Sensor Fusion in Head Pose Tracking for Augmented Reality

Stelian-Florin Persa

Sensor Fusion in Head Pose Tracking for Augmented Reality

PROEFSCHRIFT

Ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus, Prof.dr.ir. J.T. Fokkema,
voorzitter van het college voor promoties,
in het openbaar te verdedigen op dinsdag 6 Juni om 10.00 uur

door Stelian-Florin PERSA

Inginer Universitatea Tehnica din Cluj-Napoca
geboren te Cluj-Napoca (Roemenie)

Dit proefschrift is goedgekeurd door de promotoren:

Prof.dr.ir. I.T. Young

Prof.dr.ir. R.L. Lagendijk

Toegevoegd promotor:

Dr.ir. P.P. Jonker

Samenstelling promotiecommissie:

Rector Magnificus	Technische Universiteit Delft, voorzitter
Prof.dr.ir. I.T. Young	Technische Universiteit Delft, promotor
Prof.dr.ir. R.L. Lagendijk	Technische Universiteit Delft, promotor
Dr.ir. P.P. Jonker	Technische Universiteit Delft, toegevoegd promotor
Prof.dr.ir. H.J. Sips	Technische Universiteit Delft
Prof.dr.ir. J.J.M. Braat	Technische Universiteit Delft
Prof.dr.ir. F.C.A. Groen	Vrije Universiteit Amsterdam
Prof.dr.ir. A. Vlaicu	Technical University Cluj-Napoca, Romania

The work presented in this thesis was supported by Ubiquitous Communications (UBICOM) programme, funded by Delft University of Technology DIOC research programme.

ISBN-10: 90-9020777-5

ISBN-13: 978-90-9020777-3

Printed by Wöhrmann Print Service

Devoted to my wife Monica, my daughter Irina and

to my parents

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, there has been an explosion of interest in virtual reality systems. Virtual reality (VR) is relevant to many applications involving data visualization, communication, and immersive entertainment. In virtual reality concepts, a user is represented by a virtual self or virtual representative 'living' or existing in a virtual world. Time and place boundaries are no longer present. A simple example of virtual presence is 'always-on' surfing on the internet combined with a user homepage. Physical mobility of 'on-line' end-users introduces a new dimension to virtual reality and virtual presence.

A similar area, with perhaps even more commercial applications than virtual reality, is augmented reality (AR). Whereas in VR systems the user physically remains in place and moves only virtually, in AR systems the user moves in the physical world that is augmented with virtual objects and scenes. When physical mobility is added, it implies that the users experience in addition a dynamic physical environment, meaning that:

- the virtual environment can influence the behavior and decisions taken in the physical world
- the changing physical environment and the user's behavior therein can influence the virtual environment of the user and the services desired and/or offered in this environment.

Augmented reality systems differ from virtual reality systems in that the user is not completely immersed in the virtual environment. In augmented reality systems, a heads-up display is used to superimpose computer-generated graphics on the user's view of the real world. The superimposed images supplement the information available to the user in the natural scene. For example, an augmented reality system could be used to help a maintenance technician find the appropriate adjustment points in a complicated piece of machinery, or to help a surgeon by superimposing CT or MRI data on a patient's body, essentially giving a surgeon X-ray vision.

Despite its potential, the development of functional AR systems faces several technical challenges. In most AR applications, it is crucial that the synthetic images are registered precisely with the real world. The degree of accuracy required depends on the task, but in many cases the requirements are quite stringent. Furthermore, many tasks require large motions of the user's head with high accelerations, which place certain demands on the sensors that track the head motion. Finally, in nearly all cases, display updates must occur with a latency of only a fraction of a second. These technical challenges have hampered the development of viable, inexpensive AR systems for precise applications.

Inertial measurement components, which sense either translational acceleration or angular rate, are being embedded into common user interface devices more frequently. Examples include the VFX1 virtual reality headtracking systems[1], the Gyro Mouse (a wireless 3D pointer)[2], and Microsoft's SideWinder tilt-sensing joystick[3]. Such devices hold a number of advantages over

other sensing technologies such as vision systems and magnetic trackers: they are small and robust, and can be made wireless using a lightweight radio-frequency link.

However, in most cases, these inertial systems are put together in a very ad hoc fashion, where a small number of sensors are placed on known fixed axes, and the data analysis relies heavily on a priori information or fixed constraints. This requires a large amount of custom hardware and software engineering to be done for each application, with little possibility for reuse.

The pose of the head is defined as the position and orientation of the head in a 3D world. There are two aspects to the problem of head-tracking: relative sensing of the head pose and absolute sensing of the head pose. Relative sensing can be performed by inertia tracking systems based on accelerometers and gyroscopes and use the human head as their frame of reference. Although these systems can be made fast, they usually build up errors rather quickly. Consequently they must be calibrated using the world/earth as a frame of reference. For this, the pose must be sensed with the earth as a frame of reference. Systems that are able to do this are magnetometers, that sense the earth's magnetic field, inclinometers, that sense the earth's gravitation, computer vision based systems, that are able to measure features in the camera's field of view, and GPS systems that are based on line of sight to satellites with fixed orbits around the earth.

This thesis proposes to solve the problem of head tracking for augmented reality systems based on optical see-through head-mounted displays, by developing a compact, lightweight, low power, six degrees-of-freedom inertial measurement unit (IMU) based on gyroscopes and accelerometers, for relative positioning, combined with an absolute positioning framework based on the sensing of the earth's magnetic and gravitation fields, GPS and computer vision. The system should be light, small and should easily be incorporated into almost any interface or device so that AR systems and applications can be simply and quickly developed. The system software should run on standard computer hardware with standard operating system, and in the near future inexpensive back or waist mounted versions should be easily developed.

1.1 Previous Work

Current augmented reality systems differ from each other primarily in three ways: the display technology used to overlay synthesized graphics on the user's field of view, the sensing technology used to track the user's head, and the calibration method used to determine system parameters.

Many research projects in augmented reality have used optical see-through head-mounted displays [5], [7], [9]. These displays work by optically combining light from the environment with the overlay images. The combination is done using lenses, half-silvered mirrors, or other optical components. The principal advantage of this type of display is that the user's view of the real world is substantially unobstructed. Consequently, the user has a high resolution, high contrast view of the workspace. One disadvantage of optical see-through head-mounted displays is that the optics used to combine the images typically have a narrow field of view, and also somewhat decrease the light intensity reaching the user's eyes. Another disadvantage is that the software in the augmented reality system has no access to the combined image (natural scene plus overlay), so correcting registration errors and establishing system calibration are difficult.

A second category of display system for augmented reality is a video-based display, which is typically used in medical augmented reality applications [10]. In this type of display, the user views the workspace through one or two video cameras, which may be head mounted. The real and synthesized images are merged as video signals, and presented to the user through a video display which occludes the user's natural view of the environment. While this type of display provides flexibility in video composition strategies, and gives the underlying software access to the com-

bined images, the workspace view lacks the fidelity of natural vision, and the user's view of the workspace is from the perspective of the system video camera(s), which generally does not match that of the user's eye(s).

Four types of sensors have traditionally been used for head tracking in augmented reality applications. Mechanical sensors measure the position of the user's head using an attached linkage. This type of sensor is typically very accurate, and can meet the bandwidth requirements of augmented reality, but is often somewhat cumbersome, and restricts the user's range of motion. Magnetic position sensors (Polhemus, etc.) have seen wide use in virtual reality applications, and limited use in augmented reality [9], [7]. These sensors are inexpensive and readily available, but data rates are typically slow, and their accuracy suffers in applications where a large working volume is required, or where there are nearby ferromagnetic objects such as steel wall studs. Acoustic position sensors are inexpensive, fast, and accurate, but latency increases with distance between the acoustic transmitter and receiver [11]. Optical sensors using video cameras have the potential to be inexpensive, fast, accurate, and offer large working volume. Unfortunately, systems to date require either large arrays of markers, such as LEDs, to be installed at precise locations in the workspace, or use custom camera hardware [5], or have a limited working volume [9]. One disadvantage of optical position sensors is that there must be an unobstructed line of sight between the sensor and a target [11].

Inertial measurement devices have a very eventful history. The field began with motion-stabilized gunsights for ships and was later driven by guidance systems for aircraft and missiles (dating back to the V2 rocket), providing a large body of work to draw on. Because of the relatively large cost, size, power and processing requirements of these systems, they were previously not appropriate for human-computer interfaces and consumer applications. However, recent advances in micro-electromechanical systems (MEMS) and other microfabrication techniques have led to lower costs, more compact devices, while at the same time, the processing power of personal computers has been increasing exponentially. Therefore, it is now possible for inertial systems, which previously required large computers and large budgets, to reach end-users. The Intersense[16] inertial-acoustic tracking system is an example of a commercial product exploiting this new market.

There is currently a number of six degree-of-freedom systems commercially available, and several of them are targeted at either the high-end user interface market or the motion capture market. The Ascension Technology miniBird 500[17] magnetic tracker is the smallest available at 10mm × 5mm × 5mm making it particularly easy to use. However, the closed-loop nature of the sensor requires that it be wired, and the base unit is fairly cumbersome. The Intersense IS-600 inertial-acoustic system[16] offers excellent accuracy over a very large range, but requires a fair amount of infrastructure for the sonar grid (used in position tracking). Crossbow Technologies offers the DMU-6X inertial measurement unit[18] which has excellent accuracy, but is quite large (> 600 cm^3). Also, all these systems are fairly expensive and none matches our specification in terms of ease of use (small, wireless, low-cost, low power)[Chapter 1.3].

Inertial tracking systems such as Intersense's are known as strapdown systems, because the sensors are fixed to the local frame of the instrumented object. Many of the early military applications were closed-loop systems, where the inertial sensors are mounted on a controlled gimbaled platform which attempts to remain aligned with the world frame, regardless of the motion of the body. Such systems can operate over a much smaller dynamic range and therefore provide higher accuracy, but they also tend to be fairly large and costly. Therefore, for low-cost human interface applications, open-loop strapdown systems are more appropriate.

Recent uses of inertial sensors in major products have tended toward the automotive sector. The first major application of MEMS accelerometers was as a cheap, reliable trigger mechanism for airbag deployment and they have since been applied to active suspension control, as well as other

applications. Gyroscopes are most often used to provide turn rate information to four-wheel steering systems to help the front and rear tires have matching speed. They have very recently been used to provide heading information for in-vehicle tracking systems (which obtain position from the speedometer or a Global Positioning System unit).

1.2 Problem Formulation and Requirements Outline

Augmented Reality (AR) differs from Virtual Reality (VR) in the sense that virtual objects are rendered on a see-through headset. As with audio headphones, which make it possible to hear sound in private, partly in overlay with the sounds from the environment, see-through headsets can do that for visual information. The virtual objects are in overlay with the real visual world. It can also be used to place visual information on otherwise empty places, such as white parts on the walls of a museum. The 3D vector of position and orientation is referred to as pose. Knowing the pose of those walls and the pose of a person's head, visual data can be perfectly inlayed on specific spots and kept there while the head is moving. To lock the virtual objects in the scene, the head movements must be sampled with such a frequency and spatial accuracy that the rendering of virtual images does not cause motion sickness. Augmented Reality systems can be applied in Tour Guiding, Remote Maintenance, Design Visualization and Games.

Mobile augmented reality [4] is a relatively new and intriguing concept. The ability of augmented reality to present information superimposed on our view on the world opens up many interesting opportunities for graphical interaction with our direct environment. Combining this with mobility further increases the potential usage of this technology for direct daily use.

However, the technical problems with mobile augmented reality are just as great. As with other head-mounted display systems, augmented-reality displays also require an extremely high update rate. Simple head movements may, in short time, give rise to significant changes in viewing position and viewing direction. The virtual information associated with objects in the scene and displayed within the viewing window will then have to be updated to maintain the proper alignment with the objects in the real world. The viewpoint changes will therefore have to be tracked and fed back to the display system, in order to re-render the virtual information in time at the correct position.

No research has been reported yet on the effects of jitter on virtual environment users, although it seems obvious that jitter which is visible will reduce the illusion of presence, and may even contribute to simulator sickness if it is too extreme. The threshold of being detectable is not known, although it would be a very easy experiment to conduct. From experience, it seems that jitter of 0.05° r.m.s. in orientation and 1 mm r.m.s. in position is generally unnoticeable in an HMD with magnification of 1, but becomes fairly visible in a virtual binoculars simulator or virtual set camera tracker with 7X zoom. Note that when viewing distant virtual objects, tracker position jitter becomes irrelevant, and orientation jitter multiplied by zoom factor is all that matters. For viewing close objects, translational jitter becomes dominant. It is an interesting question whether we are sensitive to perceived jitter in world space or screen space (pixels), or some combination. If the former scenario is the case, than we might be psychologically more forgiving of an object jittering 2 pixels at 1 meter apparent depth (4 mm in world space) than an object jittering 1 pixel at 10 meters apparent depth (2 cm in world space). This again is an easy experiment. Other factors which affect the perception of jitter are display resolution and whether or not the graphics are anti-aliased. Summarizing, we may say that the alignment criteria both for accurate positioning and for time lag are extremely high.

Not all AR systems require every virtual object to be precisely registered on a real object. Some applications consists of displaying virtual objects that appear to be floating in mid-air within the user's view of the real world. This is useful for AR gaming, in which virtual beasts might jump in through the windows and attack the player, or for shared AR visualization, in which a 3D model or dataset might hover above a table while multiple participants view it from different angles. In this type of applications precise registration to the nearest mm or even cm level may not be required. Thus a slight spatial distortion such as a systematic offset or nonlinearity may be less noticeable, but sensitivity to latency is probably nearly the same. The threshold for noticing latency in see-through display modes is thought to be lower than for video see-through immersive displays because there are real objects having zero latency visible for comparison. On the other hand, the unconscious effects of latency such as decreased presence or simulator sickness are probably worse in video see-through because the whole world loses its perceived stability.

1.3 System Concept

The Ubicom System [8] is an infrastructure for mobile multi-media communication. The system consists of a backbone compute server, several base stations, and a possible large number of mobile units. The base stations maintain a wireless (radio or infrared) link to the mobile units. The radio transmission will account for approximately 10 Mbit/s of data bandwidth per user, enough to transmit compressed video with high quality. The cell size (distance between the base stations) is in the order of 100 metesr: typically the distance between lamp posts to which the base stations may be attached.

The mobile unit consists of a receiver unit and a head-set. The head-set contains a light-weight head-mounted display that offers the user a mix of real and virtual information. This may be realised by superimposing the virtual information on the real world or by replacing parts of the real world with virtual information. In the latter case, we need partial visual blocking of the view on the outside world. In addition to the display facilities, the head-set will also have a light-weight video camera that is used for position tracking and to record video data. In order to keep the power consumption low, the head-set and receiver unit will only have limited processing and memory capabilities.

The headtracking system for Augmented Reality that is proposed in this thesis is a system based on a cascade of three sensor systems:

- A system that detects the user's pose in the world. This is based on the use of a differential global positioning system (DGPS). The user's position is assumed to be in the range of meters, his orientation in steps of 45 degrees, his velocity in the order of a few km/h, with an update frequency in the order of a few minutes. This sensor system is a virtual sensor, it involves a variety of calculations.
- A system that detects the coarse user's head pose. This is based on the use of a Vision system. The user's head position is assumed to be in the range of a few centimeters, his head orientation in steps of a few degrees, his head velocity in the order of a a few m/s, with an update frequency in the order of a second. This sensor system is a virtual sensor, it involves a variety of image processing operations and possibly communication with some backbone system to match data with a GIS or CAD system.

- A system that detects the fine user's head pose. This is based on the use of an Inertia Tracking system. The user's head position is assumed to be in the range of a few millimeters, his orientation in steps of a few arc seconds, his velocity in the order of a a few mm/10msec, with an update frequency in the order of 10 msec. This sensor system is a virtual sensor, based on the fusion of many small sensors. This fusion is done using Kalman filtering.

Position tracking is done in three steps. A first position estimation is done using GPS or similar position detecting techniques. One option is to calculate the position relative to the base stations. A second level of position tracking is using object and scene recognition. Given a 3D description of the environment (e.g. a CAD-model) and an initial position estimate, an accurate position may be calculated iteratively. However, the model data will only be available at the backbone and most of the calculations to derive the viewing position will have to be performed at the backbone as well. Part of this computation could be offloaded to the active base station. The latency introduced by first sending the video-captured scene information from the mobile unit to the backbone, then the processing at the backbone or base station and the transmission of the obtained viewing parameters, will be too large to update of the visual display. Therefore to be able to anticipate on small position changes immediately, the direction and acceleration of the movement will be sensed with an inertial tracker and directly fed back to the display system. In the same way, the orientation tracking will be based on object recognition and direct feedback from the inertial tracker.

1.4 Objectives

This thesis addresses the issue of providing a low cost, high integrity, aided inertial navigation system for mobile augmented reality applications.

Inertial Navigation is the implementation of inertial sensors to determine the pose (position and orientation) of a mobile user. Inertial sensors are classified as dead reckoning sensors since the current evaluation of the state of the mobile user is formed by the relative increment from the previous known state. As such, inertial navigation has unbounded error growth since the error accumulates at each step. Thus in order to contain these errors, some form of external aiding is required. In this thesis, the aided information will derive from Global Navigation Satellite Systems (GNSS) such as the Global Positioning System (GPS) for outdoor applications, and from vision for indoor applications.

In summary, the goal of this thesis is to provide an aided inertial navigation system which can be used cost-effectively by the civilian sector for augmented reality applications and autonomous navigation.

The objectives of this thesis in order to reach this goal are:

- To understand the implications of implementing low cost inertial units for navigation.
- High grade inertial sensors can be an expensive approach to navigation. However, by implementing low cost inertial sensors one correspondingly introduces greater errors to the navigation solution. The sources of these errors need to be understood in order to minimize their impact on the performance of the system.
- To understand the effect of GNSS accuracy on moving land vehicles. High accuracy satellite navigation receivers are generally produced for surveying purposes and not for dynamic movement. Furthermore, associated errors such as multipath and satellite signal blockage, common with terrain-based navigation, need to be comprehended.

- To develop navigation algorithms which assist in limiting the errors of the inertial navigation system while also detecting multipath errors and providing data during satellite blockages, and hence increasing the integrity of the navigation loop.
- To develop this algorithm in real time so as to provide navigation data to an autonomous control system. Furthermore, address the issue of data latency commonly associated with satellite-based navigation systems and its effect on real time applications.
- To investigate the addition of mobile user modeling to the navigation system in order to increase the performance and integrity of the navigation data.
- Furthermore, to address the issue of multiple sensor aiding to a single inertial unit for further improvement in performance.
- To investigate and develop a redundant inertial unit in order to provide the foundations for future work and to address the issues behind increase in navigation performance and autonomous fault detection techniques. Redundancy in satellite numbers and its effect on navigation and fault detection are well documented for satellite-based positioning systems. This theory is in turn reflected in the development of this redundant inertial unit

1.5 Contributions

The scale of the human motion-tracking problem is vastly different from that of global navigation. Tracking is only required over a small area, but requires precision in the order of a centimeter or less, while with navigation a kilometer is often sufficient. The size and cost of the sensors must also be scaled down tremendously for human body-mounted ”consumer” use. Thus inertial human motion tracking would need to achieve far higher accuracy using tiny sensors than navigation systems are able to achieve using instruments far larger and more costly.

The main contributions presented in this thesis are as follows:

- We present an overview of position measurement technology, with both advantage and disadvantage.
- We present sensors that are often used in pose determination with their advantages and disadvantages. Based on the requirements formulated for Augmented Reality Applications, we select some and combine them in an Inertial Measurement Unit.
- Since existing technology or sensor alone cannot solve the pose problem, we combine information from multiple sensors to obtain a more accurate and stable system. This integration is achieved using a Kalman filter. We present the formulation for a new Kalman filter implementation based on quaternions.
- We present the development of an entire pose determination system using off-the-shelve existing sensors integrated using separate Kalman filters. Where the research and implementation were not complete due to the time constraint, we provide simulations to prove the validity of the concept. Still, a unified solution is presented: inertial measurement integration for orientation and GPS in combination with a differential correction unit for positioning. The accuracy obtained is 0.5 degrees for orientation, at an update rate of 100 Hz, and 5 m accuracy for positioning at 1 Hz.
- We present all the necessary steps for implementing a vision positioning system. Integration with the other sensor systems is left to future research.

1.6 Thesis Outline

The thesis is organized as follows. In Chapters 2-3 we tackle the problem of pose determination. In Chapter 2 we present the most used pose determination technologies together with requirements for Augmented Reality tracking.

Chapter 3 surveys the existing sensors for pose determination, presenting their operating principle and their characteristics. It also makes a selection from among them based on the requirements formulated for Augmented Reality pose, and combines them in an Inertial Measurement Unit.

In Chapter 4 we proceed with the design of an inertial navigation system based on sensor data fusion using a novel approach: Kalman filtering using quaternions. It contains an overview of the estimation theory necessary to understand the quaternion Kalman filter. This chapter also presents results of the field tests conducted to study the benefits of integration under various environments.

As the Vision subsystem is far more complex, due to the perception and segmentation issues of complex objects in a 3D world, the Vision subsystem is treated in a subsequent chapter: Chapter 5. This complex “real-time sensor” can be plugged into the sensor data fusion system described in Chapter 5.

In Chapter 6 we summarize the presented work with concluding remarks. Here, we also present ideas and possibilities for future research.

Chapter 2

Survey of Positioning Technologies

2.1 Introduction

In this chapter we will review a variety of existing techniques and systems for position determination [78]. Nowadays, due to the complexity of mobile systems and in particular those of the autonomous nature, navigation is encapsulated by the science and technology of being able to determine the position, velocity and orientation of a system in real time with a greater demand on accuracy.

A navigation system provides the required information by either sensing the relative movement of the mobile system, or by determining where the system is with respect to external features, or both. This is accomplished through the implementation of either dead reckoning or absolute sensors. Dead reckoning sensors measure the relative movement of the vehicle with respect to a previously known state. Examples include inertial units, wheel encoders and air data systems. Absolute sensors observe the external environment and relate the vehicle's state to those observations. Examples include vision, radar and the Global Positioning System (GPS). Dead reckoning sensors usually output their data at high frequencies, however, due to their relative accumulation of data, errors also accumulate with time. The errors associated with absolute sensors on the other hand are fixed. However, the update rates are generally low.

To enjoy the benefits of both, navigation systems generally include both types of sensors and either select which is the most appropriate/correct piece of information, or employ a system which fuses the data from both in some optimal fashion. A common methodology for fusion is through the implementation of a statistical filter.

2.2 Types of AR Systems

In order to combine the real world with virtual objects in real-time we must configure camera and display hardware. The three most popular display configurations currently in use for augmented reality are Monitor-based, Video See-through and Optical See-through.

2.2.1 Monitor-based Display

The simplest approach is a monitor-based display, as depicted in Figure 2-1. The video camera continuously captures individual frames of the real world and feeds each one into the augmentation system. Virtual objects are then merged into the frame, and this final merged image is what users ultimately see on a standard desktop monitor. The advantage of this display technology is its simplicity and affordability, since a consumer-level PC and USB or FireWire video camera is all that is required. Additionally, by processing each frame individually, the augmentation system can use vision-based approaches to extract pose (position and orientation) information about the user for registration purposes (by tracking features or patterns, for example). However, this simplicity comes at the expense of immersion. Clearly, viewing the real world through a small desktop mon-

itor limits the realism and mobility of the augmented world. Additionally, since each frame from the camera must be processed by the augmentation system, there is a potential delay from the time the image is captured to when the user actually sees the final augmented image. Finally, the quality of the image is limited by the resolution of the camera and display.

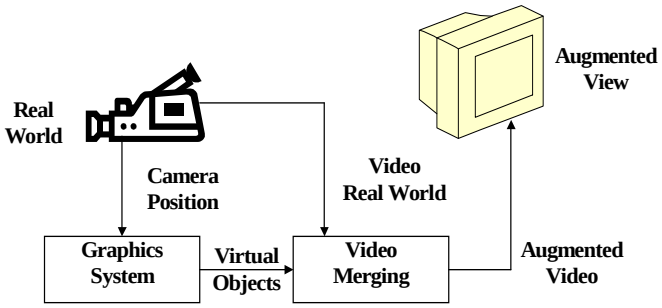

Figure 2-1. Monitor-based display

2.2.2 Video See-through Display

In order to increase the sense of immersion in virtual reality systems, head-mounted displays (HMD) that fully encompass the user's view are commonly employed. There are two popular methods to bring HMDs into the augmented reality environment. Figure 2-2 shows a schematic for a video see-through augmented reality system. In this configuration, the user does not see the real world directly, but instead only sees what the computer system displays on the tiny monitors inside the HMD. The difference between this and a virtual reality HMD is the addition of video cameras to capture images of the real world. While this configuration is almost identical to the monitor-based technology in terms of functionality, the use of a stereo camera pair (two cameras) allows the HMD to provide a different image to each eye, thereby increasing the realism and immersion that the augmented world can provide. Like the monitor-based setup, the video see-through display is prone to visual lags due to the capture, processing, augmentation, and rendering of each video frame. Additionally, a large offset between the cameras and the user's eyes can further reduce the sense of immersion, since everything in the captured scenes will be shifted higher or lower than where they should actually be (with respect to the user's actual eye level).

