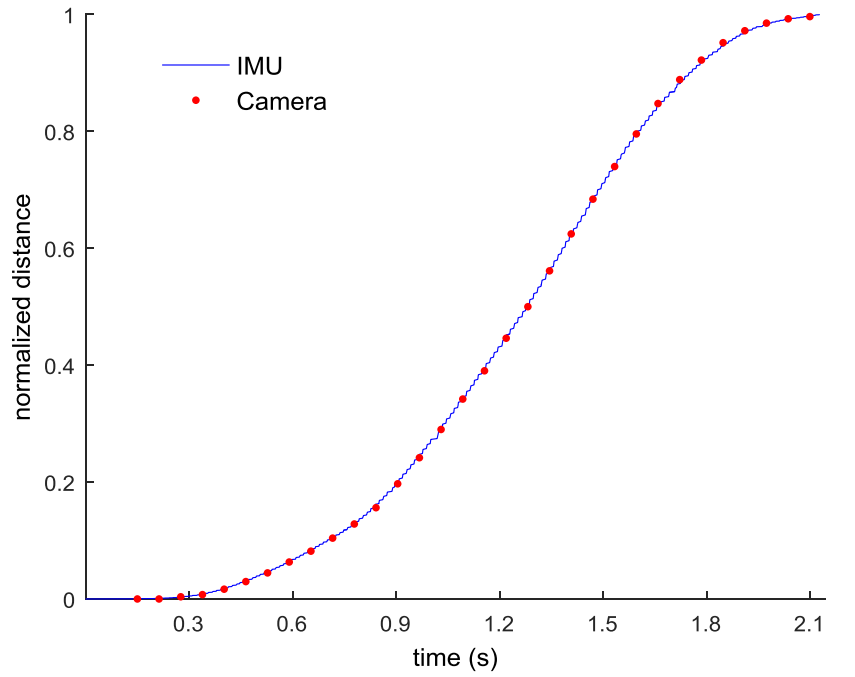

Figure 17: Camera and IMU positions with time delay correction.

5.3.8 Rendering

In previous sections, it is described how the pose of the system is calculated. The final essential part of the application is rendering of the images with virtual data. The question is what kind of virtual data should be shown to the forest machine operator and how.

For this prototype it was decided to show a 3D point cloud of the environment and a simple wireframe model of the crane and the tool. Since the tool is sometimes out of sight, it can be beneficial to have a so called “X-ray vision”. For the future development, one could also draw trajectories of the tool and the crane which the machine has planned. Furthermore, this could be used in training of a forest machine operator. Also, with the point cloud data in hand, it is easy to highlight certain parts of the environment, e.g. a tree to be cut.

The point cloud data is collected with the help of the 2D laser scanner attached in the crane so that it scans the environment vertically. The data points are updated as new information is available simultaneously rejecting dynamic objects like the markers on the crane. This is accomplished by a yet unpublished work by the thesis advisor.

Visualization of the point cloud is important because if the data points are just colored with one single color, it is difficult to distinguish objects from each other. Fortunately, the position of each point is available and this information can be used when coloring the pixels. One possibility is to change the RGB-values according to distance and height from the main marker, which is also the method chosen in this work.

Additionally, the positions of the joints of the crane and the tip of the tool are measured with the hardware described in Section 5.2.4. In this work, very simple model where the joints and the tool are connected with a straight line is used. The joint positions are also highlighted with different colored circles. One could also draw a CAD-model of the crane if more sophisticated looking rendering is required.

OpenCV has a function for projecting points to the image screen when pose and camera parameters are known. It works well in case all the projected points are in front of the camera. But, it also projects the points that are backside of the camera in the image. With known camera pose it is however possible to calculate if the point lies in the positive side of the Z-axis in the camera frame i.e. in front of the camera, (see (3) in Section 2.3).

This chapter described how the AR prototype was implemented. Following chapter will show how the system was evaluated. It will also present the results including an image from the augmented reality display.

6 Evaluation of the prototype

The testing was performed in an outdoor environment with the hardware described in Section 5.2. It is important to go in field testing in order to actually see if the prototype system works. Also, the errors affecting the pose estimation can be observed more realistically. Due to the lack of reference measurements, the evaluation of the pose estimate is done qualitatively from the augmentation success. In addition, the pose estimates from each of the markers, the camera's pose estimate and the sensor fused estimate are recorded and compared. Also, the real time capability of the system is inspected in this chapter.

6.1 Test setup

The camera and IMU were attached to the user's head in order to get natural movements inside the tractor. All the measurements were recorded from the whole 3 minutes 30 seconds lasting test which enabled the possibility to play-back the whole sequence over and over again. Testing was performed under a bright sunlight that had a negative effect for the quality of the captured images (Figure 18 and Figure 19).

Figure 18: The view inside the tractor cabin. IDs of the visible markers are marked in red.

In Appendix B, roll-, pitch- and yaw-angles are plotted from the whole test. One can view the extent of the head motions from those figures. For example the heading-angle (yaw) changes between −60 and 60 degrees.

Several markers were attached on to the structures of the tractor. They were able to put on to places that did not occlude the user's view of the outside environment as can be seen in Figure 18. A separate video was recorded for the use of an offline automatic calibration of marker positions.

The user moved the forestry crane with a dummy tool in to several different positions during the test sequence. The laser point cloud data was all the time updated as new measurements came through. Tractor itself was in place for the whole time but swayed slightly due to the motion of the crane.

Figure 19 shows one of the augmented video images. The forestry crane and the laser data points are rendered to the image with the method described in Section 5.3.8. Although there are seemingly some misalignments with the real and virtual objects, the overall look is still believable most of the time. In the next section, the errors in the augmentation result regarding to pose estimation are discussed.

Figure 19: An augmented image from the video. The crane and the tool is shown in blue along with the point cloud data from the lidar.

6.2 Pose analysis

The accuracy of the pose estimate is evaluated visually with the support of collected pose measurement data. The error in the pose estimate is easy to see because the virtual augmentation can be compared to the real world scene. By taking pose estimates from each individual marker, combination of markers and after sensor fusion, the causes behind the error can be better pointed out.

First of all, the noise in the measurements, or jitter, is perceived. The jitter makes the augmentation shaky and it can be very annoying for the user. One of the main ideas of sensor fusing the camera pose estimate with IMUs is to reduce jitter. It is evaluated by comparing the variance of derivatives of pitch-, roll- and yaw-angles, i.e. angular velocities, from the whole video sequence to each other. It is not an absolute measure because there are faster motions about certain axes. However, it is a useful measure for comparison.

In Table 4 on the next page, the variances of angular velocities are seen. The measurements support the assumption that IMU is useful in noise reduction. The greatest reduction of jitter happens in pitch-angle which is the lateral tilt of the head. In Figure 20, there are clearly identifiable moments in time, when the camera estimate becomes significantly noisy. Usually, these moments included a situation where only one marker was seen or some inaccurately detected marker came through from the quality test.