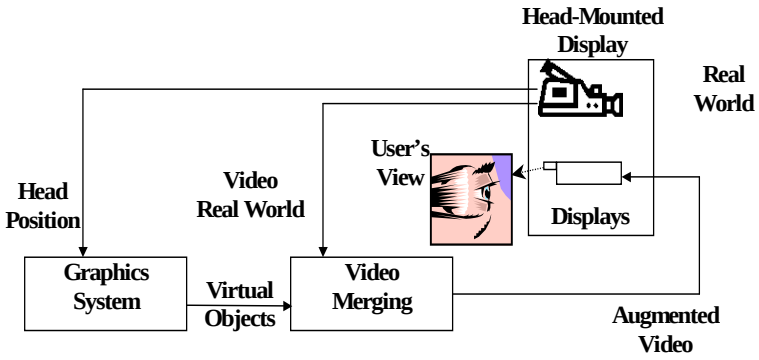

Figure 2-2. Video see-through display

2.2.3 Optical See-through Display

The other popular HMD configuration for augmented reality is the optical see-through display system, as depicted in Figure 2-3. In this setup, the user is able to view the real world through a semi-transparent display, while virtual objects are merged into the scene optically in front of the user's eyes based on the user's current position. Thus when users move their heads, the virtual objects maintain their positions in the world as if they were actually part of the real environment. Unlike video see-through displays, these HMDs do not exhibit the limited resolutions and delays when depicting the real world. However, the quality of the virtual objects will still be limited by the processing speed and graphical capabilities of the augmentation system. Therefore, creating convincing augmentations becomes somewhat difficult since the real world will appear naturally while virtual objects will appear pixilated. The other major disadvantage with optical see-through displays is their lack of single frame captures of the real world, since no camera is present in the default hardware setup. Thus position sensors within the HMD are the only facility through which pose information can be extracted for registration purposes. Some researchers [61] have proposed hybrid solutions that combine position sensors with video cameras in order to improve the pose estimation.

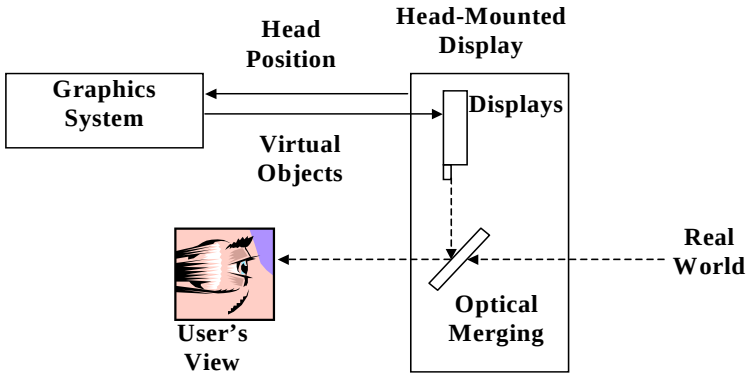

Figure 2-3. Optical see-through display

2.3 Relative Position Measurements

Perhaps the most important result from surveying the vast body of literature on mobile user positioning is that, to date, there is no truly elegant solution for the problem. The many partial solutions can roughly be categorized into two groups: relative and absolute position measurements. Because of the lack of a single, generally good method, developers of automated guided vehicles (AGVs) and mobile navigation usually combine two methods, one from each category. The two categories can be further divided into subgroups.

Acquiring relative measurements is also referred to as dead reckoning, which has been used for a long time, ever since people started traveling around. Originally, this is the process of estimating the position of an airplane or a ship, only based on the speed and direction of travel and the time that has passed since the last known position. Since the position estimates are based on earlier positions, the error in the estimates increases over time.

2.3.1 Odometry

Odometry works by integrating incremental information over time. By using wheel encoders to count the number of revolutions of each wheel, the robot measures the distance it has traveled and

its heading direction. Odometry is widely used, because it gives good short-term accuracy, is inexpensive, and allows for very high sampling rates.

However, due to drift and slippage the integration of the wheel revolutions leads to errors in both traveled distance and orientation. These errors accumulate over time unless an independent reference position is used periodically to reduce the error. In particular, errors in the orientation cause large positioning errors.

2.3.2 Inertial Navigation

This method uses gyroscopes and sometimes accelerometers to measure rate of rotation and acceleration. Measurements are integrated once (or twice) to yield position. Inertial navigation systems also have the advantage that they are self-contained. On the downside, inertial sensor data drifts with time because of the need to integrate rate data to yield position; any small constant error increases without limit after integration. Inertial sensors are thus unsuitable for accurate positioning over an extended period of time. Another problem with inertial navigation is the high equipment cost. For example, highly accurate gyros, used in airplanes, are prohibitively expensive. Very recently fiber-optic gyros (also called laser gyros), that are very accurate[18], have fallen dramatically in price and have become a very attractive solution for mobile navigation.

In inertial navigation, acceleration sensors[18] are used for making distance measurements. Inertial measurements are frequently required in the tracking of planes, boats, and automobiles over long distances and long time constants. Inertial navigation is an extremely demanding application for sensors and many factors contribute to the performance of an inertial navigation system. Alignment, scale factor errors, and offset errors are crucial, because a constant error in these readings will result in a quadratically growing position error as given in the following Equation 2-1.

$$X_{Error} = \frac{1}{2} Acc_{Error} \bullet T^2 \qquad \text{eq. 2-1.}$$

A simple 1 dimensional system is shown in the next Figure 2-4. This configuration would be used for measuring the distance traveled by a projectile fired down a tube, or the quarter-mile time of an automobile on a straight track. The acceleration is integrated into a velocity signal and a position signal.

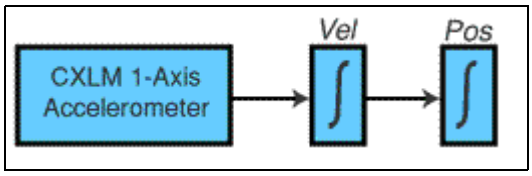

Figure 2-4. 1 Dimensional Position Measurement

The double integration leads to an unacceptable rate of positional drift and must be corrected frequently by some external source. The techniques that we can use to correct the error are:

- for indoor operation, we can use acoustic (ultrasound) range measurement
- for outdoor operation, we can use GPS system to update the position
- for both indoor and outdoor operation, we can use image processing to extract features like: corners of the room or of buildings, straight lines of buildings or roads, object matching or stereo vision.

A more complex inertial measurement is that of a 6 degree-of-freedom system as found in an airplane or spacecraft. These systems are free to move in any direction. Figure 2-5 shows the block

diagram of such a system. The GPS system provides periodic updates in order to prevent error build-up within the navigation solution. This feedback loop typically makes use of a control algorithm such as a Kalman filter. Also notice that the acceleration readings have to be transformed (rotated) to the Earth frame. This rotation is necessary because the accelerations, as measured by the sensors, are referenced to the local (body) coordinate frame. The distances that the system reports are measured with respect to the Earth.

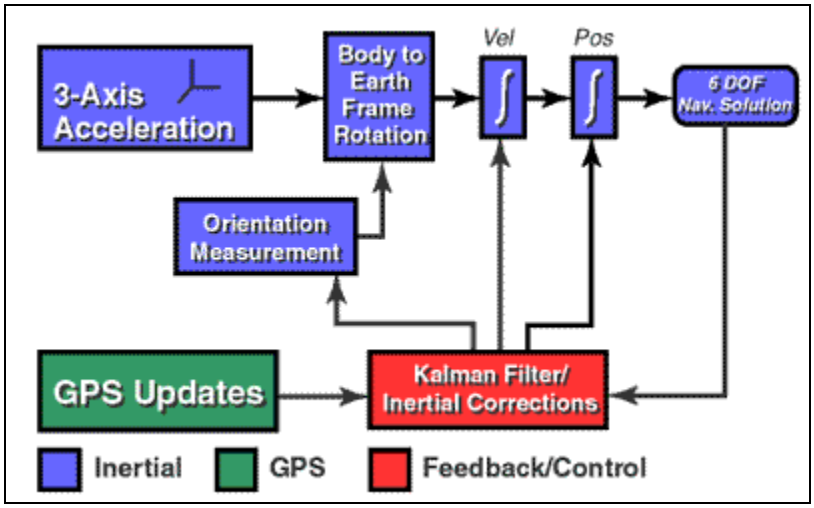

Figure 2-5. 6 DOF Inertial Measurement

Inertial sensors [13] are used in applications where rotational and linear movements are to be measured without reference to external coordinates. Gyroscopes and accelerometers can measure these movements. A major user of such sensors and systems is aviation, with its widespread use of artificial horizon and other navigational systems.

2.3.2.1 Accelerometers

An accelerometer is a precision instrument, which couples a mass to an instrument case through an elastic, viscous, or electromagnetic restraint. Typically, the mass is only allowed a single degree of freedom, which may be either linear or rotary. Accelerometers are typically divided into two classes, depending upon their intended use.

- Guidance accelerometers are those intended for use in measuring the steady state accelerations of rigid bodies. One might use a guidance accelerometer for measuring the acceleration of an automobile.
- Vibratory or seismic accelerometers are those intended to measure sinusoidal accelerations. They are used to measure vibrations in applications as varied as structural testing, and earthquake and tsunami detection.

All accelerometers operate on the same principle, namely, measuring the relative displacement of a small mass, called a proof or seismic mass, constrained within an accelerating case. Generally, the constraining device is a transducer that returns a signal proportional to the displacement of the proof mass.

An accelerometer measures platform acceleration, which can be integrated to produce the velocity, and double integrated to produce the distance that was traveled. Other uses for accelerometer data include the detection of impacts for air bag deployment, and inclination measurement through sensing of the Earth’s gravitation. Vibration measurements are used to detect potential engine damage

or failure. Sensor drift makes regular zero velocity updates (ZUPTs) necessary for periods without an accurate external reference, since the double integral can accumulate substantial errors. The drift is mostly temperature related, so we investigated the various compensation schemes. The use of constant temperature ovens can ensure good stability after warm-up, but an oven uses too much power and space for a mobile user. A possible valuable technique for a mobile user is to generate a temperature profile and count on temperature repeatability. The temperature profile can be done a priori or built up over time in a real time application.

Accelerometers are generally based on observing the displacement of a suspended mass caused by inertia. Two common implementations are a damped spring and a pendulum. Methods such as differential capacitance, inductance, or optical methods can be used to measure the displacement. Sometimes a magnetic field or servo is employed to keep the mass in a fixed position. A damped spring will allow a suspended mass to displace under acceleration. The movement of the mass would be sensed through capacitance, an optical method, or otherwise. Damping is usually accomplished by the use of a viscous fluid medium. The displacement can be described by:

$$F = ma = m\left(\frac{d^2x}{dt^2}\right) + c\left(\frac{dx}{dt}\right) + Kx \qquad \text{eq. 2-2.}$$

where *F* is the applied force, *m* is the mass of the suspended mass, *c* is the damping coefficient (a function of the medium), *K* is the spring stiffness, *x* is the displacement of the spring relative to resting position.

Many modern designs use MEMS technology, e.g. those from Analog Devices (2000). Here, a small proof mass is suspended from two sides with flexible coils. When the platform is accelerated, the displacement of the mass is measured by a differential capacitance as shown in Figure 2-6

.

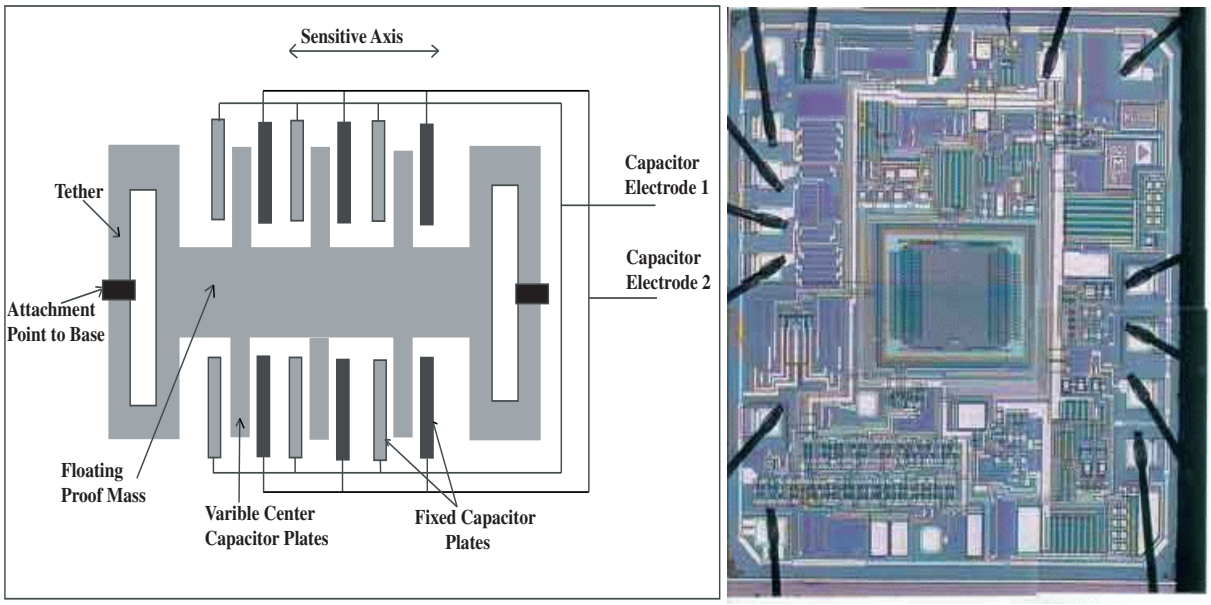

Figure 2-6. A diagram and die photo of the ADXL105 MEMS sensor

2.3.2.2 Gyroscopes

Gyroscopes (or gyros) measure rotational values without reference to external coordinates [94]. Most gyros measure the speed of rotation (also known as 'rates') in one axis and are known as 'single axis gyros'. Speed of rotation is normally measured in units of degree per second or hour (°/sec.

or °/h). The operating principle of these sensors can be split into two groups: Mechanical and Optical.

The mechanical gyroscope, a well-known and reliable rotation sensor, is based on the inertial properties of a rapidly spinning rotor. One of today's high precision gyros is still the mechanical, rotating gyro; however, it is dependent on linear acceleration because of its mechanical measuring principle.

The optical group comprises the fiber optic and laser types. These use the Sagnac Effect (named after its French discoverer), which, when the sensor is turned, results in a difference in transit time between two light waves passing through the same optical path but in opposite directions. Optical gyros therefore do not depend on acceleration as gyros in the mechanical group. This is one of their biggest advantages, and for this reason we prefer to use this type of gyroscope in standard mid-range systems (for high dynamic performance applications).

A single gyro measures rotation on a single plane, but a triad of gyros is mounted, preferably orthogonally, in a single enclosure to monitor the three possible rotations in 3-D space. Many types of gyros are available, ranging in price and stability. Gyros are classified into gimbaled or strapdown varieties, where gimbaled gyros maintain a fixed orientation in an inertial frame. Low cost, or potentially low cost, gyro alternatives -all of the strapdown variety- will be presented below with some examples. Table 2-1 summarizes some low cost gyro technologies with associated cost and accuracy.

Table 2-1. Comparison of low cost Gyro Technologies

Gyro Type	Principle of Operation	Cost($)	Stability(^{o}/h)
Rotating	Conservation of Angular Momentum	10-1000	1-100+
Fiber Optic	Sagnac Effect	50-1000	5-100+
Vibrating Piezoelectric	Coriolis Effect	10-200	50-100+

Strapdown gyros measure rotation on a fixed plane with respect to the vehicle, which is generally not on a plane orthogonal to the gravitation vector. Therefore, they do not sense the entire rotation in heading, but they also sense rotations in pitch and roll.

Piezoelectric materials exhibit the piezoelectric (PE) effect; a vibrational motion of crystals creates an electric potential within the material. The reverse piezoelectric effect is also used, whereby application of an electric field to a PE crystal will cause it to vibrate. This process is used in the operation of most wristwatches. Quartz is used in many PE applications, however it tends to have temperature instability and physical limitations that have given rise to recent advances in PE ceramics. Piezoelectic applications usually make use of resonant or harmonic frequencies, which are a function of the size, shape, and dielectric properties of the piece of material. Vibrating gyroscopes are usually designated as micro-electro-mechanical-system (MEMS) sensors, i.e. sensors that couple electrical and mechanical interactions with microchip fabrication methods.

Two basic types have been constructed, the free-free-bar and the tuning fork. Both use an excitation signal to drive the oscillation of a piezoelectric crystal, then sense rotation through a piezoelectrically generated output signal. The tuning fork variety is simply a tuning fork that is made of piezoelectric material. The name free-free-bar comes from the use of a slender bar, with PE ceramics attached, which is fixed at the centre and has both ends free to vibrate.

Piezoelectric gyros are essentially Coriolis sensors. The Coriolis force is a fictitious force exerted on a body when it moves in a rotating reference frame. It is a fictitious force as, like the centrifugal force, it is a by-product of measuring coordinates with respect to a rotating coordinate system as opposed to the acceleration of a mass in an inertial frame. It is given by the cross product

$$\vec{F}_c = 2m(\vec{v} \times \vec{\omega}) \qquad \text{eq. 2-3.}$$

where *m* is the mass of the object, *v* is the velocity vector of the object and ω is the angular rotation rate vector.

The excited vibration of the bar or fork creates an oscillating velocity vector. If this system is rotated around the sensitive axis, an oscillating force will be induced, which causes vibration of the piezoelectric crystal. This vibration can be sensed as a varying voltage, which is then processed into an output signal. The operation of the Murata Gyrostar free-free-bar implementation, which was the gyro used for our project, is described below. An example of the tuning fork type can be found in the Systron Donner GyroChip series [84].

2.4 Absolute Position Measurements

Absolute position measurements supply information about the location of the robot, irrespective of previous location estimates; the location is not derived from integrating a sequence of measurements, but directly from one measurement. This has the advantage that the error in the position does not grow without limit, as is the case with relative position techniques. Absolute measurements can either supply the full location, or just a part of it, like for example the orientation.

2.4.1 Active Landmarks

Active landmarks, also called beacons, are landmarks that actively send out location information. Active landmarks can take on the form of satellites or other radio transmitting objects. A mobile system senses the signals sent out by the landmark to determine its position. Two closely related methods are commonly used to determine the absolute position of the robot using active landmarks: triangulation and trilateration. Triangulation techniques use distances and angles to three or more active landmarks; trilateration techniques only use distances. The angles and/or distances are then used to calculate the position and orientation of the mobile user.

2.4.1.1 Wireless Location Systems

Though wireless users are mobile by nature, knowledge of their dynamic location is very useful information. In emergency situations, it is essential to know a wireless user's location to be able to manage the emergency effectively. On the other hand, if the technology is available to accurately determine the location, location-specific content (e.g. closest airport, closest restaurant, closest hotels, etc.) can be delivered to the user as an add-on wireless service. Transport companies spend a large amount of money on proprietary solutions for tracking their fleet. A wireless phone-based solution will be very suitable in such cases, since it is global and likely to benefit from economies of scale. A vast geographical region is broken down into small areas called “cells”. Each cell has a radio tower and serves the area where its radio signal is strong enough. The radio towers are connected to the Base Transceiver System (BTS) which provides the signal processing capability. Radio resources for a group of BTSs are managed by a Base Station Controller (BSC). Connections from a group of BSCs are managed by the Mobile Switching Center (MSC), which is also the gateway to the Public Switched Telephone Network. Thus in network hierarchy, MSC is the top-level entity followed by the BSCs, followed by the BTSs and finally the mobile stations. The connectivity between the mobile station and the base station is through radio signals. As mobile moves from

one cell to another, its connection is broken with the former and re-established with the latter. Since all communications from the mobile station happens through radio waves, one has to rely on the properties of the radio waves in order to figure out where the source of the signal might b.e

2.4.1.2 Ground-Based RF Systems

Active beacons have been used for many centuries as a reliable and accurate means for navigation. Stars can be considered as active beacons with respect to navigation; and lighthouses were early man-made beacon systems. Typical applications for active beacon navigation include marine navigation, aircraft navigation, race car performance analysis, range instrumentation, unmanned mobile target control, mine localization, hazardous materials mapping, dredge positioning and geodetic surveys.

Modern technology has vastly enhanced the capabilities of active beacon systems with the introduction of laser, ultrasonic, and radio-frequency (RF) transmitters. It should be noted, though, that according to manufacturers, none of the RF systems can be used reliably in indoor environments.

Ground-based RF position location systems are typically of two types:

- Passive hyperbolic line-of-position phase-measurement systems that compare the time-of-arrival phase differences of incoming signals simultaneously emitted from surveyed transmitter sites.
- Active radar-like trilateration (triangulation) systems that measure the round-trip propagation delays for a number of fixed-reference transponders. Passive systems are generally preferable when a large number of vehicles must operate in the same local area, for obvious reasons.

2.4.1.3 Loran

An early example of the first category is seen in Loran (short for long range navigation). Developed at MIT during World War II, such systems compare the time of arrival of two identical signals broadcast simultaneously from high-power transmitters located at surveyed sites with a known separation baseline. For each finite time difference (as measured by the receiver) there is an associated hyperbolic line of position. Two or more pairs of master/slave stations are required to obtain intersecting hyperbolic lines resulting in a two-dimensional (latitude and longitude) fix.

The original implementation (Loran A) was aimed at assisting convoys of liberty ships crossing the North Atlantic in stormy winter weather. Two 100 kW slave transmitters were located about 200 miles on either side of the master station. Non-line-of-sight ground-wave propagation at around 2MHz was employed, with pulsed as opposed to continuous-wave transmissions to aid in sky-wave discrimination. The time-of-arrival difference was simply measured as the lateral separation of the two pulses on an oscilloscope display, with a typical accuracy of around 1 µs. This numerical value was matched to the appropriate line of position on a special Loran chart of the region, and the procedure then repeated for another set of transmitters. For discrimination purposes, four different frequencies were used, 50 kHz apart, with 24 different pulse repetition rates in the neighborhood of 20 to 35 pulses per second. In situations where the hyperbolic lines intersected more or less at right angles, the resulting (best-case) accuracy was about 1.5 kilometers.

Loran A was phased out in the early '80s in favor of Loran C, which achieves much longer over-the-horizon ranges through use of 5 MW pulses radiated from 400-meter (1300 ft.) towers at a lower carrier frequency of 100 kHz. For improved accuracy, the phase differences of the first three cycles of the master and slave pulses are tracked by phase-lock-loops in the receiver and converted to a digital readout, which is again cross-referenced to a preprinted chart. Effective operational range is about 1000 miles, with best-case accuracies in the neighborhood of 100 meters (330 ft.).

Coverage is provided by about 50 transmitter sites to all U.S. coastal waters and parts of the North Atlantic, North Pacific, and the Mediterranean.

2.4.1.4 Cell-based tracking

An example of such a system is the current cellular phone system[27],[30]. How and why should the cellular system know the location of a phone that is just quietly monitoring a paging channel, waiting either for the user to place a call or for a call to come in?

It has to do with efficiency. If cell phone users only placed calls and never received them, there would not be any need to track their locations, even when idle. But a substantial fraction of calls are made to cellular phones. When someone calls a cell phone, a message is sent over the paging channel to the phone. This is why the phone monitors this channel whenever it is on but idle. But which cell's paging channel should the system use to page the mobile? The system may have literally hundreds of cells or sectors, and the user might be in any one of them -- or indeed, nowhere at all if he's out of town or has his phone switched off. The system could simply send the page over every cell in the system repeatedly until the mobile answers or the system gives up -- a practice called flood paging - but this is obviously rather inefficient. It was done in the early days, before the number of cells and customers made it impractical. After all, each paging channel is only 10 kb/s, and each unanswered page has to be re-sent a reasonable number of times before the system can give up.

The alternative to flood paging is registration-based paging. That's where the phone announces itself to the system with a short message on the access channel so that the system knows exactly where to direct a page should an incoming call come in. If the mobile moves to another cell, it re-registers in that new cell and the system updates its database accordingly. The mobile also re-registers occasionally even if it stays in the same cell, just to refresh the database entry (the phone might be switched off without warning, or its battery could run down). The precision of such a system is limited by the cell dimension, and can be improved by measuring the signal strength [28],[29].

Another solution would be for the cell base stations to transmit their own signal, and the phones to be able to work out where they are from the relative skew of the signal from the nearest 3 or 4 base stations (similar to GPS with very low satellites).

2.4.1.5 The GUIDE system

The GUIDE [26] system has been developed to provide city visitors with a hand-held context-aware tourist guide, and used in the city of Lancaster (UK). The GUIDE end-system is composed of a TeamPad running Windows 95, and equipped with a PCMCIA-based wireless networking card. The network infrastructure that is used by the GUIDE system comprises a number of interconnected cells. The wireless network is based on Lucent Technologies' 802.11 compliant WaveLAN system, operating in the 2.4GHz band, and offering a maximum bandwidth of 2 Mbps per cell. Currently, six communication cells have been deployed within a region of the city that is popular with tourists.

Although, the range of WaveLAN is approximately 200m in free space, WaveLAN signals have very poor propagation characteristics through buildings and therefore, by strategic positioning of cell-servers, it is possible to create relatively small, asymmetric cells. Within the context of GUIDE this is a positive feature because by creating smaller, non-overlapping cells more accurate positioning information can be provided.

When visitors leave the cell coverage and up-to-date positioning information becomes unavailable, the GUIDE system tries to locate the visitor by establishing a form of partnership between itself

and the visitor. In more detail, the visitor is shown a series of thumbnail pictures showing attractions in the vicinity of the visitor's last location. Providing the visitor is then able to recognize and select one of the pictures, the GUIDE system tries once again to ascertain the visitor's location within the city.

2.4.1.6 Ultrasonic Tracking

The CONSTELLATION tracking system, proposed by Eric Foxlin [15] from InterSense, is similar in its basic principles of operation to an aided inertial navigation system (INS), except that it operates indoors, has much finer resolution and accuracy, and uses acoustic rather than RF technology for range measurements.

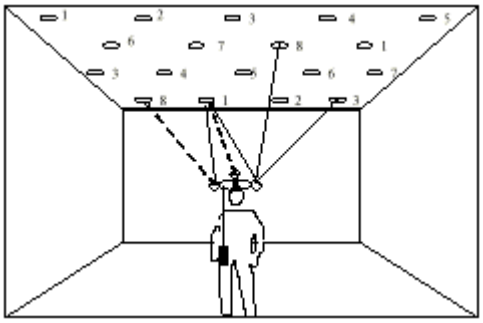

Figure 2-7. General Idea of Constellation system

Figure 2-7 illustrates the system, configured for tracking an HMD (Head Mounted Display) in a wide-range VR or AR application. The HMD is equipped with an integrated inertial sensing instrument called the InertiaCube™ and, in this example, 3 ultrasonic range-finder modules (URMs). The range-finder modules communicate with a constellation of transponder beacons, which may be mounted at any known locations in the environment.

Eric Foxlin describes the hardware that uses an InertiaCube [16] to sense angular rate and linear acceleration along each of three orthogonal body axes. We present it below.

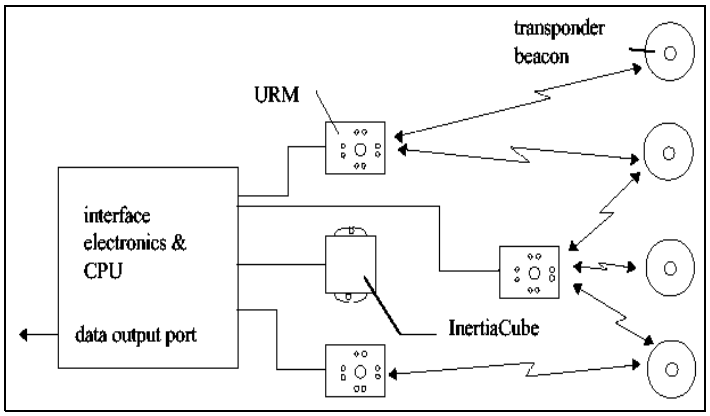

Figure 2-8. Hardware overview

Figure 2-8 illustrates the main hardware components of the tracking system. Just as GPS has a space-based constellation of satellites and a vehicle-borne receiver with antennae, this system has a ceiling-based constellation of transponder beacons and a camera or tracker unit worn by a person with ultrasonic range-finder modules (URMs) and an InertiaCube™ inertial sensing device.

2.4.1.7 Global Positioning Systems (Space-based Radio System)

The GPS (Global Positioning System) tracking principle uses 24 satellites (Figure 2-9) and 12 ground stations. The ground stations control the accuracy of the atomic clock and the orbit drift of the satellites. The system can determine the position of a user having a GPS receiver by the reception of three signals from three satellites and the computation of the TOF by subtracting the time of emission of the signals coming from the satellites from the time of reception. In practice, the receiver clock is not precise and has a bias, which is unknown. The use of a signal coming from another satellite eliminates the unknown bias. The resolution accomplished with such a system is in the order of the decameter. A more precise system, the differential GPS, uses emitting ground stations that refine the resolution to the order of meters.

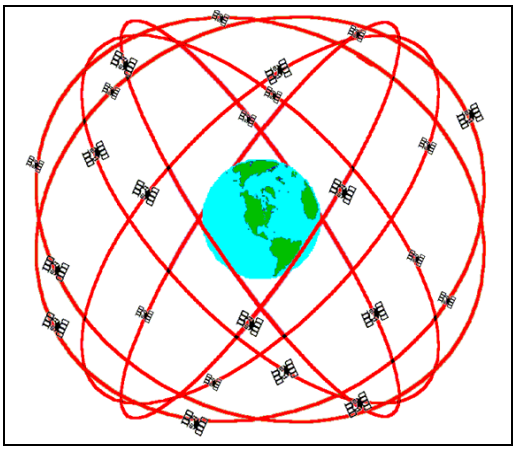

Figure 2-9. GPS Constellation - 24 Satellites in 6 Orbital Planes

The absolute three-dimensional location of any GPS receiver is determined through simple triangulation techniques based on time of flight for uniquely coded spread-spectrum radio signals transmitted by the satellites. Precisely measured signal propagation times are converted to pseudoranges representing the line-of-sight distances between the receiver and a number of reference satellites in known orbital positions. The measured distances have to be adjusted for receiver clock offset, as will be discussed later, hence the term pseudoranges. Knowing the exact distance from the ground receiver to three satellites theoretically allows for calculation of receiver latitude, longitude, and altitude.

Although conceptually very simple, this design philosophy introduces at least four obvious technical challenges:

- Time synchronization between individual satellites and GPS receivers.
- Precise real-time location of satellite position.
- Accurate measurement of signal propagation time.
- Sufficient signal-to-noise ratio for reliable operation in the presence of interference and possible jamming.

With the exception of multi-path effects, all of the error sources can be essentially eliminated through use of a practice known as differential GPS (DGPS). The concept is based on the premise that a second GPS receiver in fairly close proximity (i.e., within 10 km) to the first will experience basically the same error effects when viewing the same reference satellites. If this second receiver is fixed at a precisely surveyed location, its calculated solution can be compared to the known position to generate a composite error vector representative of prevailing conditions in that immediate

locale. This differential correction can then be passed to the first receiver to null out the unwanted effects, effectively reducing position error for commercial systems to well under 10 meters. The fixed DGPS reference station transmits these correction signals every two to four minutes to any differential-capable receiver within range. Many commercial GPS receivers are available with differential capability, and most now follow the RTCM-104 standard developed by the Radio Technical Commission for Maritime Services to promote interoperability.

Table 2-2. Summary of achievable position accuracies for various implementations of GPS

GPS IMPLEMENTATION METHOD	POSITION ACCURACY
C/A-code stand alone	100 m SEP(Spherical Error Probability) (328 ft.)
Y-code stand alone	16 m SEP(52 ft.)
Differential (C/A-code)	3 m SEP(10 ft.)
Differential (Y-code)	unknown (TBD)
Phase differential (codeless)	1 cm SEP(0.4 in)

2.4.2 Passive Landmarks

If the landmarks do not actively transmit signals, they are called passive landmarks. The mobile system has to actively look for these landmarks to acquire position measurements. Techniques using passive landmarks in determining the position of the mobile system rely on detection of those landmarks from sensor readings. The detection of landmarks depends on the type of sensor used. For example, in detecting landmarks in images from a vision system, image processing techniques are used. When three or more landmarks are detected by the system, it can use the triangulation or trilateration techniques to compute its location. Passive landmarks can be either artifcial or natural and the choice of which kind of landmarks to use can play a significant role in the performance of the localization system.

- Artificial Landmark Recognition - In this method distinctive artificial landmarks are placed at known locations in the environment. The advantage of artificial landmarks is that they can be designed for optimal detectability even under adverse environmental conditions. As with active beacons, three or more landmarks must be “in view” to allow position estimation. Landmark positioning has the advantage that the position errors are bounded, but detection of external landmarks and real-time position fixing may not always be possible. Unlike the usually point-shaped beacons, artificial landmarks may be defined as a set of features, e.g., a shape or an area. Additional information, for example distance, can be derived from measuring the geometric properties of the landmark, but this approach is computationally intensive and not very accurate.
- Natural Landmark Recognition - Here the landmarks are distinctive features in the environment. There is no need for preparation of the environment, but the environment must be known in advance. The reliability of this method is not as high as with artificial landmarks.

2.4.2.1 Geomagnetic Sensing

Vehicle heading is the most significant of the navigation parameters in terms of its influence on accumulated dead-reckoning errors. For this reason, sensors which provide a measure of absolute heading or relative angular velocity are extremely important in solving the real world navigation needs of a mobile user. The most commonly known sensor of this type is probably the magnetic

compass. The terminology normally used to describe the intensity of a magnetic field is magnetic flux density B, measured in Gauss (G).

The average strength of the earth's magnetic field is 0.5 Gauss and can be represented as a dipole that fluctuates both in time and space, situated roughly 440 kilometers off center and inclined 11 degrees to the planet's axis of rotation. This difference in location between true north and magnetic north is known as declination and varies with both time and geographical location. Corrective values are routinely provided in the form of declination tables printed directly on the maps or charts for any given locale.

A **magnetometer** or magnetic compass is the only low cost absolute heading reference presently available for augmented reality applications. Other absolute references, such as north-seeking gyros, are far too expensive. The serious drawback in using a magnetic compass for a mobile user is the hostile magnetic environment that can be encountered during navigation.

A magnetic compass senses the magnetic field of the Earth on two or three orthogonal sensors, sometimes in conjunction with a biaxial inclinometer. Since this field should point directly North, some method can be used to estimate the heading relative to the magnetic North pole. There is a varying declination between the magnetic and geodetic North poles, but models can easily estimate this difference to better than one degree.

The magnetic sensors are usually flux-gate sensors. The operation of a fluxgate is based on Faraday’s law, which states that a current (or voltage) is created in a loop in the presence of a changing magnetic field. A fluxgate is composed of a saturating magnetic core, with a drive winding and a pair of sense windings on it. The drive winding is wrapped around the core, which is normally a toroid. These sense windings are often wound flat on the outside of the core and are arranged at precisely 90° to each other. When not energized, a fluxgate’s permeability ‘draws in’ the Earth’s magnetic field. When energized, the core saturates and ceases to be magnetic. As this switching occurs (hence the name fluxgate), the Earth’s magnetic field is drawn into or released from the core, resulting in a small induced voltage that is proportional to the strength and direction of the external field.

2.4.2.2 Inclinometers

An **inclinometer** is a sensor used to measure the angle between the gravity vector and the platform to which it is mounted. This can be in a single direction, i.e. for sensing vehicle roll only, or two directions to estimate pitch as well. Inclinometers suffer an error due to vehicle accelerations since it is not possible to separate them from the gravity vector. It is possible to make corrections to the inclinometer output using gyros. However, GPS positioning cannot provide an accurate estimate of acceleration to compute this correction.

The liquid-bubble inclinometer uses a vial, partially filled with a conductive liquid, to determine the tilt of the platform in one or more axes. Electrodes around the vial estimate the liquid height or height difference between sensors. These measurements are converted to pitch or roll measurements. A diagram of a typical sensor is found in Figure 2-10

.

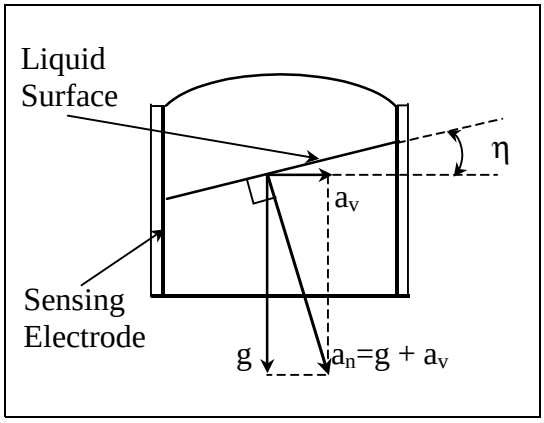

Figure 2-10. Bubble Inclinometer

The acceleration-related error suffered by bubble inclinometers is due to the fact that vehicle acceleration will make the liquid rise up one side of the vial. Also, there will be some sort of damped oscillation in the fluid even after the acceleration has finished. The error due to mobile unit acceleration can be given by:

$$\eta = \text{atan}\left(\frac{a_v}{g}\right) \qquad \text{eq. 2-4.}$$

where $\boldsymbol{g}$ is the magnitude of gravity and a_v is the acceleration of the mobile unit (see Figure 2-10).

2.4.2.3 Vision-Based Positioning

A core problem in mobile person tracking is the determination of the position and orientation (referred to as the pose) of a mobile person in its environment. The basic principles of landmark-based and map-based positioning also apply to vision-based positioning or localization, which relies on optical sensors [23] in contrast to ultrasound, dead-reckoning and inertial sensors. Common optical sensors include laser-based range finders and photometric cameras using CCD or CMOS arrays.

Visual sensing provides a tremendous amount of information about a mobile person's environment, and it is potentially the most powerful source of information among all the sensors used on mobile persons to date. Due to the wealth of information, however, extraction of visual features for positioning is not an easy task. The problem of localization by vision has received considerable attention and many techniques have been suggested. The basic components of the localization process are:

- representations of the environment,
- sensing models, and
- localization algorithms.

Most localization techniques provide absolute or relative position and/or the orientation of sensors. Techniques vary substantially, depending on the sensors, their geometric models, and the representation of the environment. The geometric information about the environment can be given in the form of landmarks, object models and maps in two or three dimensions. A vision sensor or multiple vision sensors should capture image features or regions that match the landmarks or maps. On the other hand, landmarks, object models, and maps should provide necessary spatial information that

is easy to be sensed. When landmarks or maps of an environment are not available, landmark selection and map building should be part of a localization method.

2.4.2.4 Camera Model and Localization

Geometric models of photometric cameras are of critical importance for finding the sensors' geometric position and orientation. The most common model for photometric cameras is the pin-hole camera with perspective projection as shown in Figure 2-11. Photometric cameras using an optical lens can be modeled as a pinhole camera. The coordinate system (X, Y, Z) is a three-dimensional camera coordinate system, and (x, y) is a sensor (image) coordinate system. A three-dimensional feature in an object is projected onto the image plane (x, y).

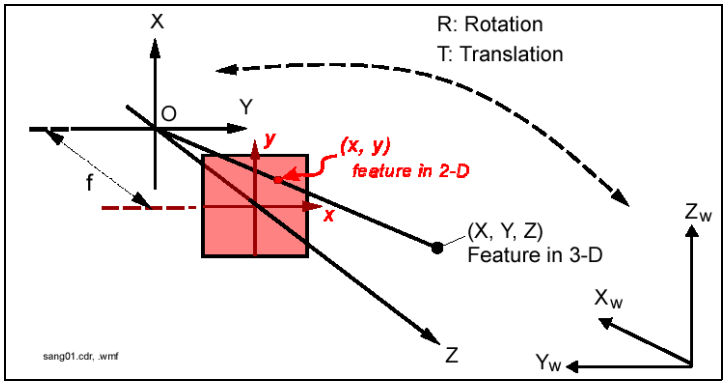

Figure 2-11. Perspective Camera Model

Although the range information is collapsed in this projection, the angle or orientation of the object point can be obtained if the focal length f is known and there is no distortion of rays due to lens distortion. The internal parameters of the camera are called intrinsic camera parameters and they include the effective focal length f, the radial lens distortion factor, and the image scanning parameters, which are used for estimating the physical size of the image plane [21]. The orientation and position of the camera coordinate system (X, Y, Z) can be described by six parameters, three for orientation and three for position, and they are called extrinsic camera parameters. They represent the relationship between the camera coordinates (X, Y, Z) and the world or object coordinates (Xw, Yw, Zw). Landmarks and maps are usually represented in the world coordinate system. The problem of localization is to determine the position and orientation of a sensor (or a mobile person) by matching the sensed visual features in one or more image(s) to the object features provided by landmarks or maps. Obviously a single feature would not provide enough information for position and orientation, so multiple features are required. Depending on the sensors, the sensing schemes, and the representations of the environment, localization techniques vary significantly.

The representation of the environment can be given in the form of very simple features such as points and lines, more complex patterns, or three-dimensional models of objects and environment. In this section, the approaches based on simple landmark features are discussed.

If a camera is mounted on a mobile person with its optical axis parallel to the floor and vertical edges of an environment provide landmarks, then the positioning problem becomes two-dimensional. In this case, the vertical edges provide point features and two-dimensional positioning requires identification of three unique features. If the features are uniquely identifiable and their positions are known, then the position and orientation of the pin-hole camera can be uniquely determined as illustrated in Figure 2-12a. However, it is not always possible to uniquely identify simple features such as points and lines in an image. Vertical lines are not usually interpretable unless a strong constraint is imposed. This is illustrated in Figure 2-12b.

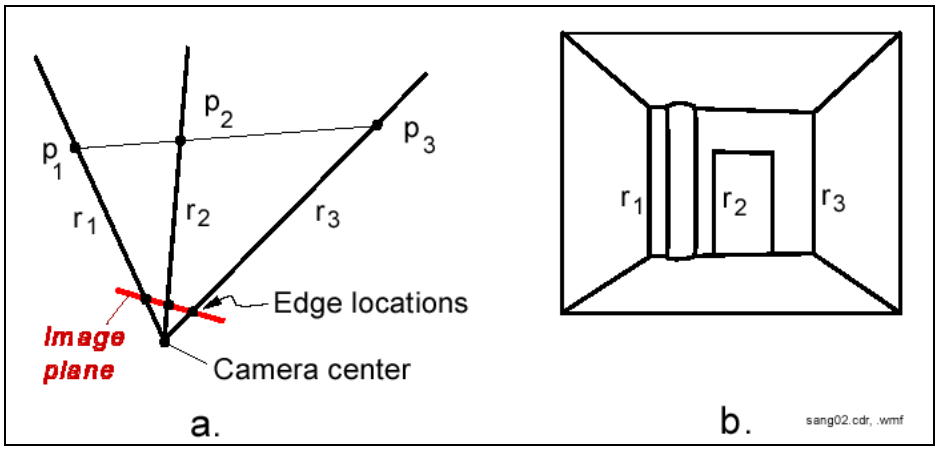

Figure 2-12. Localization using landmark features

2.4.3 Model-Based Approaches

Another group of localization techniques are map-based positioning or model matching techniques. These approaches use geometric features of the environment to compute the location of the mobile system. Examples of geometric features are the lines that describe walls in hallways or offices. Sensor output from, for example sonars or camera, is then matched with these features. Model matching can be used to update a global map in a dynamic environment, or to create a global map from different local maps.

In this method, information acquired from the mobile user computer onboard sensors is compared to a map or world model of the environment. If features from the sensor-based map and the world model map match, then the vehicle's absolute location can be estimated. Map-based positioning often includes improving global maps based on the new sensory observations in a dynamic environment and integrating local maps into the global map to cover previously unexplored areas. The maps used in navigation include two major types: geometric maps and topological maps. Geometric maps represent the world in a global coordinate system, while topological maps represent the world as a network of nodes and arcs.

A priori information about an environment can be given in more comprehensive form than features, such as two-dimensional or three-dimensional models of environment structure and digital elevation maps (DEM). The geometric models often include three-dimensional models of buildings, indoor structure and floor maps. For localization, the two-dimensional visual observations should capture the features of the environment that can be matched to the preloaded model with minimum uncertainty. Figure 2-13 illustrates the match between models and image features. The problem is that the two-dimensional observations and the three-dimensional world models are in different forms. This is basically the problem of object recognition in computer vision: (1) identifying objects and (2) estimating pose from the identified objects.

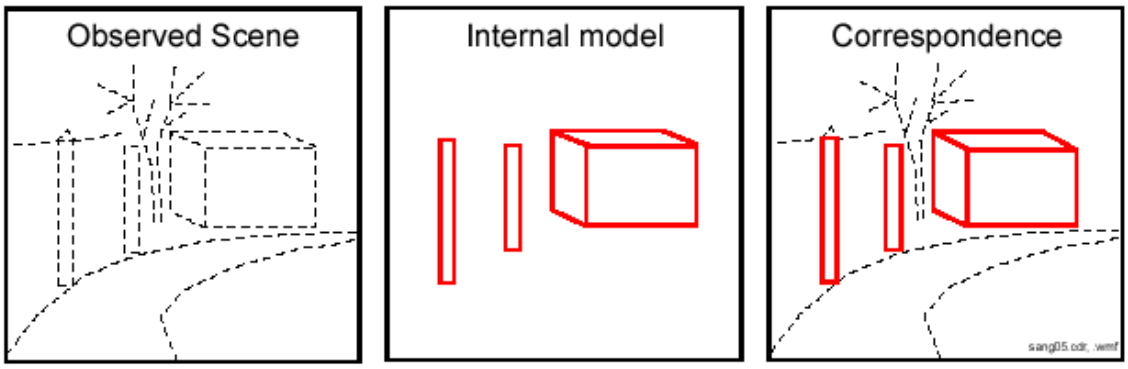

Figure 2-13. Finding correspondence between an internal model and an observed scene.

2.5 Multi-Sensor Fusion and Inertial Navigation

Algorithms that solve the localization problem combine initial information and relative and absolute position measurements to form estimates of the location of the mobile system at a certain time. If the measurements are considered to be readings from different sensors, the problem becomes how to combine the readings from different sensors to form a combined representation of the environment. This problem is studied by research in multi-sensor fusion.

Fusion of information from multiple sensors is important, since combined information from multiple sensors can be more accurate. Particularly when not all sensors are able to sense things to the same extent. Some features may be occluded for some sensors, while visible to others. Together the sensors can provide a more complete picture of a scene at a certain time. Multi-sensor fusion is also important since it can reduce the effects of errors in measurements.

A person walking around with an AR headset is comparable to a vehicle roaming around the earth. In the previous chapter we elaborated on the requirements of such an AR system, strapped onto the “human vehicle”, whose parameters differ from man-made earthbound or airborn vehicles, in terms of: a high update frequency, low latency, low power consumption, low ground speed, high head accelerations. The *navigation framework* for the “human vehicle” and man-made vehicles, however, remains the same.

Inertial navigation is the process of measuring acceleration on board a vehicle[1] and then integrating the acceleration to determine the vehicle’s velocity and position relative to a known starting point. Accelerometers are used to sense the magnitude of the acceleration, but acceleration is a vector quantity having direction as well as magnitude. For this reason a set of gyroscopes are used to maintain the accelerometers in a known orientation with respect to a fixed, non rotating coordinate system, commonly referred to as the *inertial space*. This does not mean that the accelerometers themselves are kept parallel to the axes of this non rotating coordinate system, although some systems are implemented this way.

Inertial navigation systems can be classified according to the way they perform the basic operations of inertial navigation. A navigation system is called geometric or gimbaled if the orientation of the navigation frame is physically maintained by system gimbals, and analytic or strapdown if the orientation of the navigation platform is maintained by the navigation algorithm.

1. Human, earthbound or airborne

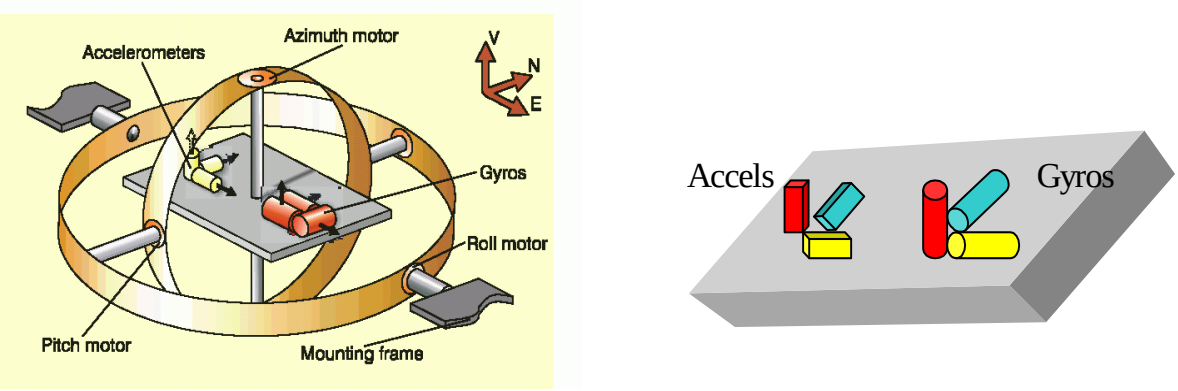

Figure 2-14. Gimbaled and Strapdown INS

In gimbaled navigation systems (see Figure 2-14), the inertial sensors are mounted on an actuated platform whose orientations are nominally stationary relative to the reference coordinate system. The gimbal angles are commanded to maintain the platform frame alignment with a specified navigation coordinate system. This is achieved by attempting to maintain the gyro outputs at the rotational rates computed for the navigation frame. If this is achieved, then the platform does not experience any rotation relative to the navigation frame, in spite of vehicle motion. In this approach, accelerometers aligned with the platform measure the specific force along the navigation coordinate system axes. Scaling and integration of this measured acceleration yield the desired navigation-frame position and velocity vectors. Vehicle attitude is determined by measurement of the relative angles between the vehicle and platform axes.