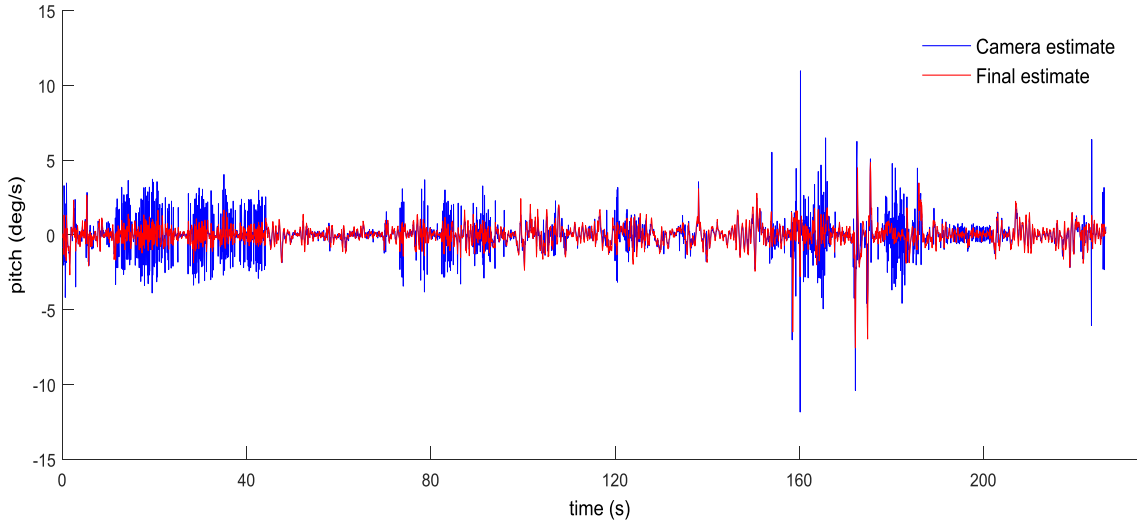

Figure 20: Derivatives of pitch-angle estimates. The camera estimate and the sensor fused (final) estimate are presented.

Table 4: Variance of angular velocities for roll, pitch and yaw.

	roll (°/s)	pitch (°/s)	yaw (°/s)
Camera	432.63	257.39	597.04
DCM IMU	375.78	125.84	536.19
Sensor fusion	358.04	118.66	543.04

However, as the position is estimated only from the camera pose estimate, the jitter is clearly visible at these moments. In Figure 21, position in z-axis (height) is inspected in the region were lot of jitter is recorded in the camera estimate. It is obvious that a single bad detection of one marker can hamper the estimation result.

Despite the fact that the marker acceptance test based on the quality measure sometimes fails, it still rejects most of the time markers that would otherwise produce noisy measurements. This is demonstratively seen in Figure 22, where the quality test was set off, compared to the Figure 23 with quality test on. The noisy main marker detection was almost completely discarded from the pose estimation process with the quality test on.

Another kind of pose estimation error that is clearly visible during the video sequence is shift. The virtual objects sometimes shift from their corresponding real world object for a longer time. From the analysis of the pose estimates, it can be deduced that this error is due to inaccurate calibration of marker positions.

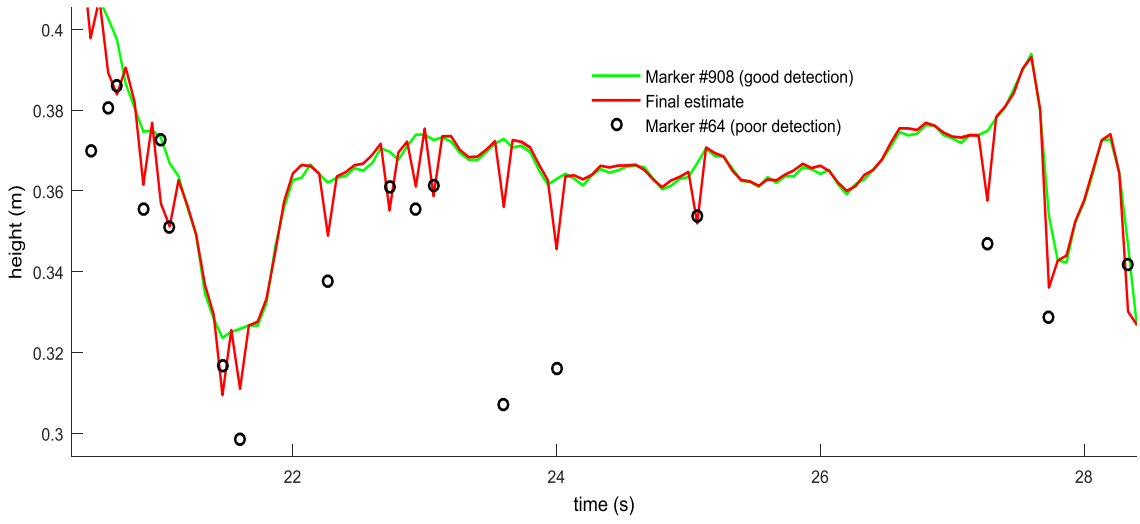

Figure 21: Jitter in height estimate caused by inaccurate detection of marker.

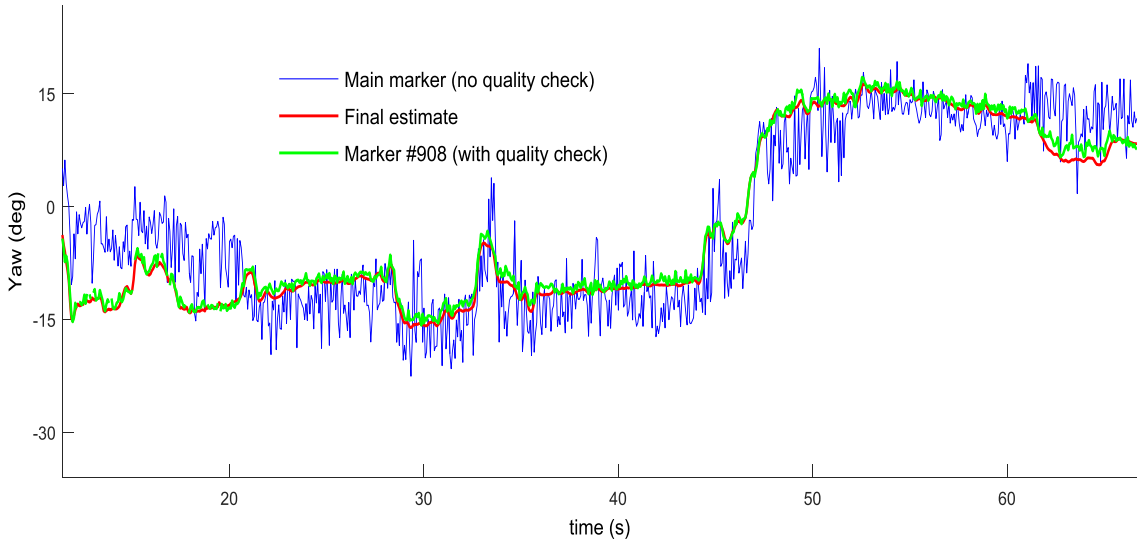

Figure 22: Without quality test of markers, the pose estimates can become noisy due to inaccurate detection. Here, a yaw-angle is presented.

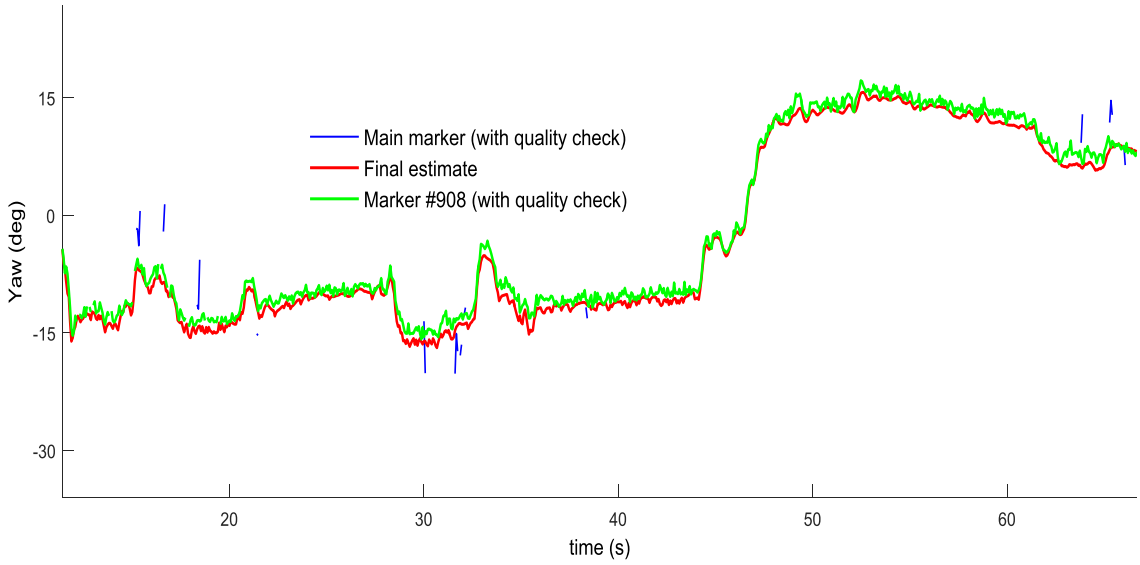

Figure 23: The quality test rejects almost all the measurements from the inaccurately detected main marker.

Because the main marker is known in the world coordinate system as it contains the origin, it gives accurate target estimates. In Figure 24, height of the head is measured from three different markers and the combination of all detected markers. It is clearly visible that the marker #64 gives significantly shifted estimates. Marker #908 is much closer to the estimate given by the main marker.

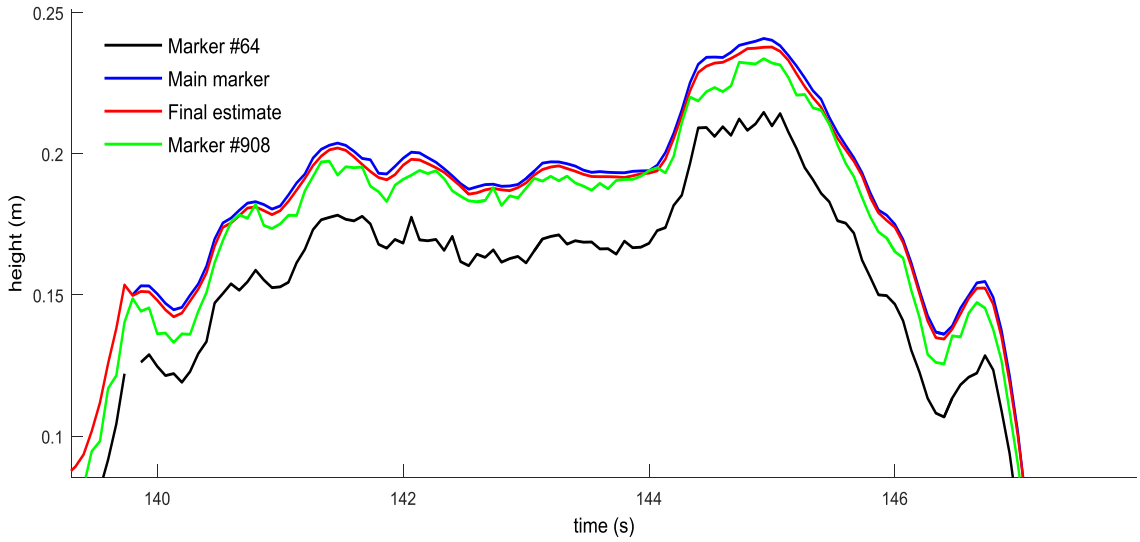

Figure 24: The error in automatic position calibration leads to shifted measurements. Marker #64 has significantly shifted height estimate.

Sometimes in the video sequence only one or two marker are detected. And, if one of them has large position error in the world coordinate system, the pose estimate tends to shift. Especially, it is evidential in the yaw (heading) estimate in which the camera estimate has a larger impact than in roll and pitch estimates.

From the pose estimation figures it is noticeable that the marker #908 is much more accurately detected inside the tractor than the marker #64. This difference has an effect on the quality of automatic calibration of marker position. Even though the automatic calibration procedure gives weights to the measurements based on the quality, it still does not completely correct the results.

Final observed error is shown as swaying of the virtual objects. The swaying effect of the virtual objects is gone when the orientation is acquired just from camera estimate. Hence, the error is deduced to be in the DCM estimate. The phenomenon can be seen in Figure 25 on the next page, where the DCM estimate in roll-angle has larger amplitude than what the camera estimate gives. The cause is yet unknown but it could be caused by the motion of the tractor cabin which is not compensated in this work. The sensor fusion algorithm now just assumes that the main marker is stationary in the Earth frame, which is not exactly true in this case. The movement of the forestry crane also moves the tractor cabin that has suspension system.