In a strapdown inertial navigation system, there is no gimbal to be maintained in alignment with a reference frame, so the orientation of a reference frame is maintained in the navigation algorithm as a mathematical coordinate transformation. Strapdown inertial navigation systems are rigidly fixed to the moving body. In this approach, the sensors move with the body, their gyros experiencing and measuring the same changes in angular rate as the body in motion, therefore higher bandwidth rate gyros with higher dynamic ranges are required. The strapdown INS accelerometers measure changes in linear rate in terms of the body's fixed axes. The body's fixed axes are a moving frame of reference as opposed to the constant inertial frame of reference. The navigation computer uses the gyros' angular information and the accelerometers' linear information to calculate the body's 3D motion with respect to an inertial frame of reference.

Actuated systems have smaller computational burdens and expose the inertial sensors to a more benign inertial environment, but are typically larger and more expensive than strapdown systems because of the requirement of the actuated platform. Stand-alone INSs require high accuracy for long periods of time (multiple days) and are built around accurately calibrated sensors and actuated platforms. Advances in sensor and computer technologies over the past decades have resulted in a shift towards strapdown systems. Aided strapdown systems are receiving renewed interest in applications, requiring high-accuracy and high rate outputs, while also being inexpensive, small, and low power.

2.6 Summary of Sensing Technologies

There are several sensing technologies [12],[14] in use, with the most common being magnetic, acoustic, and optical technologies. Table 2-3 provides an overview of the different technologies and their advantages and disadvantages.

Table 2-3. Tracking Technologies

TECHNOLOGY	DESCRIPTION	STRENGTHS	WEAKNESSES
Mechanical	Measure change in position by physically connecting the remote object to a point of reference with jointed linkages	Accurate Low lag No line of sight (LOS) problems No magnetic interference problems Good for tracking small volumes accurately	Intrusive, due to tethering Subject to mechanical part wear-out
Magnetic	Use sets of coils (in a transmitter) that are pulsed to produce magnetic fields Magnetic sensors (in a receiver) determine the strength and angles of the fields Pulsed magnetic field may be AC or DC	Inexpensive Accurate No LOS problems Good noise immunity Map whole body motion Large ranges; size of a small room	Ferromagnetic and/or metal conductive surfaces cause field distortion Electromagnetic interference from radios Accuracy diminishes with distance High latencies due to filtering
Sourceless, Non-inertial	Use passive magnetic sensors, referenced to the earth's magnetic field, to provide measurement of roll, pitch, and yaw, and as a derivative, angular acceleration and velocity	Inexpensive Transmitter not necessary Portable	Only 3 DOF Difficult to mark movement between magnetic hemispheres
Optical	Use a variety of detectors, from ordinary video cameras to LEDs, to detect either ambient light or light emitted under control of the position tracker. Infrared light is often used to prevent interference with other activities	High availability Can work over a large area Fast No magnetic interference problems High accuracy	LOS necessary Limited by intensity and coherence of light sources High weight Expensive
Inertial	Use accelerometers and gyroscopes. Orientation of the object is computed by jointly integrating the outputs of the rate gyros whose outputs are proportional to angular velocity about each axis. Changes in position can be computed by double integrating the outputs of the accelerometers using their known orientations	Unlimited range Fast No LOS problems No magnetic interference problems Senses orientation directly Small size Low cost	Only 3 DOF Drift Not accurate for slow position changes

Table 2-3. Tracking Technologies

TECHNOLOGY	DESCRIPTION	STRENGTHS	WEAKNESSES
Acoustic (Ultrasound)	Use three microphones and three emitters to compute the distance between a source and receiver via triangulation. Use ultrasonic frequencies (above 20 kHz) so that the emitters will not be heard	Inexpensive No magnetic interference problems Light weight	Ultrasonic noise interference Low accuracy since speed of sound in air varies with environmental conditions Echoes cause reception of "ghost" pulses LOS necessary
Radio / GSM based location	Use three or more transmission radio emitters to compute the distance between a source and receiver via triangulation. Can use different modalities to obtain the location: measuring signal attenuation, angle of arrival, time difference of arrival	Inexpensive because the existing GSM infrastructure can be used. Light weight	Radio noise interference Low accuracy Problem with multi-path reception and interference

Position and orientation trackers can be described in terms of a small set of key characteristics that serve as performance measures for their evaluation and comparison. Meyer et al. (1992) define these characteristics as resolution, accuracy, and system responsiveness (additional characteristics of robustness, registration, and sociability are not considered here).

- Resolution. Measures the exactness with which a system can locate a reported position. It is measured in terms of degrees for orientation and cm per cm of transmitter and receiver separation for position.
- Accuracy. The range within which a reported position is correct. This is a function of the error involved in making measurements and often it is expressed in statistical error terminology as degrees root mean square (RMS) for orientation and cm RMS for position.
- System responsiveness. Comprises:
    - Sample rate: The rate at which sensors are checked for data, usually expressed as frequency.
    - Data rate: The number of computed positions per second, usually expressed as frequency.
    - Update rate: The rate at which the system reports new position coordinates to the host computer, also usually given as frequency.
    - Latency: also known as lag, is the delay between the movement of the remotely sensed object and the report of the new position. This is measured in milliseconds (ms).

These characteristics provide some guidance for tracker performance. One of the most important is latency. Durlach [24] states that delays greater than 60 msec. between head motion and visual feedback impair adaptation.

Latencies between systems are difficult to compare because they are not always calculated the same. Bryson [25] identifies several sources of latency: delays in the tracker signal, delays in communication between the tracker and the computer system, delays due to computations required to process the tracker data, and delays due to graphical rendering. However, several manufacturers suggested that 1/frequency is the preferred measure.

Sometimes the application also requires the head movement information. One important parameter of a head tracker is its responsiveness. With respect to responsiveness, Durlach [24] contends that head movements can be as fast as 1,000°/sec. in yaw, although more usual peak velocities are 600°/sec. for yaw and 300°/sec. for pitch and roll. The frequency content of volitional head motion falls off approximately as $1/f^2$, with most of the energy contained below 8 Hz and nothing detectable above 15 Hz. Tracker-to-host reporting rates must, therefore, be at least 30 Hz.

An additional important characteristic that is included is working volume or range, which may be bound by intrinsic limitations such as mechanical linkage or signal strength. This is the volume in which a position tracker accurately reports position.

2.7 Conclusions

Each tracking approach has limitations. Noise, calibration error, and the gravity field impart errors on the signals, producing accumulated position and orientation drift. Position requires double integration of linear acceleration, so the accumulation of position drift grows as the square of elapsed time. Orientation only requires a single integration of rotation rate, so the drift accumulates linearly with elapsed time. Hybrid systems attempt to compensate for the shortcomings of each technology by using multiple measurements to produce robust results.

No single tracking technology has the performance required to meet the stringent needs of outdoor or indoor positioning. However, appropriately combining multiple sensors may lead to a viable solution sooner than waiting for any single technology to solve the entire problem. The system described in this paper is a first step in this process.

To simplify the problem, we assume real-world objects are distant (e.g., 50+ meters), which allows the use of GPS for position tracking. In order to develop a robust system, one set of sensors must provide information in the case of data loss of another. For example, accelerometer data could be used for short periods to get the position when the GPS sensor has not enough satellites in view. Also one set of sensors could be used to estimate and compensate for the inaccuracy of another, for example, the inclinometer data could be used to compensate for gyro drift.

In summary:

- The following are fast relative pose sensors: Accelerometers, Gyroscopes, Tilt sensors, Magnetometers.
- The following are absolute positioning sensors: Grid of active beacons (IR, Ultrasound, RF), Cell-Based Tracking, Differences in time-of-arrival measurement on multiple receivers , GPS, Vision-based sensors.
- Visual sensing provides a tremendous amount of information about one's environment, and it is potentially the most powerful source of information among all tracking sensors. However, extraction of visual features and filtering them for positioning is not an easy task. The geometric information about the environment can be given in the form of active and passive landmarks, both artificial and natural, object models and maps in two or three dimensions. In every cases, the features found should be matched with knowledge of the environment provided by the network. Although much is possible in this technology, a severe constraint is processing time.
- Sensor data fusion can be performed by Kalman or Extended Kalman filtering

Chapter 3

Sensor Selection, Errors and Calibration

3.1 Introduction

A major issue in headtracking for augmented reality systems is to create such a high speed, low latency system that the chance of the user experiencing motion sickness is low. In this chapter we will review the various physical sensors, such as gyroscopes, accelerometers, magnetometers, inclinometers and DGPS, that are available for pose tracking. We will subsequently discuss the realization of a sensor cube and end this chapter with the initial calibration and alignment of stand-alone sensors.

This chapter presents the hardware components we have developed to operate interactive AR applications in an outdoor environment [8]. AR requires that equipment such as an HMD, computer, and tracking hardware be worn outdoors, with some early examples being the Touring Machine by Feiner et al. [4]. This research requires new hardware components capable of supporting the desired interactions. While commercially available components are used in many cases, these must be modified to suit the requirements of the mobile AR task. In some cases, components cannot be purchased off the shelf, and so must be designed and constructed. This chapter describes the integration of existing components and the development of a custom backpack, helmet, and positioning system to support the research in this thesis.

The computer currently being used is a Pentium-III 1.2 GHz processor, 512 MB of memory, and a 20 GB hard drive. Most importantly of all, it contains two VOODOO graphics cards that are capable of rendering complex 3D texture-mapped graphics with hardware accelerated OpenGL support [73]. Using two Lithium-Ion batteries it can operate for more than 2 hours.

The head-mounted display is an iGlasses ProTek with a maximum resolution of 800x600. This display can be used for either video or optical-based augmented reality, and has one of the highest quality displays available.

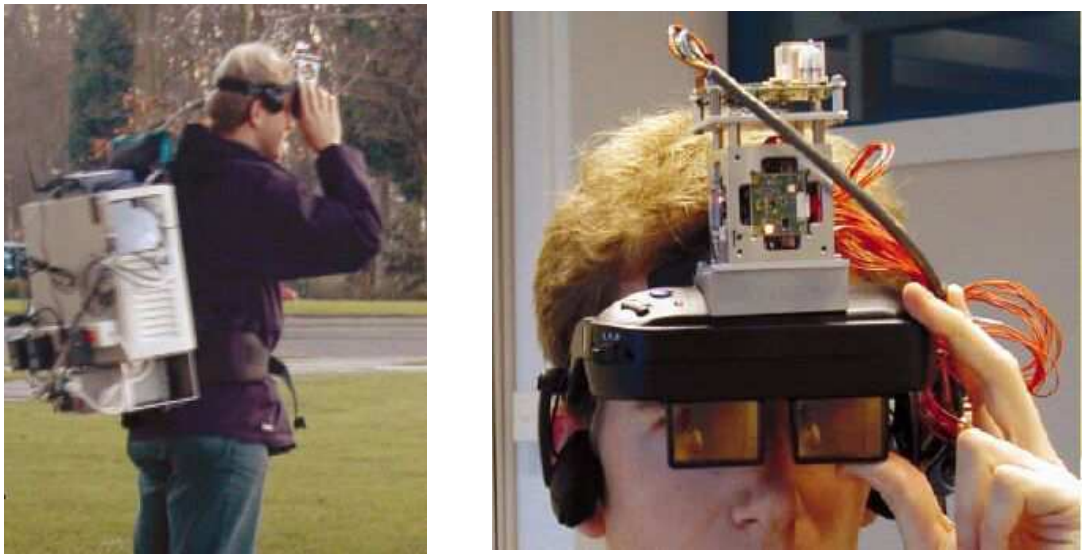

Figure 3-1. UbiCom AR demonstrator - the system and headset

The body position tracker used is a a Garmin 25 GPS, which is low quality with 5-10 meter accuracy. This device is a low power and low cost solution. The device outputs NMEA 0183 updates via an RS-232 serial port at 1 Hz, making it suitable for position sensing in outdoor AR. The head orientation tracker is an custom-made hybrid magnetic and inertial tracker, with the sensor cube mounted on the helmet. This device produces updates via an RS-232 serial port at up to 100 Hz. The development process of this hybrid inertial measurement unit is described in this chapter.

3.2 Building an Inertia Measurement Unit

The instrument cluster that implements an inertia measurement unit (IMU), usually includes a number of gyros and accelerometers which provide measurements of angular rates and specific forces respectively. Such a unit may contain three single-axis gyros, as well as three single-axis accelerometers, all attached to a rigid block which can be mounted on a mobile user. The sensors on such a cube can be augmented with other sensors like three magnetometers, two axis liquid inclinometers, etc. The sensitive axes of the sensors are most commonly mutually orthogonal in a Cartesian reference frame. This arrangement of the instruments allows the components of angular rate and acceleration in the three mutually orthogonal directions to be measured directly, thus providing the information required to implement the strapdown computing tasks.

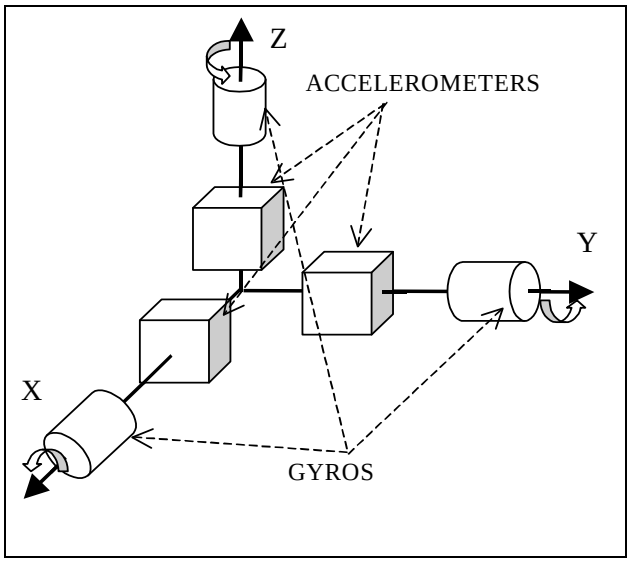

Figure 3-2. Orthogonal sensor cluster arrangement

In selecting components, we consider two sets of goals gleaned from the above design. The first are the functional requirements - 6 DOF sensing. The second are the usability goals - small, as well as low cost and low power.

In the case of the accelerometers, the Analog Devices components are notably superior for our applications. They provide adequate accuracy and bandwidth as well as excellent power drain and price per axis. The ability to switch between the pin compatible 2 g and 10 g version is very useful as well, as are the dual analog and digital outputs. Finally, ADXL202 is small, with a footprint of 5.0 × 5.0 × 2.0 mm.

In selecting gyroscopes, the matter is not quite as simple. While the Gyration[2] gyroscope has the best power drain and price, as well as good accuracy, it is simply far too large for our design. While far from perfect, the Murata gyroscopes are small and have reasonable performance and price, and will therefore be used in this design. The Crossbow[18] gyroscopes' specifications clearly demonstrate that an order of magnitude increase in price will buy an order of magnitude increase in noise performance, these are not appropriate in this case because of their large size, cost and power drain.

3.2.1 Position and Orientation Sensing Hardware

The hardware that we used in our design was:

- A Garmin GPS 25 LP receiver combined with an RDS OEM4000 system to form a DGPS unit
- A Precision Navigation TCM2 compass and tilt sensor
- Three rate gyroscopes (Murata)
- Three accelerometers (ADXL202)
- A light-weight, low power LART platform for (mobile) data processing

The LART platform was developed at the Delft University of Technology [96] (Figure 3-3). This Linux-based system contains an 8-channel fast 16-bit AD-converter to acquire synchronous data from the accelerometers, gyros and -in future- from temperature data. The latter is useful to compensate the drift due to temperature variations in the sensors.

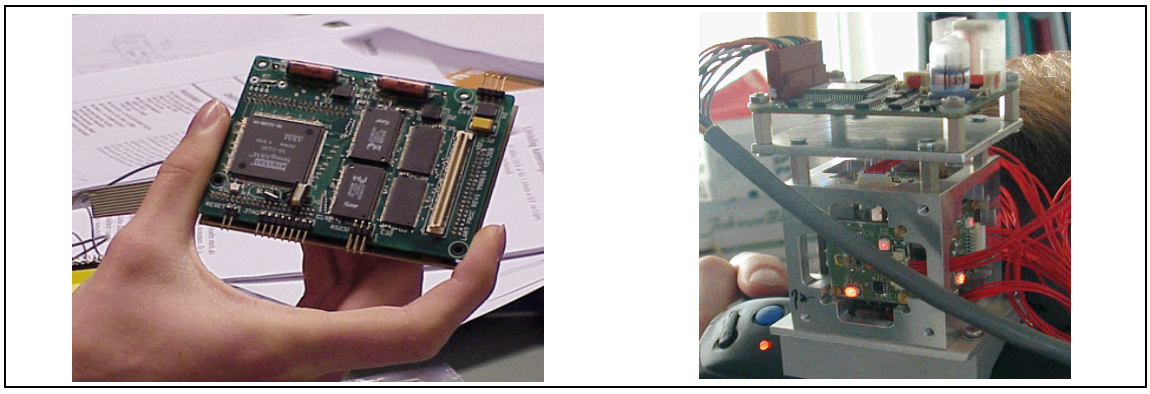

Figure 3-3. The LART board and the sensor cube (IMU)

The Garmin GPS provides outputs at 1 Hz, with an error of 5-10 m and an error of 2-3 m in a DGPS configuration. The TCM2 updates at 16 Hz and claims ±0.5 degrees of error in yaw. The gyros and the accelerometers are analog devices, which are sampled at 100 Hz by an AD converter on the LART board. The other sensors are read via a serial line [74].

3.2.2 Gyroscopes

The Murata Gyrostar piezoelectric vibrating gyro uses an equilateral triangular bar composed of elinvar (elastic invariable metal) with three PE ceramics attached to the sides. The ceramics are made from Murata's patented Piezotite materials, which claim better temperature stability and electro-mechanical coupling than other PE materials [85]. The bar is excited into motion from two sides, and the third side is used to drive a feedback loop, which controls the bar's oscillation. The drive ceramics are also used to detect the motion of the bar. A diagram of the bar is shown in Figure 3-4.

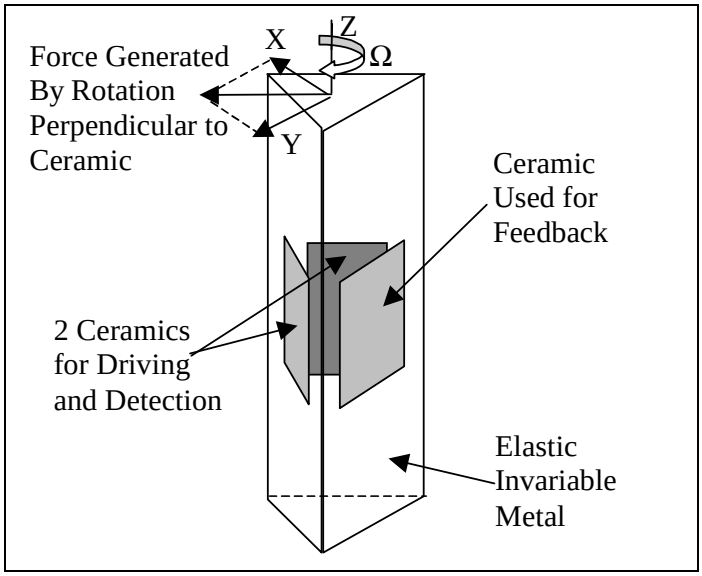

Figure 3-4. Gyrostar Free-Free-Bar and Ceramics

The bars can be very easily tuned to match the driving (resonant) and detecting frequency of both detectors compared to the rectangular bars that were developed in earlier days. This is because adjustments tothe edges of the prism only affect one flexural mode of vibration. Rotation detection

is also innovative due to the process of differencing the output from the two detectors. The phase angles and magnitudes of the vibration are compared to give a direction and magnitude of rotation. The specifications of Murata gyro (ENC-03J series) are found in Table 3-1.

Table 3-1. Selected Murata Gyrostar Specifications

Characteristics	Min.	Std.	Max	Unit
Supply Voltage	+2.7	+3.0	+5.5	VDC
Current @3.0VDC	2.5	3.2	4.5	mA
Max Angular Rate	-300	-	+300	deg/s
Output at zero rate	+1.25	+1.35	+1.45	VDC
Scale Factor	-20%	0.67	+20%	mV/deg/s
Temperature Scale	-20	-	+10	%FS
Linearity	-5	-	+5	%FS
Response		DC-50		Hz
Offset Drift	-	-	>9	deg/s
Dimension	15.5x8.0x4.3			mm

This gyro was selected in our project for its low cost (approximately $15 in large quantities), small size, light weight, and good performance relative to the others in its class. The voltage output from the gyro is about 1.35 V when stationary and extends about 0.201 V in each direction under 300°/s of rotation.

The Murata gyro has instability in both the zero-rotation output voltage and the scale, mainly due to temperature variations. Since we are aiming at mobile users, the sensor cannot be kept in a controlled environment, and since the cost of using a temperature-controled enclosure is restrictive, regular calibration of these values is required. The gyro also demonstrated limited endurance to shock and some susceptibility to vibration.

Difficulties also arise in trying to calibrate both the zero-rotation offset and scale factor simultaneously without a precise reference. The zero-rotation offset is the primary cause of heading drift and is easier to calibrate. The scale factor can only be calibrated by analyzing a series or history of turns to see if there is a tendency to overestimate or underestimate rotation. The scale factor does not seem to vary nearly as much as the zero-rotation offset. The scale factor was determined by scaling the integrated output of the gyro over 10 circles ($\Delta\theta=10*(2\pi)$), on a level surface. The resulting value was used throughout the testing.

3.2.3 Accelerometers

The sensors are surface micro-machined as in some standard integrated circuits (IC), and are therefore available in standard IC sizes ready to be mounted on a printed circuit board (PCB). The moving parts are cast in place using a sacrificial layer of material that is then dissolved, making the positioning of the parts highly accurate. On the single silicon die, there is the mass-spring system and also the entire electrical circuit to calculate the acceleration from the measured displacement of the mass. This is the circuit surrounding the proof mass. Analog Devices named this technology iMEMS, for integrated MEMS. The final output of the chip is an analog voltage, ranging from 0 to 5 volts, linearly proportional to the acceleration.

The specifications of Analog Devices ADXL105 accelerometer are found in Table 3-2.

Table 3-2. Selected ADXL105 Specifications

Characteristics	Min.	Std.	Max	Unit
Supply Voltage	+2.7		+5.25	VDC
Current @3.0VDC		1.9	2.6	mA
Measurement Range	±5	±7		g
Nonlinearity		0.2		%FS
Scale Factor	225	250	275	mV
Temperature Scale		±0,5		%
Alignment Error		±1		deg
Response	10	12		kHz
Offset Drift		50		mV
Noise Performance		225	325	$\mu g\sqrt{Hz}$

The typical noise floor is $225\mu g\sqrt{Hz}$ allowing signals below 2mg to be resolved. A 10kHz wide frequency response enables vibration measurement applications. The ADXL105 can measure both dynamic accelerations (vibrations), or static accelerations (such as inertial force, gravity or tilt). Output scale factors from 250mV/g to 1.5V/g can be set using the on-board uncommitted amplifier and external resistors. The device features an on-board temperature sensor with an output of 8mV/ ^{o}C for optional temperature compensation of offset vs. temperature for high accuracy measurements.

3.2.4 Magnetometers

It should be possible to compensate for the magnetic field of the mobile system since it is constant and keeps its orientation with the user, while the Earth's magnetic field always points North. However, disturbances such as moving metal objects like cars, or a variable magnetic field generated by high voltage lines can change the magnetic field. Such temporary external disturbances such as nearby vehicles cannot be calibrated out. Since the total magnetic field strength should not vary with orientation, magnetic disturbances are often identified by checking that total measured field strength variation does not exceed a threshold. The magnetic field of the Earth has a strength of about 0.5 G, but this value, or certainly its horizontal component, can easily be exceeded by the magnetic field of the mobile system.

The horizontal components of the magnetic field are needed to compute the heading relative to the North magnetic pole. If the sensor is tilted with respect to the local level plane, some amount of the vertical component will be sensed by the horizontal axis sensors. For this reason, a two-axis sensing compass can not properly determine the heading of the vehicle if the vehicle is tilted. A sensor with three magnetic sensing axes can determine the orientation of its axes only if the magnitude of the magnetic field is known or assumed. The addition of an inclinometer allows for full determination of the strength and orientation of the magnetic field with respect to the sensing apparatus. A method developed by Plessey Overseas for handling a complement of three magnetic sensing coils and a biaxial inclinometer, similar to the Precision Navigation TCM-2 [20] used for this project, is presented below.

If a three-axes magnetometer is operated without the presence of spurious magnetic fields, the measurement locus of the magnetic field of the Earth would appear as a sphere by allowing all values

of pitch, roll, and heading for the sensor. In the case of the sensor being mounted on a mobile unit which has a static magnetic field, the locus becomes an ellipsoid [20] that is shifted from the origin and has arbitrary orientation and size, depending on the permanent magnetic field of the mobile unit.

The TCM2 uses a patented magneto-inductive sensing technique that makes use of a material whose inductance varies with the magnetic field strength to which it is subjected. The compass provides RS232 proprietary or National Marine Electronics Association (NMEA) format output, but a proprietary format can be chosen if an increased level of detail is needed. The unit specifications are given in Table 3-3.