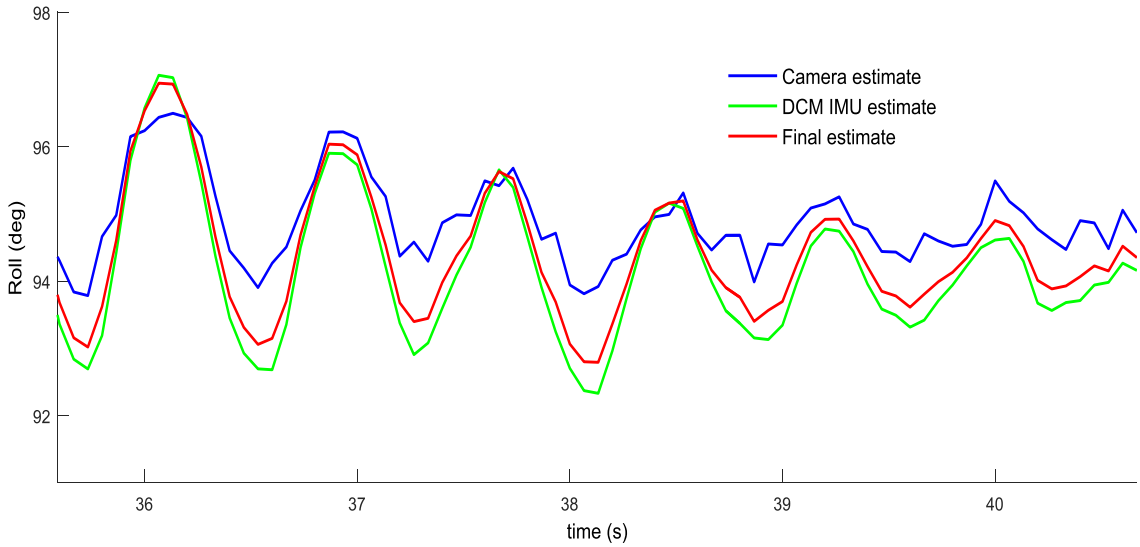

Figure 25: The amplitude of DCM IMU roll estimate is too high in this sequence. It shows as a swaying of virtual elements.

6.3 Real-time capability

The proposed system was executed on an Intel Core i5-760 processor with four cores operating at 2.8 GHz. The system is proved to work in real time as long as the size of the virtualized point cloud is restricted. The time consumption of different modules was measured during the whole video sequence. One augmentation of an image should last no more than ~67 ms which is the time between two adjacent camera frames in this system. In Table 5, the mean time consumption measurements of sensor fusion/synchronization, rendering, and pose estimation are presented.

Table 5: Time consumption for real-time capability.

Phase	**Time**
Marker detection	30 ms
Rendering	5 ms / 10 000 points
Sensor fusion / synchronization	6 ms
Camera estimate	< 1 ms
DCM estimate	< 1ms

It can be seen that the rendering time increased linearly to the size of the augmented points. It is due to the fact that the detected laser points are drawn one by one in a loop. In order to achieve real-time performance, it was decided to limit the augmented virtual points to under 20 000. This responds to a mean time of 10 ms.

Most of the computational time ~30 ms is spent in marker detection. It uses over half of the whole augmentation process. There are two main reasons for the long detection time. First of all, before the captured image is binarized, there is gain added to the image. It helps the detection module to better extract the borders of the markers but spends almost half of the detection time. Secondly, there are lots of contours found in the image after the adaptive threshold. It makes the detection of rectangle contours quite slow.

Sensor fusion and synchronization was measured together as they were tightly coupled. They spent 6 ms of time which is quite long compared to the pose estimation techniques. The pose estimation of the camera and DCM estimate is almost negligible for the real time point of view. Naturally, the DCM estimate has to be produced faster than the rate of the IMU measurements is set to be.

7 Conclusions

The designed and implemented prototype AR display has been proved to work in a forest machine in real time. Thus, the goal of this thesis has been reached. The main research problem was *how to measure the pose of an augmented reality display in a forest machine*? The problem is answered in Chapter 5 where the implementation of the system is described. The design is mostly based on previous research which was comprehensively surveyed.

For the pose estimation, the system uses a machine vision camera which detects 2D barcode markers inside the work machine. It uses ArUco library's [57] marker detection and pose estimation algorithms as it basis. The marker detection and camera's pose estimation methods are described in sections 3.3 and 5.3.3 respectively. Estimate of the orientation is sensor fused with the measurements from an IMU device. For the attitude estimation of the IMU, a DCM based method by Hyyti et al. [78] is utilized. A Kalman filter is designed to fuse the two orientation estimates together. The attitude estimation from IMU measurements and the sensor fusion module are presented in sections 4.3.2 and 5.3.5 respectively. The sensor fusion has been evaluated to reduce jitter in the augmented images that is shown in Section 6.2. This answers to the first research question: *What are suitable pose estimation methods in a forest machine*?

The second research question was, *which are the main error sources impacting on the pose estimation quality*? The pose estimate is evaluated in Section 6.2. It is deduced that jitter is mainly driven by inaccurate marker corner detection. Another visible error is a shift of the virtual points from their corresponding real objects. This is due to inaccurate calibration of marker corners in the world coordinate system. The automatic calibration process is also hampered mainly by the incorrect detection of marker corners. Also a "swaying" of the pose was perceived. It is most likely due to motion of the tractor cabin, which was not taken into account in this work. Minor errors to the pose estimation are also caused by inaccurate calibration parameters for camera and IMU.

The final research question was *how to visualize the virtual data for a forest machine operator*? It was decided to show a 3D point cloud data of the outside environment. It is produced by a 2D laser scanner attached on the forestry crane. Also, the crane itself is augmented with a simple wire frame model. The rendering of the video images is discussed in more depth in Section 5.3.8. The augmentation is executed in real time as long as the size of the virtualized point cloud is restricted. More information of the system's real time capability is provided in Section 6.3.

For future work, it would be reasonable to test the pose estimation accuracy of the head/camera against an accurate reference measurement. This would allow the possibility to compare the results achieved in this work with other state of the art methods. Also, the performance of the sensor fusion could be further reviewed. The full benefits of the sensor fusion are not utilized in this work. In the future work, a more overall IMU-camera sensor fusion algorithm should be introduced

which would fuse the whole pose and include online bias correction for both accelerometers and gyroscopes.

The ideas for development areas of the system come from the evaluation results of the prototype. An inaccurate corner detection of the markers seems to be the biggest problem affecting the pose estimation. For now, the quality of a detected marker is evaluated and the marker is rejected from the pose estimation if it is over a certain threshold. It works fine most of the time but it is clearly failing on some occasions (see Figure 21). Robust pose estimation methods that would deal with outliers should be utilized. RANSAC [68] could be a potential choice.

It was also noticed that the used camera did not survive very well in bright sunlight. The quality of the images captured with the selected camera was sometimes quite bad which also hampered the marker detection. A CMOS camera with a global shutter could be a potential choice for the future. Also, the markers could be illuminated to sharpen the corners in the images.

Furthermore, the automatic calibration of corner locations of the markers described in Section 5.3.2 needs improvement. Even though the method adds weight to the estimates based on the detection quality, it is still mostly affected by the inaccurate corner detection. Naturally, one could try to give even more weight to the better quality measurements, but it would not be fully satisfying manner. Instead, it could be interesting to try to fix the positions of the markers online based on all the other pose estimates similarly as in SLAM. On the hardware level the size of the markers could be increase when it is possible. Also, the positioning of the markers should be thought very carefully.

Development could also be done for a real time point of view. The real time capability is discussed in Section 6.3. The high marker detection time (~30 ms) is mainly caused due to preprocessing of the captured images. This could be possibly done in a separate thread which would reduce the overall time consumption massively. Furthermore, previous pose and IMU estimate could be utilized in order to search the markers inside a smaller image window. It was also noticed that it is difficult to choose the parameters of adaptive thresholding so that the marker borders would show clearly but there would not be too much detected contours due to noise. The high amount of contours slows the search for marker candidates.

The synchronization which is coupled with the sensor fusion phase takes too much time ~6 ms compared to the pose estimation computation. The wasted time could be reduced by using a common trigger for IMU and camera. Now the synchronization is done in the software side which is computationally more demanding.

Finally it must be stated that even though the system worked decently in the experiment, the robustness of the prototype has not been tested clearly enough. More testing should be performed in different lighting conditions and with different kind of movement. All in all, the methods used in this work have shown potential to be utilized as part of a fully working augmented reality user interface in a forest machine.

References

[1] Tulli (Customs), "Foreign Trade 2014 – Finnish Trade in Figures".

[2] FAO Forestry Department, "FAO yearbook 2014 - Forest Products".

[3] D. Ortiz Morales, S. Westerberg, P. X. La Hera, U. Mettin, L. Freidovich and A. S. Shiriaev, "Increasing the level of automation in the forestry logging process with crane trajectory planning and control," Journal of Field Robotics, vol. 31, pp. 343-363, 2014.

[4] O. Ringdahl, O. Lindroos, T. Hellström, D. Bergström, D. Athanassiadis and T. Nordfjell, "Path tracking in forest terrain by an autonomous forwarder," Scandinavian Journal of Forest Research., vol. 26, pp. 350-359, 2011.

[5] H. Hyyti, J. Kalmari and A. Visala, "Real-time detection of young spruce using color and texture features on an autonomous forest machine," in the 2013 International Joint Conference on Neural Networks (IJCNN), pp. 1-8, 2013.

[6] J. Kalmari, "Nonlinear Model Predictive Control of a Hydraulic Forestry Crane," Diss. Aalto University, 2015.

[7] O. Ringdahl, "Automation in forestry: development of unmanned forwarders," Diss. Umeå University, 2011.

[8] A. Tang, C. Owen, F. Biocca and W. Mou, "Comparative effectiveness of augmented reality in object assembly," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 73-80, 2003.

[9] S. J. Henderson and S. Feiner, "Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret," in 8th IEEE International Symposium on Mixed and Augmented Reality, pp. 135-144, 2009.

[10] F. Zhou, H. B. Duh and M. Billinghurst, "Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR," in Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, pp. 193-202, 2008.

[11] I. E. Sutherland, "A head-mounted three dimensional display," in *Proceedings of the Fall Joint Computer Conference, Part I,* pp. 757-764, 1968.

[12] T. P. Caudell and D. W. Mizell, "Augmented reality: An application of heads-up display technology to manual manufacturing processes," in *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences,* vol.2*,* pp. 659-669, 1992.

[13] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators and virtual environments,* vol. 6, pp. 355-385, 1997.

[14] P. Milgram, H. Takemura, A. Utsumi and F. Kishino, "Augmented reality: A class of displays on the reality-virtuality continuum," in *Photonics for Industrial Applications,* pp. 282-292, 1995.

[15] R. Azuma, "Tracking requirements for augmented reality," Communications of the ACM, vol. 36, pp. 50-51, 1993.

[16] M. D. Shuster, "Survey of attitude representations," Journal of the Astronautical Sciences, vol. 41, pp. 439-517, 1993.

[17] R. L. Holloway, “Registration Errors in Augmented Reality Systems,” Diss. University of North Carolina, 1995.

[18] G. Welch and E. Foxlin, "Motion tracking survey," IEEE Computer Graphics and Applications, pp. 24-38, 2002.

[19] J. P. Rolland, L. Davis and Y. Baillot, "A survey of tracking technology for virtual environments," Fundamentals of Wearable Computers and Augmented Reality, vol. 1, pp. 67-112, 2001.

[20] J. D. Hol, "Pose estimation and calibration algorithms for vision and inertial sensors," Diss. Linköping University, 2008.

[21] S. You, U. Neumann and R. Azuma, "Hybrid inertial and vision tracking for augmented reality registration," in Virtual Reality 1999, Proceedings, IEEE, pp. 260-267, 1999.

[22] T. Oskiper, H. Chiu, Z. Zhu, S. Samaresekera and R. Kumar, "Stable vision-aided navigation for large-area augmented reality," in Virtual Reality Conference (VR), 2011 IEEE, pp. 63-70, 2011.

[23] J. J. Craig, *Introduction to Robotics: Mechanics and Control.* Pearson Prentice Hall Upper Saddle River, 2005.

[24] E. Marchand, H. Uchiyama and F. Spindler, "Pose estimation for augmented reality: a hands-on survey," *IEEE Transactions on Visualization and Computer Graphics*, 2016.

[25] J. G. Henning and P. J. Radtke, "Detailed stem measurements of standing trees from ground-based scanning lidar," *Forest Science,* vol. 52, pp. 67-80, 2006.

[26] D. Van Krevelen and R. Poelman, "A survey of augmented reality technologies, applications and limitations," International Journal of Virtual Reality, vol. 9, pp. 1, 2010.

[27] P. D. Ritsos, “Architectures for Untethered Augmented Reality using Wearable Computers,” Diss. University of Essex, 2006.

[28] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier and B. MacIntyre, "Recent advances in augmented reality," Computer Graphics and Applications, IEEE, vol. 21, pp. 34-47, 2001.

[29] W. Broll, I. Lindt, J. Ohlenburg, I. Herbst, M. Wittkamper and T. Novotny, "An infrastructure for realizing custom-tailored augmented reality user interfaces," *IEEE Transactions on Visualization and Computer Graphics,* vol. 11, pp. 722-733, 2005.

[30] T. Kawashima, K. Imamoto, H. Kato, K. Tachibana and M. Billinghurst, "Magic paddle: A tangible augmented reality interface for object manipulation," in *Proceedings of ISMR 2001,* pp. 194-195, 2001.

[31] Apple, "Siri," Available: http://www.apple.com/ios/siri/. [Accessed Sept. 26, 2016].

[32] K. Dorfmüller-Ulhaas and D. Schmalstieg, "Finger tracking for interaction in augmented environments," in *IEEE and ACM International Symposium on Augmented Reality 2001, Proceedings,* pp. 55-64, 2001.

[33] R. Y. Wang and J. Popović, "Real-time hand-tracking with a color glove," *ACM Transactions on Graphics (TOG),* vol. 28, pp. 63, 2009.