Table 3-3. TCM2-50 Digital Compass Specifications (Precision Navigation 1999)

Parameter	Specifications	Unit
Heading Accuracy (Level)	1.0	degree RMS
Heading Accuracy (Tilted)	1.5	degree RMS
Tilt Accuracy	$\pm 0{,}2$	degree
Tilt Range	± 50	degree
Magnetometer Accuracy	$\pm 0{,}2$	μT
Magnetometer Range	± 80	μT
Supply Current	15 - 20	mA
Interface	Digital: RS232C, NMEA0183	

It is possible to calibrate the TCM2 sensor to take into account the static magnetic field of the mobile unit. The calibration process involves making slow turns through varied pitch and roll. A calibration score is given which indicates whether enough data with varied heading, pitch, and roll variation was collected to estimate the field of the vehicle. To avoid saturation of the sensors and difficulty in eliminating the permanent field, the contribution of a car should be less than 0.3 G.

3.2.5 Global Positioning

Each satellite broadcasts a unique coarse acquisition (C/A) pseudo-random noise (PRN) code, modulated onto the L1 carrier. Selective availability (SA) was used until May 1, 2000, to degrade the performance of civilian users in single-point mode through dithering of the satellite clock offset. A 50 bps navigation message, which gives the time of transmission of the PRN sequence and other necessary information, and a second code (P) are modulated on both the L1 and L2 carriers. This P code is intended primarily for military users and provides better resolution (less receiver noise) and immunity from the SA, but is encrypted through a procedure known as anti-spoofing (AS). Civilians can still make use of the P code, but a loss is suffered through correlation techniques used to sidestep the encryption, and the SA is not removed.

The master control station (MCS) processes range measurements taken at the five monitoring stations and develops predictions for the orbits and satellite clock behavior. The MCS then sends this data to the monitor stations for upload of the navigation message to the satellites. The navigation message also includes the health of the satellites and an almanac that can be used to predict the visibility of satellites for any time and place.

3.2.5.1 GPS positioning

Four or more satellites are normally required to compute a GPS position due to the use of low cost oscillators in commercial GPS receivers [95]. If the receiver clock were synchronized with GPS time, only three range observations would be required to compute the receiver coordinates in 3D space. The fourth unknown, referred to as receiver clock bias, is the difference between the time estimated by the receiver and the GPS time. A range measurement with an error in time synchronization is referred to as a pseudorange. Pseudoranges are most often used in a least squares (LS) parametric model to solve the four unknowns in code positioning.

Three observations are normally made from the signal tracking procedures of a GPS receiver. These are the pseudorange, carrier phase, and Doppler measurements. The code pseudorange measurement is derived from a delay lock loop (DLL) which correlates the incoming signal with locally generated versions of the signal. The discriminator uses the output of the correlators to make a pseudorange measurement. The carrier phase measurement is an accurate measure of the phase angle and accumulate cycles of the incoming signal. The carrier signal is normally tracked using a Costas phase locked loop (PLL). A standard digital PLL is not used since the navigation message is usually left modulated on the carrier, making the correlation positive or negative depending on the bits of the navigation message. When using Costas discriminators, the tracking error must be kept below 90°, unlike a standard PLL, which can tolerate errors of up to 180°. The Doppler is an estimate of the frequency difference between the locally generated carrier and the Doppler shifted incoming signal. It is noted that the ionosphere causes a code delay and an equivalent advance of the carrier phase. The ranges are subject to several error sources, which are summarized in Table 3-4 for a typical C/A code receiver [90].

Table 3-4. Typical Errors for C/A Code Receiver

Error Sources	Typical Values
Troposphere Delay	2-30m
Ionosphere Delay	2-30m
Ephemeris (Orbital) Error	1-5m
Clock Drift	0.1-3m
Selective Availability	1-70m
Carrier Noise	0.001-0.006m
Carrier Multipath	0.001-0.02m
Code Noise	0.1-3m
Code Multipath	0.1-100m

GPS errors are a combination of noise, bias, and blunders:

- Noise errors are the combined effect of code noise (around 1 meter) and noise within the receiver noise (around 1 meter). Bias errors result from Selective Availability and other factors.
- Selective availability (SA): SA is the intentional degradation of the GPS signals by a time varying bias. SA is controlled by the DoD to limit accuracy for non-U. S. military and government users. However, SA was removed as of May 2, 2000, and this has increased the location accuracy by 10 times (Figure 3-5).
- Other bias error sources include clock errors and errors due to atmospheric effects

- Blunders can result in errors of hundreds of kilometers and can be caused by control segment mistakes, and human mistakes; receiver errors from software or hardware failures can cause blunder errors of any size.

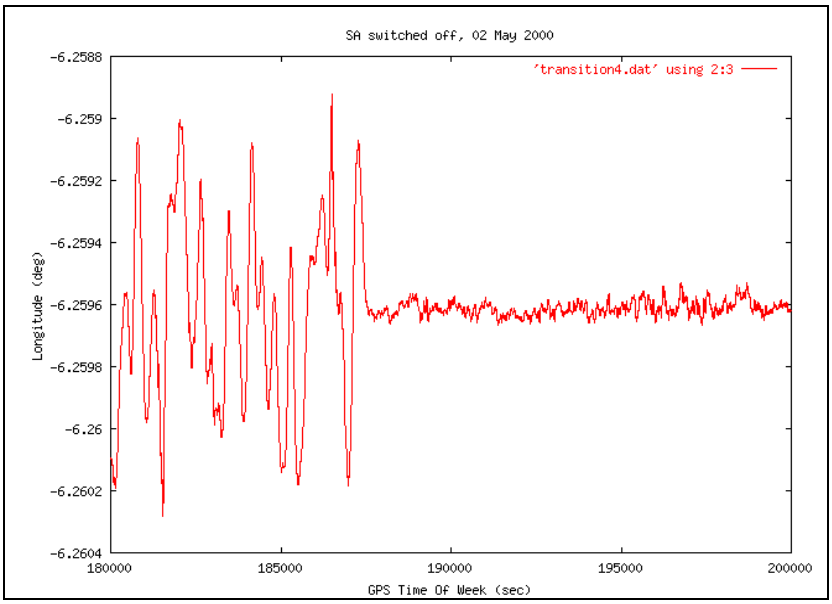

Figure 3-5. The effect of SA

Noise and bias errors combine, resulting in errors typically ranging around fifteen meters for each satellite used in the position solution. Atmospheric conditions influence accuracy because the ionosphere and troposphere both refract the GPS signals. This causes the speed of the GPS signal in the ionosphere and troposphere to be different from the speed of the GPS signal in space. Therefore, the distance calculated from “Signal Speed x Time” will be different for the portion of the GPS signal path that passes through the ionosphere and troposphere and for the portion that passes through space.

3.2.5.2 The Garmin GPS25 GPS receiver

The GPS 25LPs [19] is simultaneously tracking up to twelve satellites providing fast time-to-first-fix, one-second navigation updates and low power consumption. Its far-reaching capability meets the sensitivity requirements of land navigation as well as the dynamics requirements of high performance aircraft.

Some performance figures for the GPS25 receiver are as follows:

1. It tracks up to 12 satellites (up to 11 with PPS (pulse per second) active)
2. Update rate: 1 second
3. Acquisition time: 15 seconds warm (all data known), 45 seconds cold (initial position, time and almanac known, ephemeris unknown), 1.5 minutes AutoLocateTM (almanac known, initial position and time unknown), 5 minutes search the sky (no data known)
4. Position accuracy:

   Differential GPS (DGPS): Less than 5 meters RMS

   Non-differential GPS: 15 meters RMS (100 meters with Selective Availability on)
5. Velocity accuracy: 0.2 m/s RMS steady state (subject to Selective Availability)

6. Dynamics: 999 knots velocity, 6g dynamics
7. One-pulse-per-second accuracy: ±1 microsecond at rising edge of PPS pulse (subject to Selective Availability)

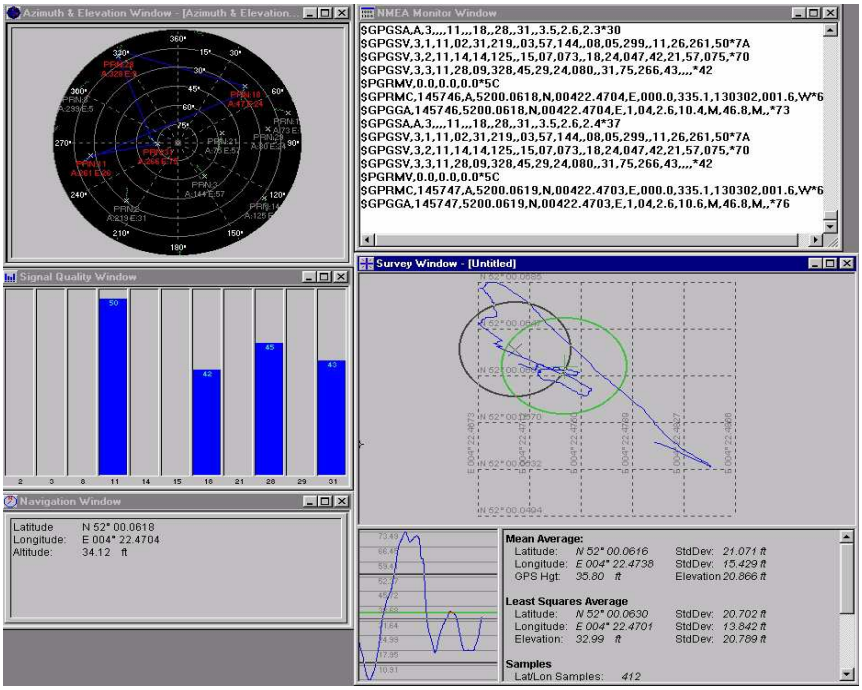

Figure 3-6. GPS navigation session visualization

We used the program VisualGPS to display and analyze the data from the Garmin GPS board. Figure 3-6 presents a short navigation session. The program displays also the number of satellites in view and the signal strength.

3.2.5.3 GPS protocols

All standard GPS hardware communications use the NMEA-0183 standard for marine electronic devices. Most GPS hardware also supports a variety of additional proprietary communication protocols, but NMEA-0183 is the ubiquitous standard that is supported by most software libraries. The NMEA-0183 standard specifies that communication between devices takes place through a standard serial link running at 4800 bps. The NMEA-0183 protocol consists of ASCII “sentences” which are sent repeatedly by the GPS receiver. These sentences always start with the character $ and end with a carriage return/newline (CRNL) sequence. The format is:

${talker id}{sentence id},{comma separated list of fields...}{optional checksum}\r\n

The talker id is a two-letter code that indicates the type of device sending the message. This will always be “GP” when reading data from a GPS receiver. The sentence id is a three-letter code that indicates the type of information being sent and the format of the following data fields. The different sentence types that can be parsed by the C++ implemented library are:

- GGA
- GLL
- RMC

These are common sentence types supported by almost all GPS systems. A detailed description of these sentences can be found in Chapter A.7.

3.2.5.4 Spatial reference systems

To measure locations accurately, a selected ellipsoid should fit an area of interest. Therefore, a horizontal (or geodetic) datum is established, which is an ellipsoid but positioned and oriented in such a way that it best fits the area or country of interest. There are a few hundred of these local horizontal datums defined in the world. Vertical datums are used to measure heights given on maps. The starting point for measuring these heights are mean-sea-level (MSL) points established at coastal sites. Starting from these points, the heights of points on the earth's surface can be measured using levelling techniques.

To produce a map, the curved reference surface of the Earth, approximated by an ellipsoid or a sphere, is transformed to the flat plane of the map by means of a map projection. In other words, each point on the reference surface of the Earth with geographic coordinates (φ, λ) may be transformed to a set of cartesian coordinates (x,y) representing positions on the map plane.

Most countries have defined their own local spatial reference system. We speak of a spatial reference system if, in addition to the selected reference surface (horizontal datum) and the chosen map projection, the origin and axes of the map coordinate system have been defined. The system used in the Netherlands is called the “Rijks-Driehoeks” system. The system is based on the azimuthal stereographic projection, centred in the middle of the country. The Bessel ellipsoid is used as the reference surface. The origin of the coordinate system has been shifted from the projection centre towards the south-west.

The RijksDriehoeksmeting is also a division of the Dutch Cadaster that is responsible for the maintenance of surveying networks in the Netherlands. Surveyors work with coordinates and a certain choice had to be made. The choice in the Netherlands was to apply a stereographic projection on an imaginary plane whose origin coincides with the tip of the O.L. Vrouwe Church in the city of Amersfoort. Tips on the towers of churches are quite visible throughout the countryside, this is the only reason why churches are so popular with surveyors who, after all, need benchmarks for their measurements. The stereographic projection is chosen for pure convenience, the reason is that angles measured in the terrain become identical to angles measured in the projected coordinates. RD coordinates can be found on all topographic maps provided by the Topografische Dienst in the city of Emmen. Large bookstores sell topographic maps, or carry catalogs from which you can order the maps in Emmen. The RD coordinate system is, as far as I know, a very common system used by most surveyors in the Netherlands. RD coordinates apply only to the Netherlands, and should not be extended beyond the largest bounding box, so to speak, excluding the Dutch colonies.

WGS84 is a completely different type. This is a global reference system that hinges on the use of a reference ellipsoid. GPS ephemeris and *xyz*'s are usually represented in this system. A conversion between the WGS84 latitude and longitudes to *xy* RD coordinates is what will be presented. The conversion itself is described exactly in a publication available from the Netherlands Geodetic Commission[1]. In a nutshell, the conversion works as follows:

1. The *x* and *y* RD coordinates are projected on a *Bessel ellipsoid*

2. The *latitude* and *longitude* on the Bessel ellipsoid are converted to WGS84

The Bessel ellipsoid is an invention of the RD, it is a so-called best fitting reference surface for the Netherlands. Latitudes and longitudes on Dutch topographic maps are (unless otherwise indicated) represented on the Bessel ellipsoid.

1. Thijsseweg 11, 2629 JA Delft, The Netherlands

The *x*-coordinate is called Easting (E) and the *y*-coordinate Northing (N) and both are given in meters. So the position of the centre of the small round island in the Hofvijver in The Hague (at the seat of the government) is noted as 081322E 455215N.

The Easting starts at about 012000 and rises nowhere higher than 276000. The Northing starts (in Zuid Limburg) with 306000 and ends up north with 615000. So, the Easting number (west-east direction) is always smaller than 300000 and Northing numbers are always bigger then 300000.

There are many ways to write down these coordinates. Usually (but not always) first the Easting then the Northing values. Sometimes they are given in kilometers like 81.4 / 455.2 but mostly in meters, yielding 6 (sometimes 5 when no leading zero is written, when the Easting <100000) digits for the Easting and 6 for the Northing.

3.2.6 Differential GPS

Differential GPS (DGPS) is a technique of reducing the error in GPS-derived positions by using additional data from a reference GPS receiver at a known position. The most common form of DGPS involves the determination of the combined effects of navigation message ephemeris and satellite clock errors (including propagation delays and the effects of SA) at a reference station and transmitting pseudorange corrections. This is done in real time to a user's receiver, which applies these corrections in its position-determining process.

DGPS can be used to minimize the errors of single-point GPS by canceling the parts of the error that are common to receivers in close proximity. Differential GPS is normally implemented by differencing the ranges to common satellites from two receivers. If the coordinates of one station are known, an accurate position of the second station can be determined. Alternatively, a coordinate difference between stations can be computed using approximate coordinates for one of the stations. Differential GPS reduces or eliminates errors caused by satellite clock and orbital errors, and atmospheric propagation. It does not reduce multipath errors, when the noise of a differenced observation is larger than that of an individual measurement by a factor of $\sqrt{2}$.

As we can see from both figures below, the DGPS navigation is comparable in accuracy with GPS alone and that is in the range of 5 meters RMS. The improvement is that the position information is more decorrelated; in Figure 3-8 we can observe that *x* and *y* position information looks correlated. This inefficiency of the DGPS correction is partially due to the distance to the station that sends RTCM correction messages, which in our case was approximately 150km.

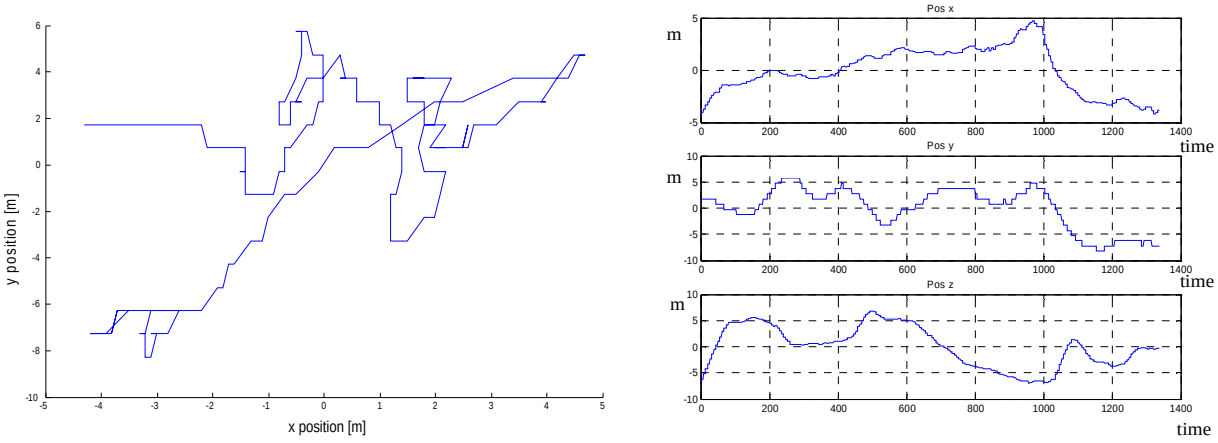

Figure 3-7. GPS navigation session with DGPS correction (available RDS data)

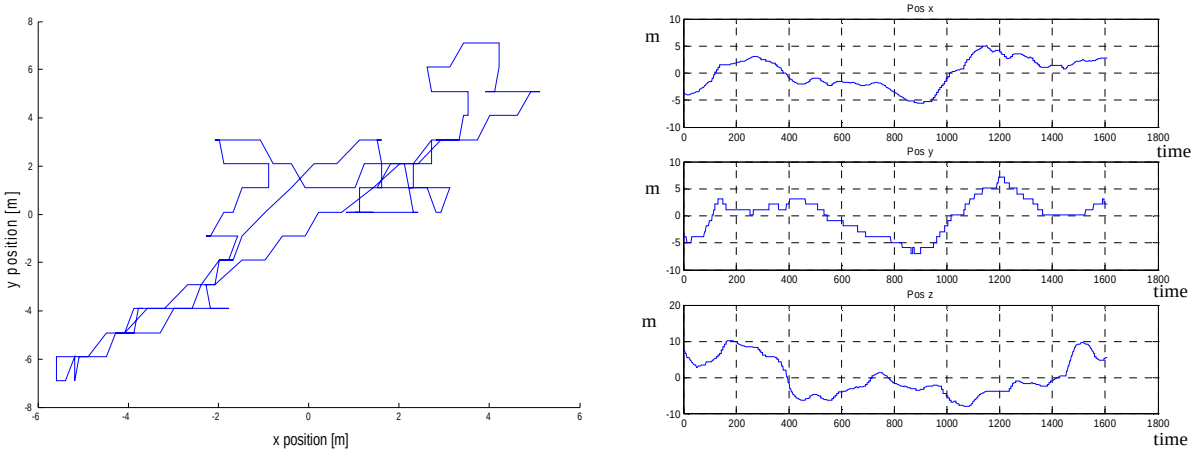

Figure 3-8. GPS navigation session without DGPS correction

3.3 Sensor Errors

In order to realize the navigation and alignment algorithms, the calibration of the sensor errors has to be done beforehand, stand alone. The sensor error model is:

$$
\begin{aligned}
\delta a_{bx} &= \alpha_x + \alpha_{xx}a_{bx} + \alpha_{xy}a_{by} + \alpha_{xz}a_{bz} \\
\delta a_{by} &= \alpha_y + \alpha_{yx}a_{bx} + \alpha_{yy}a_{by} + \alpha_{yz}a_{bz} \\
\delta a_{bz} &= \alpha_z + \alpha_{zx}a_{bx} + \alpha_{zy}a_{by} + \alpha_{zz}a_{bz}
\end{aligned}
\qquad \text{eq. 3-1.}
$$

$$
\begin{aligned}
\delta\omega_{bx} &= \beta_x + \beta_{xx}\omega_{bx} + \beta_{xy}\omega_{by} + \beta_{xz}\omega_{by} + \\
&(\beta_{xyx}a_{bx} + \beta_{xyy}a_{by} + \beta_{yzz}a_{bz})\omega_{by} + \\
&(\beta_{xzx}a_{bx} + \beta_{xzy}a_{by} + \beta_{xzz}a_{bz})\omega_{bz} \\
\delta\omega_{by} &= \beta_y + \beta_{yx}\omega_{bx} + \beta_{yy}\omega_{by} + \beta_{yz}\omega_{by} + \\
&(\beta_{yxx}a_{bx} + \beta_{yxy}a_{by} + \beta_{yxz}a_{bz})\omega_{bx} + \\
&(\beta_{yzx}a_{bx} + \beta_{yzy}a_{by} + \beta_{yzz}a_{bz})\omega_{bz} \\
\delta\omega_{bz} &= \beta_z + \beta_{zx}\omega_{bx} + \beta_{zy}\omega_{by} + \beta_{zz}\omega_{by} + \\
&(\beta_{zxx}a_{bx} + \beta_{zxy}a_{by} + \beta_{zxz}a_{bz})\omega_{bx} + \\
&(\beta_{zyx}a_{bx} + \beta_{zyy}a_{by} + \beta_{zyz}a_{bz})\omega_{by}
\end{aligned}
\qquad \text{eq. 3-2.}
$$

With:

$\delta a_{bi}, \delta\omega_{bi}, (i = x, y, z)$ - the accelerometer and gyro errors in projections on the body frame;

α_i - the accelerometer biases;

α_{ii} - the accelerometer scale factors;

α_{ij} - the accelerometer installation errors (i≠j);

β_i - the gyro biases;

β_{ii} - the gyro scale factor;

β_{ij} - the gyro installation errors (i≠j);

β_{ijk} - the gyro drift depending on the acceleration (flexure errors);

In our subsequent discussions we will not take the gyro flexure errors into consideration. The model parameters α_i and β_i are assumed to be constant, but with unknown values. The goal of the calibration is to estimate the values of the parameters above. A calibration procedure was realized by the installation of the inertial measurement unit onto a turning table, whereas the axes were oriented precisely with respect to a local-level frame. When rotating the IMU axes with respect to the local-level frame to different angles, a calibration measurement model can be generated. As the gyros are low cost and not capable of detecting the Earth's rotation rate we use for the calibration of the gyros a precise angle measure. For the accelerometer calibration we use the projection of the apparent gravity vector *g* on the body frame in different positions of the IMU.

3.3.1 Accelerometer calibration procedure

The accelerometers are calibrated by comparing the analog or digital signals produced by the sensors with a known applied motion. For example, from the rate transfer test the output signals from a gyroscope can be compared with an accurately known rotation rate and a scale factor, defined as *x* milli-volts per degree per second of rotation rate. Similarly, using the gravity vector as an accurate reference, the scale factor of an accelerometer can be defined. To observe the need for calibration, the measurements over 10 hours for three accelerometers are presented below.

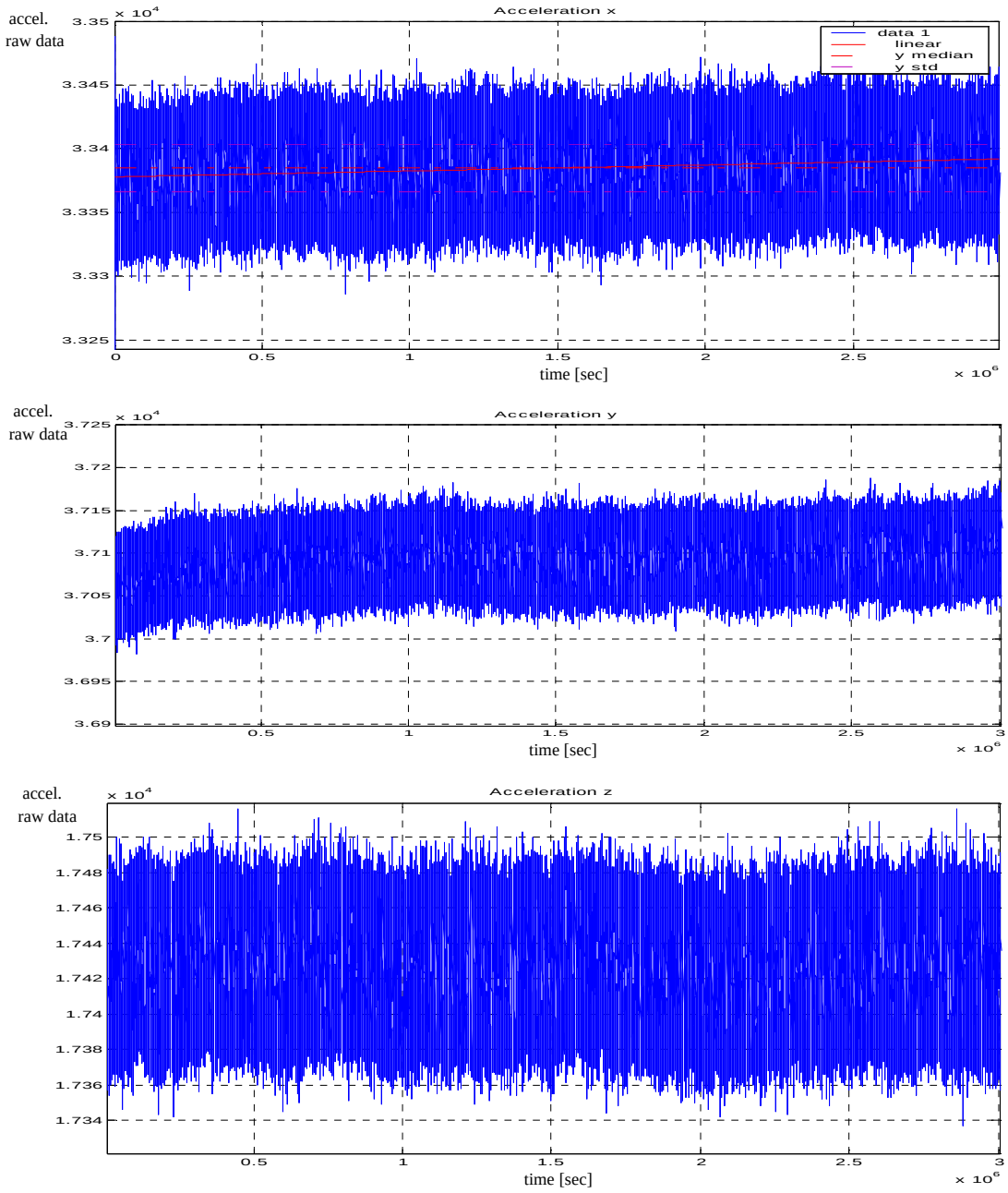

Figure 3-9. 10 Hours of accelerometer measurement

A long drift calibration of the acceleration is achieved by fitting a linear function to the data. A quadratic, cubic or an exponential fit did not yield good results. This experiment confirmed the error model that we assumed for the accelerometers: data plus slowly varying drift.