[34] A. Esteves, E. Velloso, A. Bulling and H. Gellersen, "Orbits: Gaze interaction for smart watches using smooth pursuit eye movements," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology,* pp. 457-466, 2015.

[35] Z. Szalavári and M. Gervautz, "The personal interaction Panel–a Two-Handed interface for augmented reality," in *Computer Graphics Forum,* 1997.

[36] A. Henrysson, M. Billinghurst and M. Ollila, "Face to face collaborative AR on mobile phones," in *Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality, Proceedings,* pp. 80-89, 2005.

[37] S. Siltanen, "Theory and applications of marker-based augmented reality," VTT Science 3, 2012.

[38] A. De la Escalera and J. M. Armingol, "Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration," *Sensors,* vol. 10, pp. 2027-2044, 2010.

[39] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal of Robotics and Automation,* vol. 3, pp. 323-344, 1987.

[40] J. Weng, P. Cohen and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *IEEE Transactions on Pattern Analysis & Machine Intelligence,* pp. 965-980, 1992.

[41] Y. Abdel-Aziz, "Direct linear transformation from comparator coordinates in close-range photogrammetry," in *ASP Symposium on Close-Range Photogrammetry in Illinois,* 1971.

[42] J. G. Francis, "The QR transformation a unitary analogue to the LR transformation—Part 1," *The Computer Journal,* vol. 4, pp. 265-271, 1961.

[43] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 28, pp. 1335-1340, 2006.

[44] F. Ababsa and M. Mallem, "Robust camera pose estimation using 2d fiducials tracking for real-time augmented reality systems," in *Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry,* pp. 431-435, 2004.

[45] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge university press, 2003.

[46] C. Harris and M. Stephens, "A combined corner and edge detector." in *Alvey Vision Conference,* pp. 50, 1988.

[47] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using Mixed Integer Linear Programming," *Pattern Recognition,* vol. 51, pp. 481-491, 2016.

[48] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal,* vol. 29, pp. 147-160, 1950.

[49] DAQRI, "ARToolKit", Apr. 1, 2016, Available: http://artoolkit.org/. [Accessed Sept. 26, 2016]

[50] Applications of Artificial Vision, "ArUco: a minimal library for Augmented Reality applications based on OpenCV," Available: http://www.uco.es/investiga/grupos/ava/node/26. [Accessed Sept. 26, 2016].

[51] VTT Technical Research Centre of Finland, "ALVAR", Available: http://virtual.vtt.fi/virtual/proj2/multimedia/alvar/. [Accessed Sept. 26, 2016].

[52] C. Lu, G. D. Hager and E. Mjolsness, "Fast and globally convergent pose estimation from video images," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, pp. 610-622, 2000.

[53] J. J. Moré, "The levenberg-marquardt algorithm: Implementation and theory," in *Numerical Analysis*, Springer Berlin Heidelberg, pp. 105-116, 1978.

[54] V. Lepetit, F. Moreno-Noguer and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International Journal of Computer Vision,* vol. 81, pp. 155-166, 2009.

[55] S. Meers, K. Ward and I. Piper, "Simple, robust and accurate head-pose tracking using a single camera," in *Mechatronics and Machine Vision in Practice,* pp. 111-122, 2008.

[56] M. Faessler, E. Mueggler, K. Schwabe and D. Scaramuzza, "A monocular pose estimation system based on infrared leds," in *2014 IEEE International Conference on Robotics and Automation (ICRA),* pp. 907-913, 2014.

[57] 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," *Pattern Recognition,* vol. 47, pp. 2280-2292, 2014.

[58] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., 2008.

[59] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization,* vol. 10, pp. 112-122, 1973.

[60] W. Förstner and E. Gülch, "A fast operator for detection and precise location of distinct points, corners and centres of circular features," in *Proc. ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data,* pp. 281-305, 1987.

[61] V. Lepetit and P. Fua, *Monocular Model-Based 3D Tracking of Rigid Objects.* Now Publishers Inc, 2005.

[62] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision,* vol. 60, pp. 91-110, 2004.

[63] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding,* vol. 110, pp. 346-359, 2008.

[64] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision,* pp. 430-443, 2006.

[65] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha and P. Fua, "BRIEF: Computing a local binary descriptor very fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, pp. 1281-1298, 2012.

[66] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision,* pp. 2564-2571, 2011.

[67] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond and D. Schmalstieg, "Real-time detection and tracking for augmented reality on mobile phones," *IEEE Transactions on Visualization and Computer Graphics,* vol. 16, pp. 355-368, 2010.

[68] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM,* vol. 24, pp. 381-395, 1981.

[69] J. Fuentes-Pacheco, J. Ruiz-Ascencio and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review,* vol. 43, pp. 55-81, 2015.

[70] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *6th IEEE and ACM International Symposium on Mixed and Augmented Reality,* pp. 225-234, 2007.

[71] R. Mur-Artal, J. Montiel and J. D. Tardós, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics,* vol. 31, pp. 1147-1163, 2015.

[72] N. Yazdi, F. Ayazi and K. Najafi, "Micromachined inertial sensors," *Proceedings of the IEEE,* vol. 86, pp. 1640-1659, 1998.

[73] E. L. Renk, M. Rizzo, W. Collins, F. Lee and D. S. Bernstein, "Calibrating a triaxial accelerometer-magnetometer-using robotic actuation for sensor reorientation during data collection," *IEEE Control Systems,* vol. 25, pp. 86-95, 2005.

[74] A. Kim and M. Golnaraghi, "Initial calibration of an inertial measurement unit using an optical position tracking system," in *Position Location and Navigation Symposium, 2004. PLANS 2004,* pp. 96-101, 2004.

[75] J. Lötters, J. Schipper, P. Veltink, W. Olthuis and P. Bergveld, "Procedure for in-use calibration of triaxial accelerometers in medical applications," *Sensors and Actuators A: Physical,* vol. 68, pp. 221-228, 1998.

[76] S. P. Won and F. Golnaraghi, "A triaxial accelerometer calibration method using a mathematical model," *IEEE Transactions on Instrumentation and Measurement,* vol. 59, pp. 2144-2153, 2010.

[77] L. Sahawneh and M. Jarrah, "Development and calibration of low cost MEMS IMU for UAV applications," in *5th International Symposium on Mechatronics and its Applications,* pp. 1-9, 2008.

[78] H. Hyyti and A. Visala, "A DCM Based Attitude Estimation Algorithm for Low-Cost MEMS IMUs," *International Journal of Navigation and Observation,* vol. 2015, 2015.

[79] W. Fong, S. Ong and A. Nee, "Methods for in-field user calibration of an inertial measurement unit without external equipment," *Measurement Science and Technology,* vol. 19, 2008.

[80] P. Aggarwal, Z. Syed, X. Niu and N. El-Sheimy, "A standard testing and calibration procedure for low cost MEMS inertial sensors and units," *Journal of Navigation,* vol. 61, pp. 323-336, 2008.