All sensors are sampled at 1 kHz. As we can see from Figure 3-9, all accelerometer signals are quite noisy. To reduce the noise we implement an elliptic fourth-order low pass filter with a cut-off frequency of 200 Hz. The filter was implemented in a Linux kernel driver, and all measurements were synchronized.

To calibrate the scale factor and bias of the accelerometer, we use the projection of an apparent gravity vector *g* on the body frame in different positions of the IMU. This calibration procedure has

the disadvantage that it requires a precise orientation of the IMU with respect to the local-level frame. If not, the orientation error will influence the calibration accuracy.

The thermal bias drift rate of the accelerometer placed at room temperature was found by experiments to be 0.108μg/s.

For each accelerometer:

- take the first measurement when the sensor is perfectly leveled (angle 0):

$$z^1(a_{bi}) = \alpha_i - \alpha_{ii}g - g \qquad \text{eq. 3-3.}$$

- the second measurement is taken after the sensor is rotated 180^o:

$$z^2(a_{bi}) = \alpha_i + \alpha_{ii}g + g \qquad \text{eq. 3-4.}$$

Using the above measurements, the estimates of the *x*, *y* and *z* scale factors and biases of the accelerometer can be computed as:

$$\alpha_i = \frac{z^1(a_{bi}) + z^2(a_{bi})}{2} \qquad \alpha_{ii} = \frac{z^2(a_{bi}) - z^1(a_{bi}) - 2g}{2g} \qquad \text{eq. 3-5.}$$

3.3.2 Gyroscope calibration procedure

Micromachined solid-state gyroscopes use vibrating mechanical elements to sense rotation. They have no rotating parts that require bearings, so they can be easily miniaturized. All vibration gyroscopes are based on the transfer of energy between two vibration modes of a mechanical structure, caused by Coriolis acceleration. The highest rotation sensitivity is obtained when the drive and sense modes have the same resonant frequency.

Resolution, drift rate, zero-rate output, and scale factor are the most important factors that determine the performance of a gyroscope. When a gyroscope is inertially static, the output signal is a random function that is the sum of white noise and a cyclic noise function of the mechanical resonant frequencies.

The Murata Gyrostar piezoelectric vibrating gyro was used for our IMU. This gyro was selected for its low cost, small size, and good performance relative to others in its class. The voltage output from the gyro is proportional to rotation rate, and is sampled at 1 kHz using the A/D converter of the LART system. In the actual implementation, the samples were filtered inside the kernel driver on the LART platform, and integrated at 100 Hz to give the absolute angles. The initial angles are set by the TCM2 sensor.

To develop an error model for the Murata gyroscopes, their outputs were recorded over long periods of time, subjected to zero input, i.e. the gyroscopes were fixed stationary on the laboratory bench. The result of this experiment over a period of 10 hours is shown in Figure 3-10.

Ideally, the output for zero input would be a constant voltage level corresponding to the digital output of 32768 for a 16-bit A/D converter. As we can see from Figure 3-10, the gyros have two types of errors. First of all, the zero level, or steady state, differs greatly between gyros from the same production lot. This is because we eliminated the output high pass filter from the electronic circuit that was suggested by Murata. The reason for this was that we observed a derivative effect on the output signal for low rotation rates, which was unacceptable for our application. This means, however, that we need first to calibrate the zero level. Also this obliged us to make a sensor selec-

tion in order to get those sensors with output values closest to ideal zero value, which is 32768. Another problem is the variation of this zero level with time.

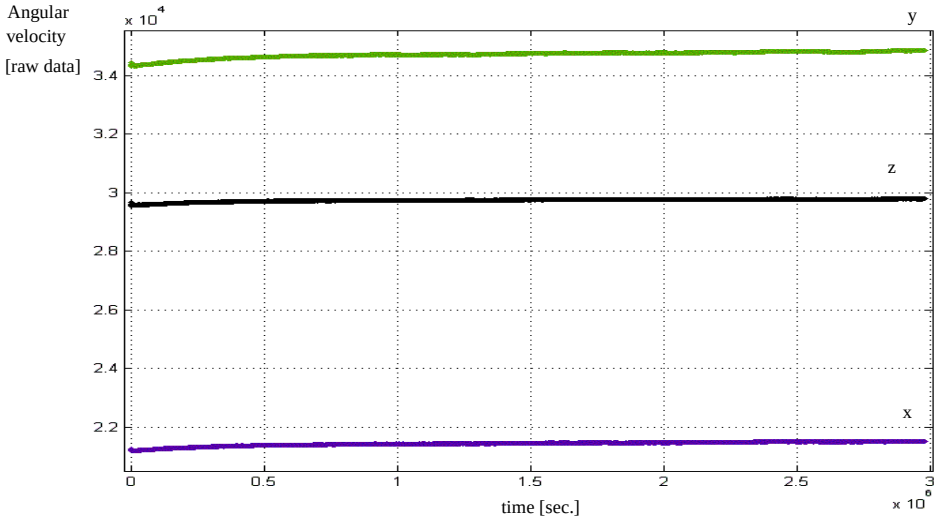

Figure 3-10. 10 hours of gyro measurement

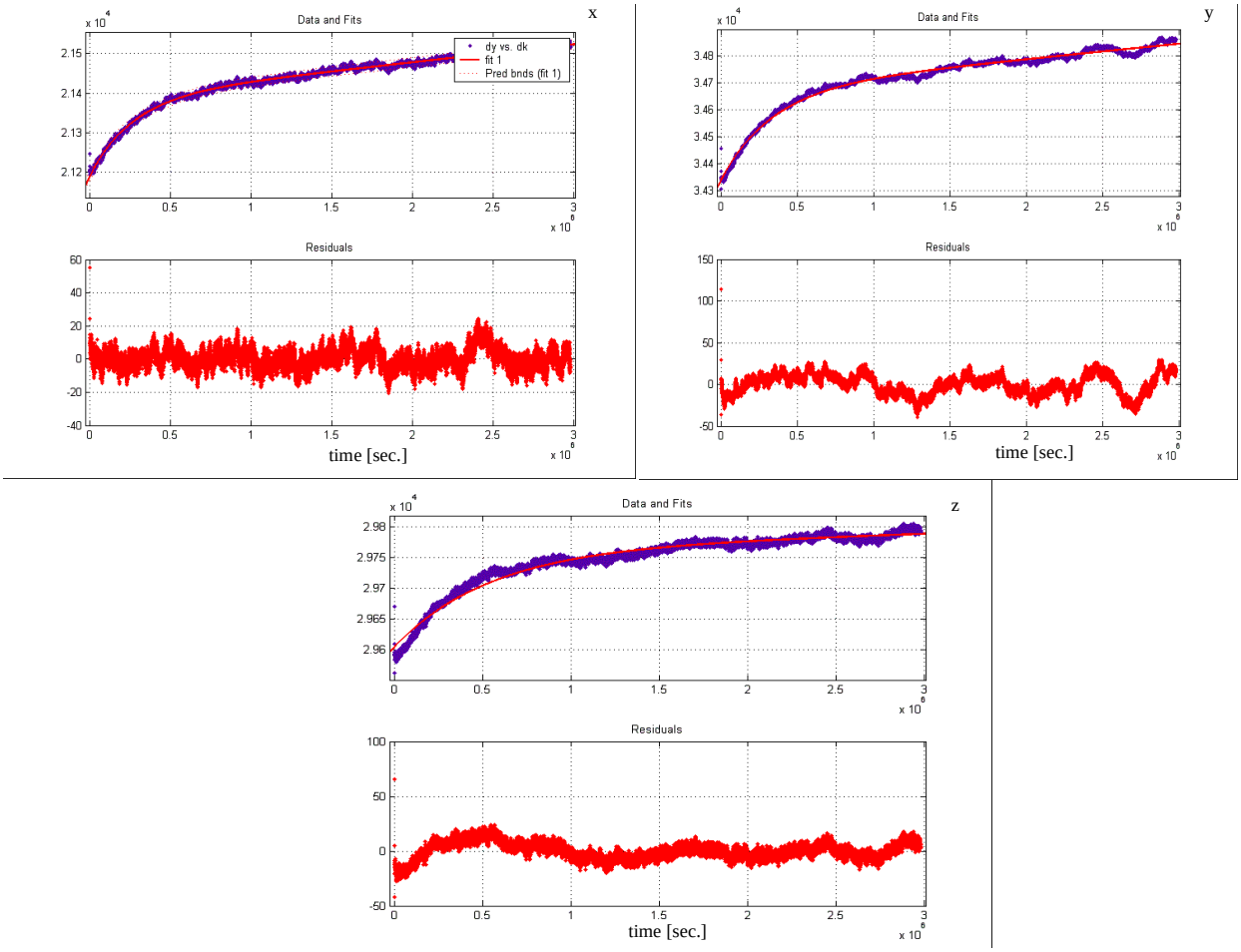

Figure 3-11. Long term gyro calibration

For a Murata gyro, the real output data is at a lower level than the ideal at start-up, and the mean gradually increases with time in an exponential way($(1 - e^{-t/b})$). The repeatability of these results

indicates that apparently a small time-varying bias is characteristic for these gyros. The time variation of the bias is attributed to thermal effects based on the observation that the gyroscope units gradually heat up during operation. The bias can taper off to a negative or positive value depending on the ambient temperature.

Long-duration gyro drift calibration is achieved by fitting an exponential function to the data. A nonlinear parametric model of the exponential form was fitted to the data from the gyroscope using the Levenberg-Marquardt iterative least-square fit method.

We tried two exponential error model functions, one with an exponential defined by three parameters, and the second containing two exponentials defined by four parameters.

$$\varepsilon_{model}(t) = a(1 - e^{-t/b}) + c \qquad \text{eq. 3-6.}$$

$$\varepsilon_{model}(t) = a(e^{bt}) + c(e^{dt}) \qquad \text{eq. 3-7.}$$

Figure 3-11 presents the signal for the three gyros over a 10-hours period, together with their fit function, as represented in Equation 3-7. For each function fit we present the residuals from the fit. In general, a model fitted to experimental data is regarded as being adequate if the residuals from the fitted model constitute a white, zero-mean process and with small σ. Hence, one can start with any reasonable model based on inspecting the original data and test its residuals for whiteness. If the test fails, the model can be further developed until the residuals pass the whiteness test. We can test the whiteness using the autocorrelation function. The autocorrelation function for a white process is well approximated by a narrow Gaussian distribution (theoretically it is a Dirac function). We present in Figure 3-12 the residuals of the fit and the spectrum of the residuals.

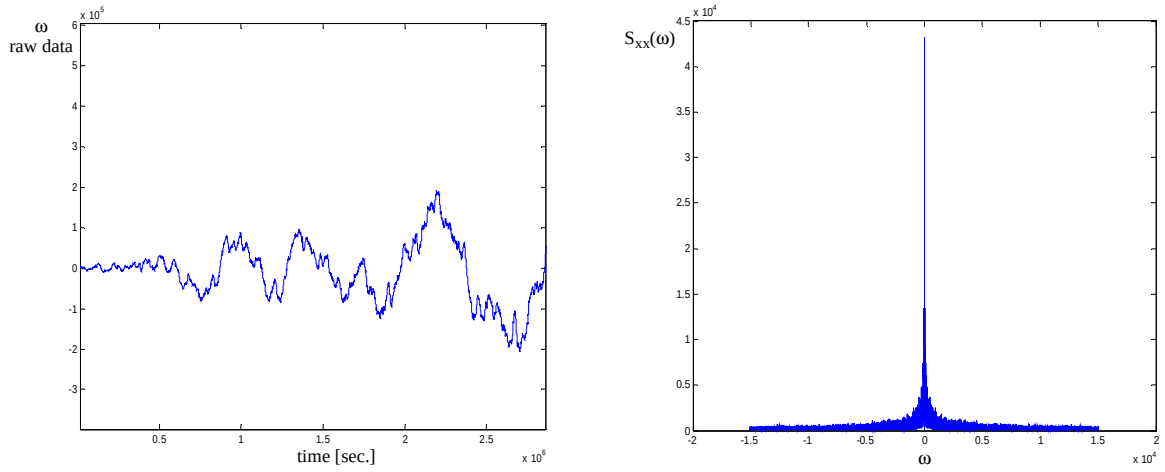

Figure 3-12. Residuals of the fit and spectrum of the residuals

For comparison purposes we present (in Figure 3-13) the two fitting functions overlayed over the original data. One can see that the second function offers a better approximation for the drift variation. The advantage of the first fit function is that one of the parameters is more meaningful.

The gyro has instability in both the zero-rotation output voltage (bias) and the scale factor, mainly as a result of temperature variations. The specifications for the zero-rotation bias and scale factor vary by 20%, so independent calibration of each unit is required. The high temperature variations require regular calibration of these values, however difficulties arise in trying to calibrate both the zero-rotation offset and scale factor simultaneously. The zero-rotation bias can be easily determined when the unit is not rotating, i.e. when the platform is stationary, but calibration of the scale factor requires rotation of the sensor with an accurate heading reference.

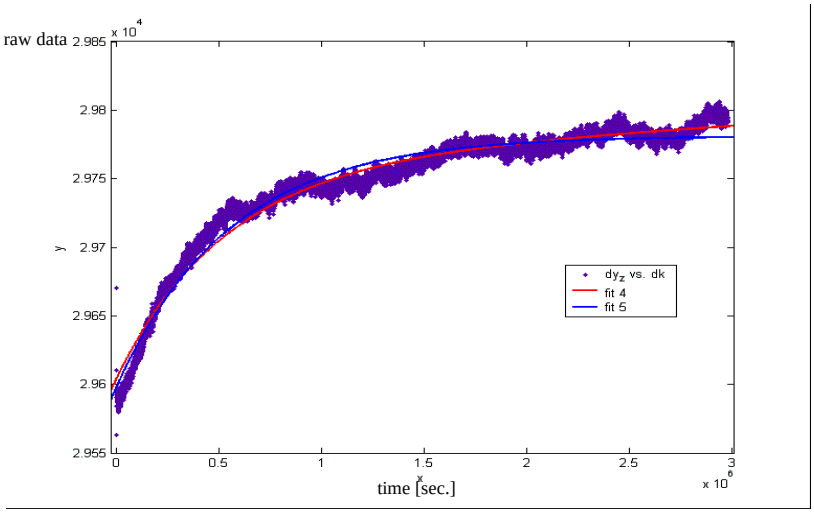

Figure 3-13. Comparison between the two fitting functions

General model Exp2:
f(x) = a*exp(b*x) + c*exp(d*x)

X gyro:

Coefficients (with 95% confidence bounds):
a = 2.139e+004 (2.139e+004, 2.139e+004)
b = 2.115e-009 (2.108e-009, 2.122e-009)
c = -196.3 (-196.8, -195.8)
d = -3.768e-006 (-3.787e-006, -3.748e-006)

Goodness of fit:
SSE: 1.091e+006
R-square: 0.9926
Adjusted R-square: 0.9926
RMSE: 6.021

Y gyro

Coefficients (with 95% confidence bounds):
a = 3.468e+004 (3.467e+004, 3.468e+004)
b = 1.638e-009 (1.628e-009, 1.649e-009)
c = -334.1 (-335.1, -333.2)
d = -2.994e-006 (-3.013e-006, -2.974e-006)

Goodness of fit:
SSE: 4.245e+006
R-square: 0.9884
Adjusted R-square: 0.9884
RMSE: 11.88

Z gyro

Coefficients (with 95% confidence bounds):
a = 2.976e+004 (2.976e+004, 2.976e+004)
b = 3.089e-010 (2.953e-010, 3.224e-010)
c = -156.7 (-157.6, -155.8)
d = -1.878e-006 (-1.9e-006, -1.857e-006)

Goodness of fit:
SSE: 1.739e+006
R-square: 0.9715
Adjusted R-square: 0.9715
RMSE: 7.601

The sensor is linear for low rotation rates. For high rotation rates the calibration is achieved by fitting a fifth order polynomial function to the measurement. In Figure 3-14 we present the error after the calibration (a nearly straight line), and the error without calibration (an S curve).

Note that a fiber optic gyro (FOG) has a range of stability of several degrees per hour. The big disadvantage of these type of sensors is their size and weight. In Figure 3-15 we present the signals of two gyros, a FOG gyro from DMU and a Murata gyro. We can see that the FOG gyro has a drift stability with an order of magnitude higher than the Murata gyro.

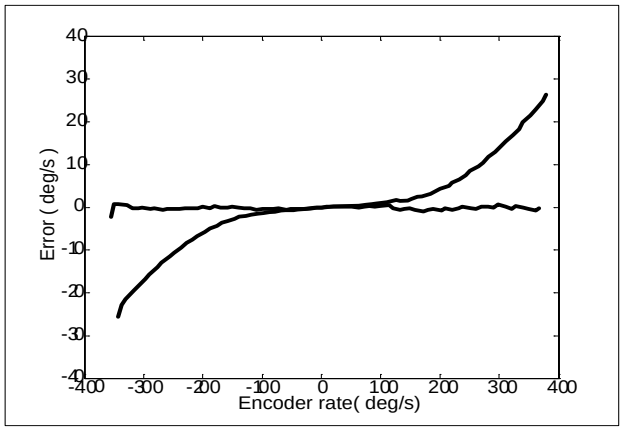

Figure 3-14. Scale factor calibration

.

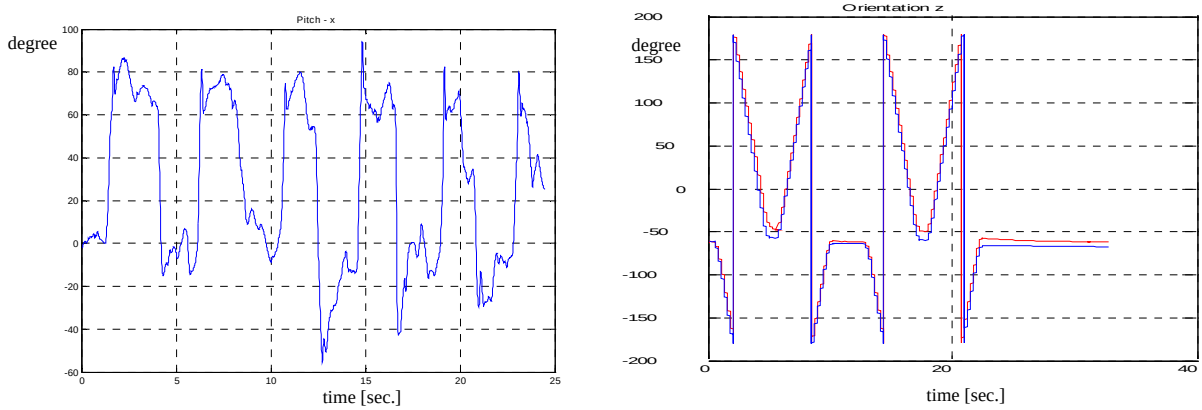

Figure 3-15. Fiber Optic Gyro stability

3.3.3 Overall Sensor Alignment

The alignment of the sensors on the IMU is done by leveling. In a strapdown inertial system this is a mathematical operation. The idea is to calculate the biases of the sensors. The scale factors are assumed to remain the same. The IMU is placed in a horizontal position and during leveling the IMU should remain stationary. If the IMU has tilt sensors the leveling equation can be corrected with their angles. From the measurements we concluded that misalignment and cross-axis sensitivity were negligible compared with drift and scale factor nonuniformity.

Accelerometers: We know the normal vectors of the accelerometers from calibration or from inclinometer measurements. The local gravity vector is projected onto the sensor axes. We can then calculate the theoretical readout from the sensor. The sensors are sampled over a period of time. The bias is the difference between the theoretical value and the calculated mean, and is calculated using:

$$b_{acc(j)} = SF^{-1}_{acc(j)} \left(n_{acc(j)} \cdot g - \frac{1}{N} \sum_{i=1}^{N} m_{acc(j)}(i) \right) \qquad \text{eq. 3-8.}$$

with: n_{acc} - the normal vector of accelerometer j, SF the accelerometer scale factor, and m_{acc} the accelerometer measurements.

Rate Gyros: When the IMU is stationary we know that its attitude remains the same. Therefore, the mean value of the sampled rotation rates should be zero.

$$b_{gyro(j)} = SF^{-1}_{gyro(j)}\left(0 - \frac{1}{N}\sum_{i=1}^{N} m_{gyro(j)}(i)\right) \qquad \text{eq. 3-9.}$$

In this way we can compute an estimate for the accelerometer bias and gyro drift, and this estimation can be use to initialize a Kalman filter.

3.4 Results and Conclusions

Conclusions on relative sensors

Relative (local) localization consists of evaluating the position and orientation through integration of encoders, accelerometers and gyros. The interration is started from initial position and orientation and is continuously updated in time. Though the technique is simple, it is prone to error due to impercision in modeling, noise, and drift. Substantial improvement is provided by applying Kalman filtering techniques, as shown in Chapter 4.

The Murata Gyrostar piezoelectric vibrating gyros can measure up to 300 °/s. They are inexpensive but have a large bias that varies with time up to 9 °/s. Consequently, we had to correct for this bias. After our correction, the noise level is around 0.2 °/s when sampled at 100Hz. The accelerometers (ADXL202) also have a varying offset. This offset can be 0.5 m/s^2 and the residual noise level is around 0.06 m/s^2 when sampled at 100Hz. The maximum acceleration that can be measured is 2g in both directions.

Conclusions on absolute sensors

Absolute (global) localization permits the mobile system to determine its position directly using navigation beacons, active or passive landmarks, map matching or satellite-based signals like GPS. Global sensor measurements can drastically increase the accuracy of the estimate and keep the associated uncertainty within certain bounds.

The TCM2-50 liquid inclinometer uses a viscous fluid to measure the inclination with respect to the gravity vector with an accuracy of 0.2°. The heading is calculated using three magnetometers with an accuracy of 0.5-1.5°. Because the liquid will slosh slightly when accelerations are applied, we have to cope with an error of about 20°.

Differential GPS reduces or eliminates errors caused by the satellite clock, orbital errors, and atmospheric propagation. It does not reduce multipath errors, when the noise of a differenced observation is larger than that of an individual measurement by a factor of $\sqrt{2}$.

In this chapter we saw that DGPS navigation is comparable in accuracy with GPS alone, i.e. in the range of 5 meters RMS. The improvement is that the position information is more decorrelated. This inefficiency of the DGPS correction is partially due to the distance to the station which sends RTCM correction messages, which in our case was approximately 150 km.

Conclusions on the making of the system

The performance requirements for the sensors are derived from the attitude determination and navigation accuracy requirements. The designer must determine that all of the important variables, including sensor drifts or biases, are observable from the chosen set of measurements. The sensor set should have bandwidth at least a decade above the movement dynamics. The correctly tuned estimation algorithm adjusts its bandwidth to take information from each sensor at the frequency

range where the sensor performance is best. The preliminary requirements on accuracy of sensors are derived from measurement equations. For example a 5m**g** accelerometer bias when used to correct low frequency gyro drifts will result in a low frequency error in estimation of local vertical of $.005(180\pi) \approx 0.3$degrees. In more complicated cases the design of the estimation algorithm has to be performed and covariance simulation run to determine the impact of errors.

Based on the analysis of the technologies presented in the previous chapter and the sensors described in this chapter, we selected a set of sensors from which to acquire and fuse the data in order to achieve the required robustness and accuracy. We selected for the inertial system three accelerometers (ADXL105) and three gyroscopes (Murata ENC05). To correct for gyro drift we use a TCM2 sensor that contains a two axis inclinometer and a three axes magnetometer (compass). Indoors we use a Firewire webcam to obtain the position and orientation information. Outdoors we use, in addition, a GPS receiver in combination with a radio data system (RDS) receiver to obtain DGPS correction information.

The LART platform, that is used for data acquisition and preprocessing, has an 8-channel fast 16-bit AD-converter to acquire synchronous data from the accelerometers, gyros and temperature data. The platform is running the Linux operating system and has very good real-time behavior. The gyros and the accelerometers are analog devices, which are sampled at 100 Hz by the AD converter. The TCM2 updates at 16 Hz and is read via a serial line. When the TCM2 and the gyros are read out simultaneously, there is an unknown difference in the time of the physical events. This difference was measured and the average was taken into account in the Kalman filter.

Conclusions on calibration

It is possible to build a look-up table based on the apparent relation between drift rate and sensor temperature. Doing so may provide a notable improvement in many applications. From an initial drift of 9 degree/second the drift dropped down to 5-10 degree/minute. For this reason, the LART platform comes with a digital A/D channel for a temperature sensor that can be read during operation. However, we found out that even with a drift look-up table the resulting bias-drift estimate is not accurate enough.

The thermal bias drift rate of the accelerometer placed at room temperature was found by experiments to be $0.108\mu g/s$. The effect of the random bias drift on the velocity and position error is quite important. The bias deviation is about 2-3mg for the entire accelerometer measurement range. Navigation grade accelerometers have about 0.1mg random bias. The velocity and position eror build-up can be computed with formulas:

Velocity error = 0.589m/s per m**g** per min

Position error = 17.66m per m**g** per min^2

Thus, for a bias of 2m**g**, the velocity error built up in one minute is 1.178m/s and position error is 35.2m. Thus, if the random bias can be modeled properly, the accuracy in distance measurement can be greatly improved.

Chapter 4

Inertial Navigation and Sensor Data Fusion

4.1 Introduction

In this chapter we will proceed with the design of a position and orientation tracking system for an Augmented Reality system. We will elaborate on coordinate frames and their transformations and introduce sensor data fusion tuned to our problem, using Kalman filtering. Specifically, we will elaborate on sensor data fusion using quaternions, as far as we know, a novel approach.

This chapter will provide the necessary background knowledge on inertial sensors and their associated errors. Furthermore, the chapter will provide, as a contribution, the inertial navigation equations needed for mobile systems. The inertial navigation equations are then linearized to develop error equations. These equations provide the mathematical foundation to analyze the propagation of errors in inertial navigation.

When designing the overall fusion algorithm, we must take into consideration the multi-rate sensor collection, and design our algorithm appropriately. That is, we need to fuse inertial and GPS data, when we are sampling our inertial sensors at 100 Hz and receiving GPS data at 1 Hz. Here, the term ‘fusion’ refers generally to the process of combining two sets of data to produce a better output. Sensor fusion can be accomplished with Kalman filtering, or it can be done by weighting each set of data based on a set of rules.

There are a number of remarkable advantages you will notice just in the form of the filter. First off all, the Kalman gain is computed from scratch each time you wish to incorporate a new measurement. This makes it very easy to track systems with time-varying dynamics or measurement processes. For adaptive filtering, one can adjust the measurement noise covariance, R_k, for each measurement to weight the measurement more or less heavily depending on the distance to the target, signal strength, or any other indicator of the probable quality of that measurement. This ability to handle time-varying models is also the key to using the Kalman filter with nonlinear systems or nonlinear measurement models. Linearizing about the current state estimate produces linear equations for the residual errors, with Φ_k (the state transition matrix) and H_k (output matrix) matrices which are functions of the current states. One then simply runs a time-varying standard Kalman filter on the error model, recomputing Φ_k and H_k at each step based on the current pose. This is the basis of the Extended Kalman Filter (EKF).

4.2 Coordinate Frames

A coordinate frame is an analytical abstraction defined by three consecutively numbered (or lettered) unit vectors that are mutually perpendicular to one another in the right-hand sense.

In the literature on inertial navigation, coordinate frames are defined in the following ways:

- The *E* frame is the Earth fixed coordinate frame used for position location definition. It is typically defined with one axis parallel to the Earth's polar axis and with the other axes fixed to the Earth and parallel to the equatorial plane.
- The *N* frame is the navigation coordinate frame, having its Z axis parallel to the upward vertical at a position location with respect to the local Earth surface. It is used for integrating acceleration into velocity and for defining the angular orientation of the local vertical in the E frame.
- The B frame is the strapdown inertial sensor coordinate frame (Body frame) with axes parallel to nominal right-handed orthogonal sensor input axes.
- The I frame is the non-rotating Inertial coordinate frame used as a reference for angular rotation measurements.

4.2.1 Strapdown Attitude Representations

4.2.1.1 The Euler angle representation

A transformation from one coordinate frame to another can be carried out as three successive rotations about different axes. This type of representation is popular because of the physical significance of the Euler angles which correspond to angles which would result by integrating the signal from three orthogonal gyros in a stable platform inertial navigation system.

The coordinate frames referred to in this thesis are orthogonal, right-handed axis sets in which positive rotations around each axis are taken to be in a clockwise direction looking along the axis from the origin, as can be seen in Figure 4-1. The rotation around the *X* axes is called *Roll*, the rotation around the *Y* axes is called *Pitch*, and the rotation around the *Z* axes is called *Yaw*. Roll, Pitch and Yaw are the names of angles in avionics books and applications. They are also referred to as *pan*, *tilt* and *heading*, e.g. in computer graphics and image processing applications

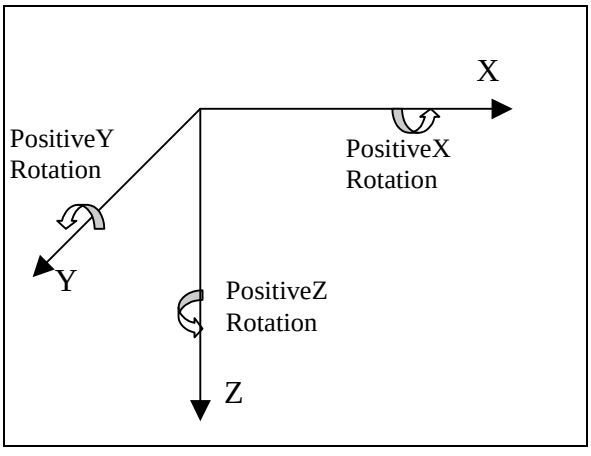

Figure 4-1. Definitions of rotation axis

4.2.1.2 Propagation of Euler angles in time

If angular rate gyros are rigidly mounted on a vehicle, they will directly measure the body-frame inertial angular rate vector (*p, q, r*) resolved along the vehicle axis. Hence it is necessary to transform (*p, q, r*) to ($\dot{\phi}, \dot{\theta}, \dot{\psi}$) [83]. This overall transformation is given by Equation 4-1:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} \frac{d\phi}{dt} \\ \frac{d\theta}{dt} \\ \frac{d\psi}{dt} \end{bmatrix} \qquad \text{eq. 4-1.}$$

The inverse transformation is:

$$\begin{bmatrix} \frac{d\phi}{dt} \\ \frac{d\theta}{dt} \\ \frac{d\psi}{dt} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \qquad \text{eq. 4-2.}$$

Equation 4-2 allows the calculation of the Euler angle derivatives for integration. Note that the solution requires integration of three first order nonlinear differential equations. This approach requires numerous trigonometric operations, and moreover, their use is limited as the solutions of $\dot{\phi}$ and $\dot{\psi}$ become undetermined when $\theta = \pm 90^{o}$.

4.2.1.3 The direction cosine matrix representation (DCM)

The direction cosine matrix denoted by C_b^n, is a 3 x 3 matrix, where the columns represent unit vectors in body axes, projected along the reference axes. The element in the *i*th row and the *j*th column represents the cosine of the angle between the *i* axis of the reference frame and *j* axis of the body frame. The transformation from body to reference axes is given by:

$$C_b^n = C_1^T C_2^T C_3^T = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} = \qquad \text{eq. 4-3.}$$

$$\begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\Psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}$$

The three rotations are expressed mathematically as three separate direction cosine matrices *C*1, *C*2 and *C*3, where ψ represents rotation around the *z* axis, θ represents rotation around the *y* axis, and ϕ represents rotation around the *x* axis.

$$C_1 = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad C_2 = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \qquad C_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \quad \text{eq. 4-4.}$$

Thus a transformation from reference to body frame may be expressed as the product of the separate transformations.

$$C_n^b = C_b^{nT} = C_3 C_2 C_1 \quad \text{eq. 4-5.}$$

For small angle rotations, where $\sin\phi \to \phi$, $\sin\theta \to \theta$, $\sin\psi \to \psi$ and the cosines of those angles approach unity and ignoring second order terms, DCM can be expressed in terms of Euler angles as a skew symmetric matrix.

$$C_b^n \approx \begin{bmatrix} 1 & -\psi & \theta \\ \psi & 1 & -\phi \\ -\theta & \phi & 1 \end{bmatrix} \quad \text{eq. 4-6.}$$

This form of matrix is used sometimes to represent the small changes in attitude which occur between successive updates in real-time computation of body attitude, and to represent the error in the estimation of a direction cosine matrix.

4.2.1.4 Propagation of the direction cosine matrix in time

The rate of change of C_b^n with time [83] is given by:

$$\frac{d}{dt}C_b^n = C_b^n\Omega \qquad \Omega = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \quad \text{eq. 4-7.}$$

The values *p*, *q*, and *r* represent the angular rate around each axis of the B-frame with respect to the N-frame. An observation *p*, *q*, *r* is noted in some papers and books as ω_x, ω_y, and ω_z.

An equation in the form of Equation 4-7 needs to be solved by the computer in a strapdown inertial navigation system to keep track of the body attitude with respect to a chosen reference frame. It can be expressed in close form as:

$$C_b^n(t_{k+1}) = C_b^n(t_k)\left(\mathbf{I} + \frac{\sin\sigma}{\sigma}A + \frac{1-\cos\sigma}{\sigma^2}A^2\right) \quad \text{eq. 4-8.}$$

with:

$$A = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}\Delta T \qquad \sigma = \sqrt{p^2+q^2+r^2}\,\Delta T$$

$$A^2 = \begin{bmatrix} -(q^2+r^2) & pq & pr \\ pq & -(p^2+r^2) & qr \\ pr & qr & -(p^2+q^2) \end{bmatrix}\Delta T^2 \quad \text{eq. 4-9.}$$

The derivation of Equation 4-8 is presented in section Section A.2 on page 149.

4.2.1.5 The quaternion representation

The quaternion attitude representation is a four parameter representation based on the idea that a transformation from one coordinate frame to another may be effected by a single rotation about vector μ defined with respect to a reference frame. The quaternion, denoted here by **q**, is a four-element vector, the elements of which are functions of this vector and the magnitude of the rotation.

$$\mathbf{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) \\ (\mu_x/\mu)\sin(\theta/2) \\ (\mu_y/\mu)\sin(\theta/2) \\ (\mu_z/\mu)\sin(\theta/2) \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) \\ \alpha\sin(\theta/2) \\ \beta\sin(\theta/2) \\ \gamma\sin(\theta/2) \end{bmatrix} \qquad \text{eq. 4-10.}$$

The meaning of α, β, and γ is as follows: the quaternion *q*, makes the angle $\cos^{-1}\alpha$, $\cos^{-1}\beta$ and $\cos^{-1}\gamma$ with the inertial axes *x*, *y*, and *z*. In figure Figure 4-2 this is presented as a vector *u* that is rotated around vector *q* with, as result, vector *v*, and μ is a unit vector.

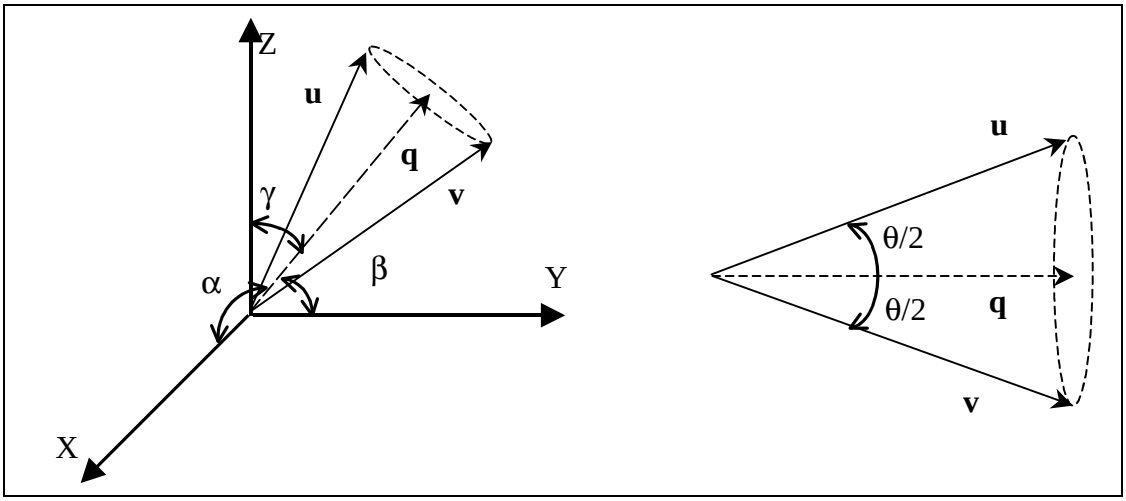

Figure 4-2. Representation of rotation using quaternion

It can be shown (see Section A.3) that the quaternion vector elements can be used to compute the direction cosine matrix C_b^n.

$$C_b^n = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \qquad \text{eq. 4-11.}$$

4.2.1.6 The quaternion norm

The norm of the quaternion vector should always be one and it is used to correct for accumulated computational errors.

$$\|\mathbf{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \qquad \text{eq. 4-12.}$$

If we look at Equation A-18 on page 153 we see that Euler angles are computed using all elements of the quaternion vector. This can introduce some unwanted effects. If one of the quaternion elements has a small error, this will affect all the other elements when the normalization is done.

4.2.1.7 The propagation of the quaternion in time

The quaternion, **q**, propagates in accordance with the following equation:

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega_q \cdot \mathbf{q} \qquad \Omega_q = \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \qquad \text{eq. 4-13.}$$

Quantities *p*, *q*, and *r* represent the angular rate about each axis of the B-frame with respect to the N-frame. For the situation in which the orientation of the rate vector remains fixed over an interval, the solution to the above equation may be written as:

$$\mathbf{q}(t_{k+1}) = \begin{bmatrix} a_c & -a_s\sigma_x & -a_s\sigma_y & -a_s\sigma_z \\ a_s\sigma_x & a_c & a_s\sigma_z & -a_s\sigma_y \\ a_s\sigma_y & -a_s\sigma_z & a_c & a_s\sigma_x \\ a_s\sigma_z & a_s\sigma_y & -a_s\sigma_x & a_c \end{bmatrix} \cdot \mathbf{q}(t_k) \qquad \begin{cases} a_c = \cos\left(\frac{\sigma}{2}\right) \\ a_s = \frac{\sin\left(\frac{\sigma}{2}\right)}{\sigma} \end{cases} \qquad \text{eq. 4-14.}$$

with:

$$\sigma_x = \int_{t_k}^{t_{k+1}} p\,dt \qquad \sigma_y = \int_{t_k}^{t_{k+1}} q\,dt \qquad \sigma_y = \int_{t_k}^{t_{k+1}} q\,dt \qquad \sigma = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} \qquad \text{eq. 4-15.}$$

The derivation of Equation 4-14 is presented in section Section A.3 on page 151.

4.3 Inertial Navigation

4.3.1 Navigation Frame Mechanization

In order to navigate over large distances around the Earth, e.g., in a vehicle, navigation information is most commonly required in the local geographic or navigation axis in terms of north and east velocity components, latitude, longitude and height above the Earth. In this mechanization, ground speed is expressed in navigation coordinates to give v_e^n. The rate of change of v_e^n with respect to navigation axes may be expressed in terms of its rate of change in inertial axes as follows:

$$\left.\frac{dv_e}{dt}\right|_n = \left.\frac{dv_e}{dt}\right|_i - \omega_{ie} \times v_e - \omega_{en} \times v_e \qquad \text{eq. 4-16.}$$

The speed rate of change in inertial axes is:

$$\left.\frac{dv_e}{dt}\right|_i = a - \omega_{ie} \times v_e + g \qquad \text{eq. 4-17.}$$

where *a* represents the specific force acceleration to which the navigation system is subjected.

Substituting Equation 4-17 in Equation 4-16 we have:

$$\left.\frac{dv_e}{dt}\right|_n = a - 2\omega_{ie} \times v_e - \omega_{en} \times v_e + g \qquad \text{eq. 4-18.}$$

This equation may be expressed in navigation axes as follows:

$$v_e^n = C_b^n a^b - 2\omega_{ie}^n \times v_e^n - \omega_{en}^n \times v_e^n + g^n \qquad \text{eq. 4-19.}$$

where C_b^n is the direction cosine matrix used to transform the measured specific force vector into navigation frame (*n* in the previous equation is not an exponential notation but represents the navigation frame).

It is necessary to consider the physical significance of the various terms in the navigation Equation 4-19. From this equation, it can be seen that the rate of change of the velocity, with respect to the surface of the Earth, is made up of the following terms:

- The specific force acting on the vehicle, as measured by a triad of accelerometers mounted within it;
- A correction for the acceleration caused by the vehicle's velocity over the surface of a rotating Earth, usually referred to as the Coriolis acceleration.
- A correction for centripetal acceleration of the vehicle, resulting from its motion over the Earth's surface. The force is equal to the product of its mass, its linear velocity and its turn rate with respect to the Earth.
- Compensation for apparent gravitational force acting on the vehicle. This includes the gravitational force caused by the mass attraction of the Earth, and the centripetal acceleration of the vehicle resulting from the rotation of the Earth. The latter term arises even if the vehicle is stationary with respect to the Earth.

4.3.2 Navigation Equations in Body Frame

We have $[\dot{u}, \dot{v}, \dot{w}]$ as the true vehicle acceleration in the body frame. This acceleration is not the acceleration measured by the IMU (Inertial Measurement Unit). The IMU accelerometer sensors will measure the gravity vector, the vehicle acceleration, the centripetal acceleration and some noise. The radius of curvature of the path is *r*. The acceleration due to angular velocity is:

$$a_{cf} = \frac{v^2}{r} = \frac{\omega r \omega r}{r} = \omega v \qquad \text{eq. 4-20.}$$

In vector form this will be:

$$a^b_{cf} = \omega \times v^b = \omega \times (\omega \times r) \qquad \text{eq. 4-21.}$$

In the figure below we illustrate the forces acting on the moving body.

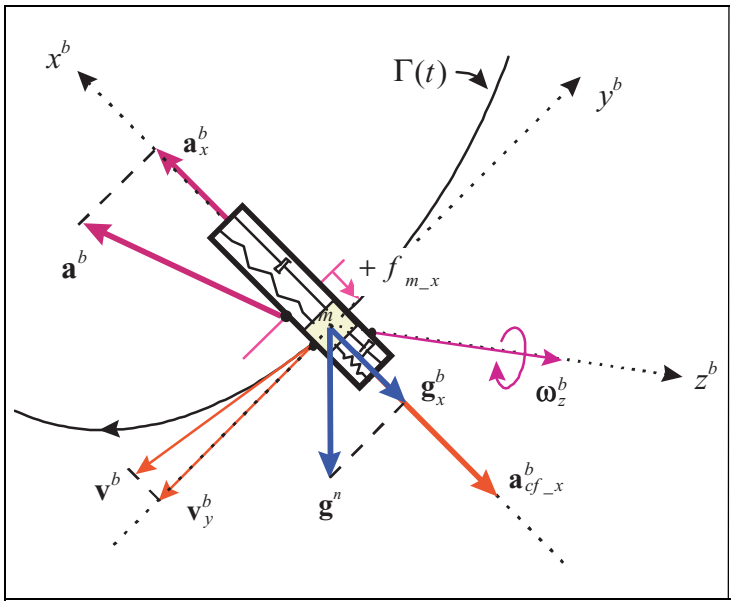

Figure 4-3. Specific force as a function of acceleration components

Figure 4-3 explains the forces acting upon the mass of the accelerometer as a function of the linear acceleration, the apparent centripetal acceleration and the corresponding axial component of the static gravitational acceleration. The superscripts *b* denote the vector components in the *body* reference system. The vectorial form for the navigation equation, with the specific force acceleration being *a* and the correction terms of centripetal and gravity accelerations expressed in the body coordinate system is:

$$a^b = a - \omega \times v^b + C^b_n \cdot g^n \qquad \text{eq. 4-22.}$$

Here we ignore the Coriolis acceleration. If the navigation system is meant to work over long distances we need to take into account the Coriolis acceleration caused by the rotation of the Earth. The Coriolis acceleration will be non-zero if the vehicle is traveling such that the latitude is changing.

The IMU will also measure the gravity acceleration and can not know the difference between this acceleration and true vehicle acceleration. To solve the vehicle acceleration we must subtract the known gravity components from the measured accelerations. This gives us Equation 4-22, where

$[a_x, a_y, a_z]$ are the measured acceleration in body frame and $[g_x, g_y, g_z]$ are the gravity components expressed in body frame. The gravity vector in the body frame is not dependent on the yaw or heading of the vehicle.

$$\begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = C_n^b \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{bmatrix} \qquad \text{eq. 4-23.}$$

We get the θ and ϕ angle from the IMU and can calculate the gravity components in the body frame. The rotation matrix from the local coordinate system *n* to the body coordinate system *b* is noted with C_n^b.

If the IMU is not placed in the center of gravity we have another term. Suppose that the IMU is placed at coordinate $[r_x, r_y, r_z]_{BODY}$ relative to the center of gravity, then there will be a centripetal acceleration when the vehicle turns. This acceleration can be noted as a_{IMU} and the magnitude is dependent on the rotation rates and the placement of the IMU. This acceleration can be computed with the following equation:

$$\begin{bmatrix} a_{IMUx} \\ a_{IMUy} \\ a_{IMUz} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \left(\begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \right) \qquad \text{eq. 4-24.}$$

The equation that takes all the influences into account is:

$$\begin{aligned} \dot{u} - vr + wq + g_x + a_{IMUx} &= a_x \\ \dot{v} + ur - wp + g_y + a_{IMUy} &= a_y \\ \dot{w} - uq + vp + g_z + a_{IMUz} &= a_z \end{aligned} \qquad \text{eq. 4-25.}$$

We can solve the true vehicle acceleration from Equation 4-25. We assume that the IMU is placed at the center of gravity and hence $a_{IMU} = 0$:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} 0 & -w & v \\ w & 0 & -u \\ -v & u & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} - \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} \qquad \text{eq. 4-26.}$$

The flow-chart [71] of the strapdown navigation algorithm implementing the equation presented above is presented in Figure 4-4.

In order to verify the model, we simulated accelerometer and gyro signals and we computed the navigation solution as in Figure 4-4. The sensors are considered to be perfect, without noise or drift. The result of this simulation is presented in Figure 4-5.