[81] J. Stuelpnagel, "On the parametrization of the three-dimensional rotation group," *Society for Industrial and Applies Mathematics Review,* vol. 6, pp. 422-430, 1964.

[82] F. L. Markley, "Attitude error representations for Kalman filtering," *Journal of Guidance, Control, and Dynamics,* vol. 26, pp. 311-317, 2003.

[83] J. Kalmari, T. Pihlajamäki, H. Hyyti, M. Luomaranta and A. Visala, "ISO 11783 compliant forest crane as a platform for automatic control," *IFAC Proceedings Volumes,* vol. 46, pp. 164-169, 2013.

[84] J. Kalmari, H. Hyyti and A. Visala, "Sway estimation using inertial measurement units for cranes with a rotating tool," *IFAC Proceedings Volumes,* vol. 46, pp. 274-279, 2013.

[85] The Imaging Source. “DFK 41AU02 Color Camera,” fact sheet.

[86] B. E. Bayer, “Color Imaging Array”, U. S. Patent 3,971,065, 1976.

[87] Tamron, “13FM22IR”, datasheet.

[88] InvenSense. “MPU-6050,” PS-MPU-6500A-01 datasheet, Sept. 2013, [revised Mar. 2014].

[89] J. Rowberg, "I2C Device Library," Available: http://www.i2cdevlib.com/. [Accessed Sept. 26, 2016].

[90] PJRC, “Teensy microcontroller,” Available: http://www.pjrc.com/teensy/. [Accessed Oct. 2, 2016].

[91] Arduino, “Software & libraries,” 2016, Available: http://www.arduino.cc/. [Accessed Oct. 2, 2016].

[92] CadSoft Computer GmbH, “EAGLE PCB design,” Available: http://cadsoft.io/. [Accessed Sept. 26, 2016].

[93] SICK, “LMS221,” 8012678/SG26 Technical information, Sept. 2008.

[94] J. Bouguet, “Camera Calibration Toolbox for Matlab,” Nov. 14, 2015, Available: http://www.vision.caltech.edu/bouguetj/calib_doc/. [Accessed Sept. 26, 2016].

[95] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, pp. 1330-1334, 2000.

[96] S. Siltanen, M. Hakkarainen and P. Honkamaa, "Automatic marker field calibration," *in Proceedings of the Virtual Reality International Conference (VRIC),* pp. 261-267, 2007.

[97] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik,* vol. 1, pp. 269-271, 1959.

[98] F. L. Markley, Y. Cheng, J. L. Crassidis and Y. Oshman, "Averaging quaternions," *Journal of Guidance, Control, and Dynamics,* vol. 30, pp. 1193-1197, 2007.

[99] D. Oberkampf, D. F. DeMenthon and L. S. Davis, "Iterative pose estimation using coplanar points," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Proceedings CVPR'93,* pp. 626-627, 1993.

[100] W. Premerlani and P. Bizard, "Direction cosine matrix imu: Theory," *Diy Drone: Usa,* pp. 13-15, 2009.

[101] J. C. Lagarias, J. A. Reeds, M. H. Wright and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *Society for Industrial and Applied Mathematics Journal on Optimization,* vol. 9, pp. 112-147, 1998.

A Direct linear transformation

DLT is a closed-form solution to the Perspective-n-Point problem [41], [45]. In case of camera calibration, it estimates the parameters of a 3x4 projection matrix $\boldsymbol{M}$. The projection matrix maps homogeneous 3D world points into homogeneous image points

$$\boldsymbol{x_I} = \boldsymbol{M}\boldsymbol{x_W}. \tag{A1}$$

At least six correspondences are needed in order to obtain all the parameters. The projection equation can be written for each point correspondences as

$$\begin{aligned} x_{Ii} &= \frac{u_i}{w_i} = \frac{m_{11}x_{Wi} + m_{12}y_{Wi} + m_{13}z_{Wi} + m_{14}}{m_{31}x_{Wi} + m_{32}y_{Wi} + m_{33}z_{Wi} + m_{34}} \\ y_{Ii} &= \frac{v_i}{w_i} = \frac{m_{21}x_{Wi} + m_{22}y_{Wi} + m_{23}z_{Wi} + m_{24}}{m_{31}x_{Wi} + m_{32}y_{Wi} + m_{33}z_{Wi} + m_{34}}, \end{aligned} \tag{A2}$$

and then it can be modified to:

$$\begin{aligned} &x_{Wi}m_{11} + y_{Wi}m_{12} + z_{Wi}m_{13} + m_{14} - x_{Ii}x_{Wi}m_{31} - x_{Ii}y_{Wi}m_{32} - x_{Ii}z_{Wi}m_{33} - \\ &x_{Ii}m_{34} = 0 \\ &x_{Wi}m_{21} + y_{Wi}m_{22} + z_{Wi}m_{23} + m_{24} - y_{Ii}x_{Wi}m_{31} - y_{Ii}y_{Wi}m_{32} - y_{Ii}z_{Wi}m_{33} \\ &-y_{Ii}m_{34} = 0 \end{aligned} \tag{A3}$$

Now it is easy to transfer this in to matrix format

$$\boldsymbol{A}\boldsymbol{m} = \boldsymbol{0}, \tag{A4}$$

where

$$\boldsymbol{A} = \begin{bmatrix} x_{\mathrm{W1}} & y_{\mathrm{W1}} & z_{\mathrm{W1}} & 1 & 0 & 0 & 0 & 0 & -x_{\mathrm{I1}}x_{\mathrm{W1}} & -x_{\mathrm{I1}}y_{\mathrm{W1}} & -x_{\mathrm{I1}}z_{\mathrm{W1}} & -x_{\mathrm{I1}} \\ 0 & 0 & 0 & 0 & x_{\mathrm{W1}} & y_{\mathrm{W1}} & z_{\mathrm{W1}} & 1 & -y_{\mathrm{I1}}x_{\mathrm{W1}} & -y_{\mathrm{I1}}y_{\mathrm{W1}} & -y_{\mathrm{I1}}z_{\mathrm{W1}} & -y_{\mathrm{I1}} \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ x_{Wn} & y_{Wn} & z_{Wn} & 1 & 0 & 0 & 0 & 0 & -x_{In}x_{WN} & -x_{In}y_{Wn} & -x_{In}z_{Wn} & -x_{In} \\ 0 & 0 & 0 & 0 & x_{Wn} & y_{Wn} & z_{Wn} & 1 & -y_{In}x_{Wn} & -y_{In}y_{Wn} & -y_{In}z_{Wn} & -y_{In} \end{bmatrix} \tag{A5}$$

$$\boldsymbol{m} = [m_{11}\, m_{12} \ldots\ m_{34}]^T. \tag{A6}$$

Parameter vector $\boldsymbol{m}$ lies in the null space of $\boldsymbol{A}$ and it can be determined by a singular value decomposition of $\boldsymbol{A}$. Solution is the column of the right unitary matrix corresponding to the smallest singular value of $\boldsymbol{A}$.

B Motions of the camera/IMU device

During the test sequence, the camera/IMU device was attached rigidly to the user's head. It allowed obtaining natural motions for the pose estimation process. In figures below, roll-, pitch and yaw-angles are measured referenced to the main marker. The largest motions happen about yaw-angle (-60° ... 60°) which is not surprising as the user follows the environment from side to side. In contrary, motions about pitch-angle are very modest since it is unnatural when sitting in a chair.

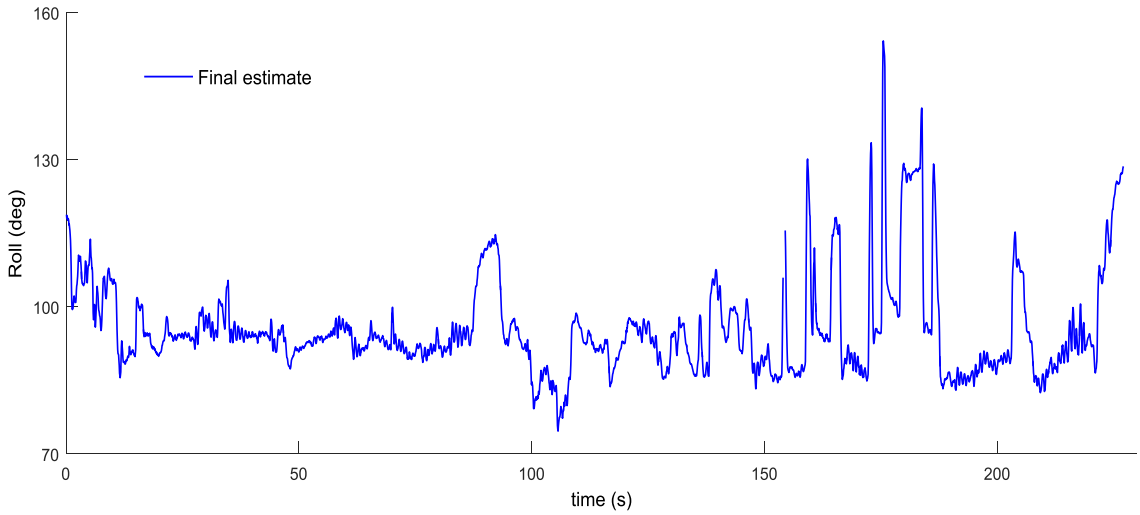

Figure B1: Roll-estimate (nodding of the head) during the experiment.

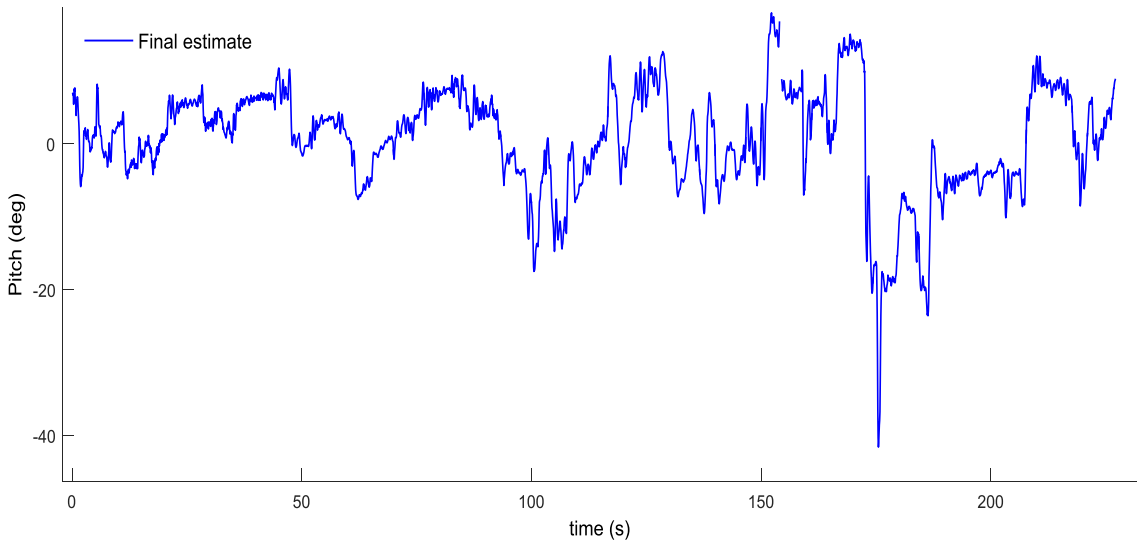

Figure B2: Pitch-estimate (lateral tilt of the head) during the experiment.

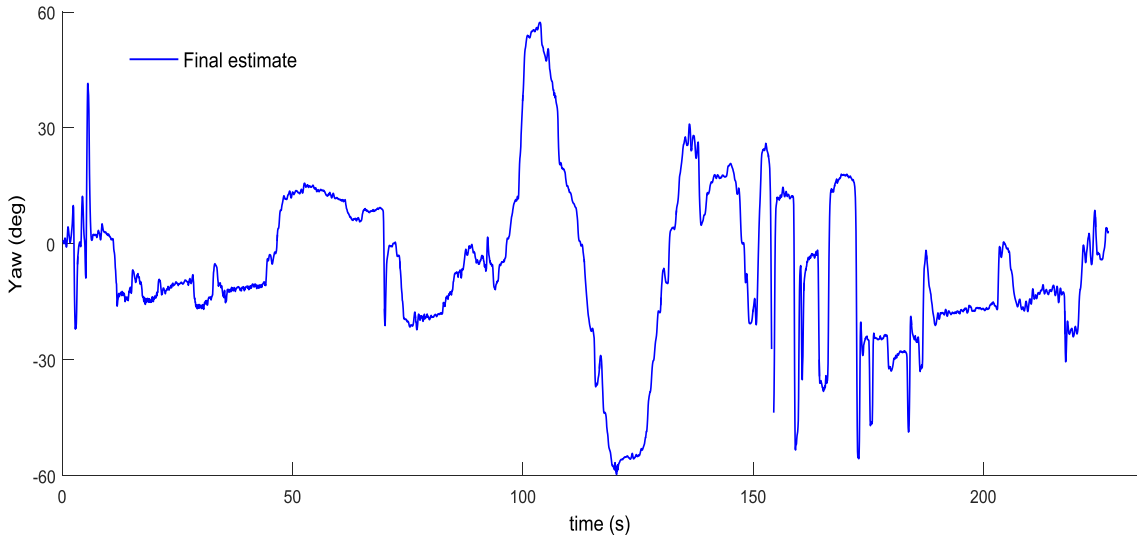

Figure B3: Yaw-estimate (heading) during the experiment.