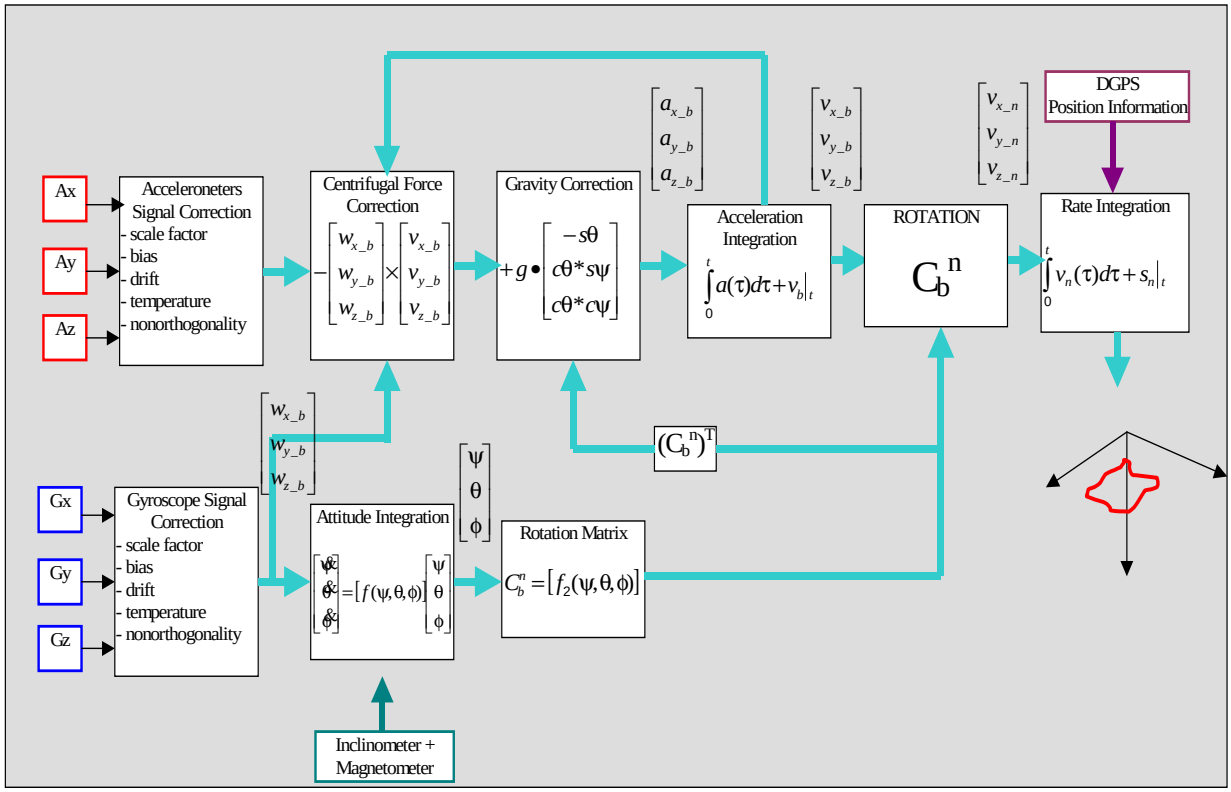

Figure 4-4. Flow-chart of the strapdown mechanization

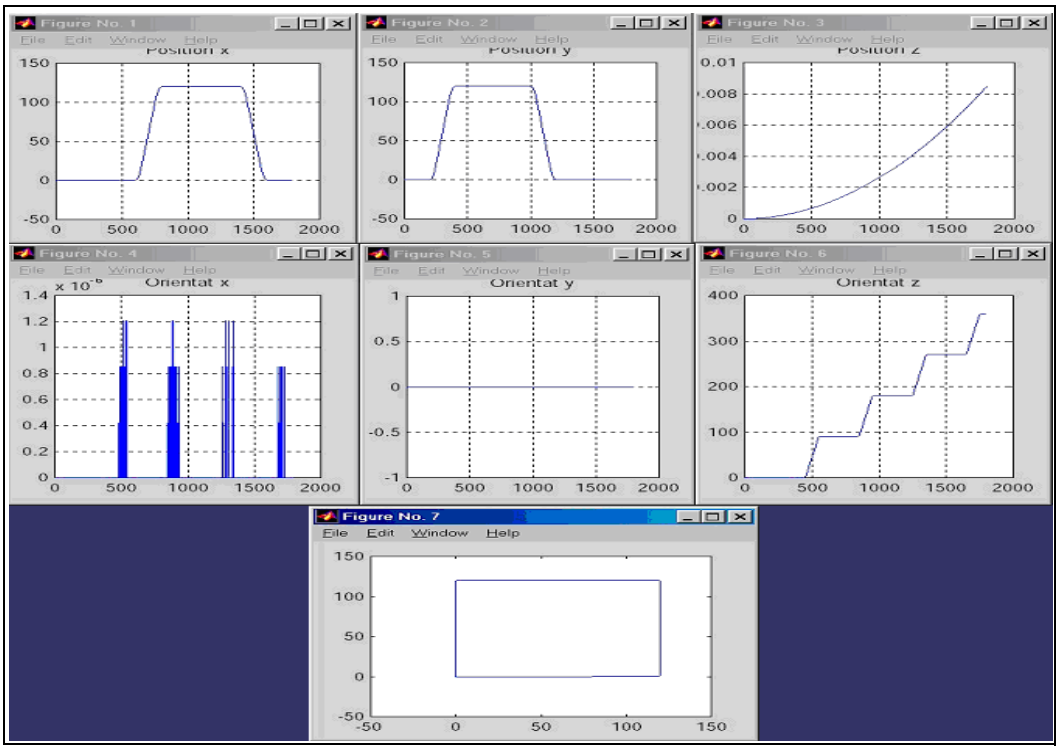

Figure 4-5. Navigation Simulation Results

A conclusion that can be drawn looking at navigation equations is related to accelerometer error and gyro drift. Drift in the linear position determined by an INS arises from several sources. First,

there are accelerometer instrument errors like bias, white noise, calibration errors and bias instability. Since position is obtained by double integrating acceleration, a fixed accelerometer bias error results in a position drift error that grows quadratically in time. It is therefore critical to accurately estimate and eliminate any bias errors. A much more critical cause of error in position measurement is error in the orientation determined by the gyros. An error of $\delta\theta$ in tilt angle will result in an error of $1g \cdot \sin(\delta\theta)$ in the horizontal components of the acceleration calculated by the navigation computer. Thus, to take proper advantage of µg-accurate accelerometers, the pitch and roll accuracy must be better than 1 µrad = 0.000057° for the duration of the navigation session, which is a far more difficult task for gyros than accelerometers. In practice, it is the gyroscopes, not the accelerometers which limit the positional navigation accuracy of most INS systems, since the effects of gyroscopic tilt error will soon overtake any small accelerometer biases.

4.4 Sensor Data Fusion with Kalman Filters

4.4.1 The Kalman Filter

Kalman filtering is the main analysis technique for inertial data and is used almost exclusively for inertial tracking, i.e. the determination of position and orientation from inertial readings and an initial state. As inertial components have become more common in user interfaces, particularly for head-tracking, a number of variations on the original algorithms have been created to ensure robustness and accuracy when working with low-cost components and processing hardware.

Kalman filtering (KF) is a state-based recursive algorithm, which works in two stages. The first is the prediction stage where, given the (possibly incorrect) current state of the system and a mapping of how the system progresses in time, a prediction of the state at the next time step is calculated. The second is the correction stage where, given a noisy observation (not necessarily full or direct) of the state at the new time step, the predicted state and the measured values are combined to give a new, more accurate estimate of the state of the system. Given certain assumptions described below, Kalman filtering is a least-squares optimal linear method of estimating a system's state.

The strength of Kalman filtering lies in its optimality, its minimal memory requirements for the state (only a single time step), and its ability to include a model of the system dynamics. There are a number of limitations which must be considered as well. The most troublesome is that it is the linear optimal solution, while most physical systems, including tracking, are non-linear. In such circumstances the Extended Kalman Filter (EKF) can be used, which, although it can no longer be proven to be optimal (or even to converge), it is still very successful in practical applications [92]. The next problem (for either type of filter) lies in the process noise models, which are assumed to be white and Gaussian.

4.4.1.1 The discrete Kalman filter

The Kalman filter addresses the general problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by a linear stochastic difference equation and a measurement, given by:

$$\begin{aligned} x_{k+1} &= \Phi x_k + \Gamma u_k + w_k \\ y_k &= H x_k + v_k \end{aligned} \qquad \text{eq. 4-27.}$$

The random variables w_k and v_k represent the process and measurement noise, respectively. They are assumed to be independent of each other, white, and with normal probability distributions. In

practice, the process noise covariance Q and measurement noise covariance R matrices might change with each time step or measurement, in this case, however, we assume they are constant.

The $n \times n$ matrix Φ in the difference equation in Equation 4-27 relates the state at the previous time step to the state at the current step, in the absence of either a driving function or process noise. Note that in practice it might change with each time step, but here we assume it is constant. The $n \times l$ matrix Γ relates the optional control input to the state x. The $m \times n$ matrix H in the measurement equation relates the state to the measurement z_k. In practice, it might change with each time step or measurement, but here we assume it is constant.

We define $\hat{x}^- \in \Re^n$ to be an *a priori* state estimate at step k, given knowledge of the process prior to step k, and $\hat{x} \in \Re^n$ to be an *a posteriori* state estimate at step k, given measurement z_k. We can define a priori and a posteriori estimate errors as:

$$\begin{aligned} e_k^- &\equiv x_k - \hat{x}_k^- \\ e_k &\equiv x_k - \hat{x}_k \end{aligned} \qquad \text{eq. 4-28.}$$

The a priori and a posteriori estimate error covariance is then:

$$\begin{aligned} P_k^- &= E\langle e_k^-, e_k^{-T} \rangle \\ P_k &= E\langle e_k, e_k^T \rangle \end{aligned} \qquad \text{eq. 4-29.}$$

In deriving the equations for the Kalman filter, we begin with the goal of finding an equation that computes an *a posteriori* state estimate $\hat{x}_k$ as a linear combination of an *a priori* estimate $\hat{x}_k^-$ and a weighted difference between an actual measurement z_k and a measurement prediction $H\hat{x}_k^-$ as shown below in Equation 4-30.

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \qquad \text{eq. 4-30.}$$

The difference $(z_k - H\hat{x}_k^-)$ in Equation 4-30 is called the measurement *innovation*, or the *residual*. The residual reflects the discrepancy between the predicted measurement and the actual measurement. A residual of zero means that the two are in complete agreement.

We can write the a posteriori estimate (updated) covariance as:

$$\begin{aligned} P_k &= E\langle [(x_k - \hat{x}_k) - K(Hx_k + v_k - H\hat{x}_k^-)], [(x_k - \hat{x}_k) - K(Hx_k + v_k - H\hat{x}_k^-)]^T \rangle \\ &= (I - KH)P_k^-(I - KH)^T + KRK^T \end{aligned} \qquad \text{eq. 4-31.}$$

The $n \times m$ matrix K in Equation 4-30 is chosen to be the *gain* or *blending factor* that minimizes the a posteriori error covariance. This minimization can be accomplished by performing the indicated expectations, taking the derivative of the trace of the P_k with respect to K (see Equation 4-32), setting that result equal to zero, and then solving it for K.

$$\begin{aligned} \frac{\partial}{\partial K}(\text{trace}P_k) &= -2(HP_k^-)^T + 2K(HP_k^-H^T + R) \\ \frac{\partial}{\partial A}(\text{trace}AB) &= B^T \qquad \frac{\partial}{\partial A}(\text{trace}ACA^T) = 2AC \end{aligned} \qquad \text{eq. 4-32.}$$

One form of the resulting K that minimizes Equation 4-31 is given by:

$$K = P_k^-H(HP_k^-H^T + R)^{-1} \qquad \text{eq. 4-33.}$$

Looking at Equation 4-33 we see that as the measurement error covariance *R* approaches zero, the gain *K* weights the residual more heavily. Specifically,

$$\lim_{R \to 0} K = H^{-1} \qquad \text{eq. 4-34.}$$

On the other hand, as the a priori estimate error covariance P_k^- approaches zero, the gain *K* weights the residual less heavily. Specifically,

$$\lim_{P_k^- \to 0} K = 0 \qquad \text{eq. 4-35.}$$

Another way of thinking about the weighting by *K* is that as the measurement error covariance *R* approaches zero, the actual measurement z_k is trusted more and more, while the predicted measurement $H\hat{x}_k^-$ is trusted less and less. On the other hand, as the a priori estimate error covariance P_k^- approaches zero the actual measurement z_k is trusted less and less, while the predicted measurement $H\hat{x}_k^-$ is trusted more and more.

Another form for Equation 4-31 to compute the a posteriori covariance matrix is obtained by substituting in the above mentioned equation for *K*:

$$P_k = (I - KH)P_k^- \qquad \text{eq. 4-36.}$$

The Equation 4-31 is sometime called the Joseph form and this helps to overcome ill-conditioning. Note that the right-hand side of Equation 4-31 is the summation of two symmetric matrices. The first one is positive definite and second is nonnegative definite.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: *time update equations* and *measurement update equations*. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback, for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate. The time update equations can also be thought of as *predictor equations*, while the measurement update equations can be thought of as *corrector equations*.

The equations derived in this section are summarized here. The basic steps of the computational procedure for a discrete Kalman filter are as follows:

- **Discrete Kalman filter time update equations**

$$\begin{aligned} \hat{x}_k^- &= \Phi\hat{x}_{k-1} + \Gamma u_k \quad \text{- state estimation extrapolation} \\ P_k^- &= \Phi P_{k-1}\Phi^T + Q \quad \text{- error covariance extrapolation} \end{aligned} \qquad \text{eq. 4-37.}$$

The time update equations project the state and the covariance estimates forward from step *k*-1 to step *k*.

- **Discrete Kalman filter measurement update equations**

$$
\begin{aligned}
K &= P_k^- H(HP_k^- H^T + R)^{-1} && \text{- Kalman gain matrix} \\
\hat{x}_k &= \hat{x}_k^- + K(z_k - H\hat{x}_k^-) && \text{- state observation update} \\
P_k &= (I - KH)P_k^- && \text{- error covariance update}
\end{aligned}
\qquad \text{eq. 4-38.}
$$

The first task in the measurement update stage is to compute the Kalman gain *K*. The next step is to actually measure the process to obtain z_k, and then to generate an a posteriori state estimate by incorporating the measurement. The final step is to obtain an a posteriori error covariance estimate.

4.4.1.2 The discrete extended Kalman filter

A Kalman filter that linearizes around the current mean and covariance is referred to as an extended Kalman filter or EKF. To estimate a process with non-linear difference and measurement relationship, we begin by writing the equations that linearize an estimate around Equation A-62 on page 161:

$$
\begin{aligned}
\dot{x}(t) &= f(x(t), u(t), t) + w(t) \\
y(t) &= h(x(t), t) + v(t) \\
x_k &\approx \tilde{x}_k + \Phi(x_{k-1} - \hat{x}_{k-1}) + \Gamma w_{k-1} \\
z_k &\approx \tilde{z}_k + H(x_k - \tilde{x}_k) + Vv_k
\end{aligned}
\qquad \text{eq. 4-39.}
$$

with:

- x_k and z_k are the actual state and measurement vectors
- $\tilde{x}_k = f(\hat{x}_{k-1}, u_k)$ and $\tilde{z}_k = h(\tilde{x}_k)$ are the approximate state and measurement vector, and $\hat{x}_{k-1}$ is an a posteriori estimate of the state (from a previous time step *k*)
- the random variables w_k and v_k represent the process and measurement noise
- *F* is the Jacobian matrix of partial derivatives of *f* with respect to *x*, and *W* is the Jacobian matrix of partial derivatives of *f* with respect to *w* (ψ was defined in Equation A-51 on page 159):

$$
\begin{aligned}
F &= \frac{\partial}{\partial x} f(\hat{x}_{k-1}, u_k) \qquad W = \frac{\partial}{\partial w} f(\hat{x}_{k-1}, u_k) \\
\Phi &= I + F\psi \qquad \Gamma = \psi G
\end{aligned}
\qquad \text{eq. 4-40.}
$$

- *H* is the Jacobian matrix of partial derivatives of *h* with respect to *x*, and *V* is the Jacobian matrix of partial derivatives of *h* with respect to *v*:

$$
H = \frac{\partial}{\partial x} h(\tilde{x}_k) \qquad V = \frac{\partial}{\partial v} h(\tilde{x}_k) \qquad \text{eq. 4-41.}
$$

The fundamental matrix, required for the discrete Riccati equations, can be approximated by the Taylor-series expansion for exp(FT) and is given in Equation 4-40. Often the series is approximated by only the first two terms. In our applications of extended Kalman filtering, the fundamental matrix will only be used in the calculation of Kalman gains. Because the fundamental matrix will not be used in the propagation of the states, it was demonstrated in Chapter 5 from Zarchan [92] that adding more terms to the Taylor-series expansion for the fundamental matrix will not generally improve the performance of the overall filter.

The basic steps of the computational procedure for discrete extended Kalman filter are as follows:

- **Discrete extended Kalman filter time update equations**

$$\begin{aligned} \hat{x}_k^- &= \int f(\hat{x}_{k-1}, u_k) && \text{- state estimation extrapolation} \\ P_k^- &= \Phi P_{k-1} \Phi^T + Q && \text{- error covariance extrapolation} \end{aligned} \qquad \text{eq. 4-42.}$$

The old estimates that have to be propagated forward do not have to be propagated with the fundamental matrix, but instead can be propagated directly by integrating the actual nonlinear differential equations forward at each sampling interval. For example, Euler integration can be applied to the nonlinear system differential equations, yielding: $\hat{x}_k^- = \hat{x}_{k-1} + \dot{x}_{k-1} T$, where the derivative is obtained from $\dot{x}_{k-1} = f(\hat{x}_{k-1})$.

- **Discrete extended Kalman filter measurement update equations**

$$\begin{aligned} K &= P_k^- H (H P_k^- H^T + R)^{-1} && \text{- Kalman gain matrix} \\ \hat{x}_k &= \hat{x}_k^- + K(z_k - h(\hat{x}_k^-)) && \text{- state observation update} \\ P_k &= (I - KH) P_k^- && \text{- error covariance update} \end{aligned} \qquad \text{eq. 4-43.}$$

The first task on the measurement update stage is to compute the Kalman gain *K*. The next step is to actually measure the process to obtain z_k, and then to generate an a posteriori state estimate by incorporating the measurement. As was already mentioned, the approximations for fundamental and measurement matrices only have to be used in the computation of the Kalman gains. The actual extended Kalman filter can be written in terms of the nonlinear state and measurement equations. With EKF the new state estimate is the old state estimate projected forward to the new sampling or measurement time plus a gain times a residual. The final step is to obtain an a posteriori error covariance estimate.

Because of their didactic value, sub-sections 4.4.1.3. and 4.4.1.4. are quotes from the (Roumeliotis, Sukhatmey, Bekey [99]).

4.4.1.3 Indirect versus direct Kalman filters

“A very important aspect of the implementation of a Kalman filter in conjunction with inertial navigation systems (INS) is the use of the indirect instead of the direct form, also referred to as the error state and the total state formulation respectively (Maybeck [89]). As the name indicates, in the total state direct formulation, total states such as orientation are among the variables in the filter, and the measurements are INS outputs, such as from gyros and external source signals. In the error state indirect formulation, the errors in orientation are among the estimated variables, and each measurement presented to the filter is the difference between the INS and the external source data.

There are some serious drawbacks in the direct filter implementation. Being in the INS loop and using the total state representation, the filter would have to maintain an explicit accurate awareness of the vehicle’s angular motion as well as attempt to suppress noisy and erroneous data. Sampled data require a sampling rate of at least twice the frequency of the highest frequency signal of interest (in practice, a factor of 5-10 is used) for adequate reconstruction of the continuous time system behavior. The filter would have to perform all the required computations within a short sampling period. In most cases, only a small portion of the compute resources of a central processor is usually allocated to the estimation algorithm and thus, to the Kalman filter, and moreover, it often runs in the background at a lower priority than more critical algorithms, such as real-time vision, obstacle avoidance, fault detection, etc.

In addition, the dynamics involved in the total state description of the filter include a high frequency component and are only well described by a nonlinear model. The development of a Kalman filter is based upon a linear system model, and as such a total state model does not exist.

Another disadvantage of the direct filter design is that if the filter fails (e.g. through a temporary computer failure) the entire navigation algorithm fails. The INS is useless without the filter. From a reliability point of view, it would be desirable to provide an emergency degraded performance mode in the event of such a failure. A direct filter would be ideal for fault detection and identification purposes.

The Indirect error state Kalman filter estimates the errors in the navigation and attitude information using the difference between the INS and external sources of data. The INS itself is able to follow the high frequency motions of the vehicle very accurately, and there is no need to model these dynamics explicitly in the filter. Instead, the dynamics upon which the Indirect filter is based are a set of inertial system error propagation equations, which have a low frequency behavior and are adequately described as linear. Because the filter is out of the INS loop and is based on low frequency dynamics, its sample rate can be much lower than that of the Direct filter. In fact an effective Indirect filter can be developed with a sample period of the external source, in the order of minutes [89]. This is very practical with respect to the amount of computer time required. For this reason, the error state formulation is used in essentially all terrestrial inertial navigation systems."[99]

4.4.1.4 Feedforward versus feedback indirect Kalman filters

"The basic difference between the feedforward and feedback Indirect Kalman filter is mainly in the way it handles the updated error estimate. In the first case, the updated error estimate is being fed forward to correct the current orientation estimate without updating the INS. In the feedback formulation the correction is actually fed back to the INS to correct its new starting point, i.e. the state from which the integration for the new time step will start. In a sense, the difference between the feedforward and feedback forms is equivalent to the difference between the Linearized Kalman filter and the Extended Kalman filter. In the second case, the state propagation starts from the corrected, updated state right after a measurement, while in the Linearized filter the propagation continues at the state that the propagation has reached when the measurement appeared, thus ignoring the correction just computed. The Linearized Kalman filter and the feedforward Indirect Kalman filter are free to drift with unbounded errors."[99]

The EKF assumes the measurement residual is small for the first order approximation in the calculation of Kalman gain to be accurate enough. If this assumption fails, the navigation states are erroneous and the solution may become unreliable.

4.4.2 Feedback Indirect Kalman Filter Equations

4.4.2.1 Gyro noise model

In the approach here, we use a simple and realistic model. In this model the angular velocity ω is related to the gyro output ω_m according to the equation:

$$\dot{\theta} = \omega_m + b + n_r \qquad \text{eq. 4-44.}$$

where b is the drift bias and n_r is the drift noise. Only n_r is assumed to be a Gaussian white-noise process, with zero mean and variance:

$$E[n_r(t)] = 0$$
$$E[n_r(t), n_r(\tau)] = Q_r\delta(t-\tau) \qquad \text{eq. 4-45.}$$

The drift bias is not a static quantity. The output of the gyro is known to err by an unknown slowly time-varying bias. The turn on bias is known and accounted for in the gyro calibration. Therefore, it is accurate to model the residual time-varying bias error as a random walk (a Brownian motion).

$$b(t) = \int_0^t n_\omega(\tau)d\tau \qquad \dot{b} = n_\omega$$
$$E[n_\omega(t)] = 0 \qquad \text{eq. 4-46.}$$
$$E[n_\omega(t), n_\omega(\tau)] = Q_\omega\delta(t-\tau) = \sigma_\omega^2$$

One often-quoted measure of sensor accuracy is the random-walk parameter. For example, a fiber-optic gyro might list its random walk parameter to be $5{,}0^0/\sqrt{h}$. The $\sqrt{h}$ in the denominator of the specification reflects that the standard deviation and the variance of the random-walk variable grow with the square root of time and linearly with time, respectively.

4.4.2.2 Quaternion error equations

The error state includes the bias error and quaternion error. The bias error is:

$$\Delta\vec{b} = \vec{b}_{true} - \vec{b}_i \qquad \text{eq. 4-47.}$$

The quaternion error is not the arithmetic difference, but is expressed as the quaternion which must be composed using the estimated quaternion to obtain the true quaternion:

$$\delta q = q_{true} \otimes q_i^{-1} \qquad \text{eq. 4-48.}$$

or, equivalently:

$$q_{true} = \delta q \otimes q_i \qquad \text{eq. 4-49.}$$

The advantage of this representation is that since the incremental quaternion corresponds very closely to a small rotation, the first component will be close to the unit and thus, the attitude information of interest is contained in the three vector component of the quaternion:

$$\delta q \cong \begin{bmatrix} 1 \\ \delta\vec{q} \end{bmatrix} \qquad \text{eq. 4-50.}$$

The physical counterparts of the quaternions are the rotation axis **n** and the rotation angle θ that are used in the Euler theorem regarding finite rotation. That is, taking the vector part of a quaternion and normalizing it, we can find the rotation axis right away, and from the first parameter we can obtain the angle of rotation. The unit quaternion is noted as:

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \qquad \text{eq. 4-51.}$$

with the constraint:

$$q^T q = 1 \qquad \sqrt{q_0^2 + q_1^2 + q_1^2 + q_1^2} = 1 \qquad \text{eq. 4-52.}$$

The real and vector part of the quaternion can be expressed using the rotation axis **n** and the rotation angle θ as:

$$q_0 = \cos\left(\frac{\theta}{2}\right) \qquad q_1 = n_x \sin\left(\frac{\theta}{2}\right) \qquad q_2 = n_y \sin\left(\frac{\theta}{2}\right) \qquad q_3 = n_z \sin\left(\frac{\theta}{2}\right) \qquad \text{eq. 4-53.}$$

with:

$$\mathbf{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \qquad \text{eq. 4-54.}$$

The rate of change of a quaternion with time can be expressed as:

$$\frac{d}{dt} q(t) = \frac{1}{2} \Omega(\vec{\omega}(t)) q(t) \qquad \text{eq. 4-55.}$$

with:

$$\Omega(\vec{\omega}) = \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & -\omega_3 & \omega_2 \\ \omega_2 & \omega_3 & 0 & -\omega_1 \\ \omega_3 & -\omega_2 & \omega_1 & 0 \end{bmatrix} \qquad \text{eq. 4-56.}$$

Starting from the equations:

$$\frac{dq_{true}}{dt} = \frac{1}{2} \Omega\left(\dot{\vec{\theta}}_{true}\right) q_{true} = \frac{1}{2} \begin{bmatrix} 0 \\ \dot{\vec{\theta}}_{true} \end{bmatrix} \otimes q_{true} \qquad \text{eq. 4-57.}$$

and

$$\frac{dq_i}{dt} = \frac{1}{2} \Omega\left(\dot{\vec{\theta}}_i\right) q_i = \frac{1}{2} \begin{bmatrix} 0 \\ \dot{\vec{\theta}}_i \end{bmatrix} \otimes q_i \qquad \text{eq. 4-58.}$$

where the operator $\otimes$ represents the quaternion multiplication (see Chapter A.3.1). From Equation 4-48 we have:

$$\frac{d}{dt}\delta q = \frac{d}{dt}(q_{true} \otimes q_i^{-1}) = \frac{d}{dt}q_{true} \otimes q_i^{-1} + q_{true} \otimes \frac{d}{dt}q_i^{-1} \quad \text{eq. 4-59.}$$

From Equation 4-57 we can express:

$$\frac{d}{dt}q_{true} \otimes q_i^{-1} = \frac{1}{2}\Omega\left(\dot{\vec{\theta}}_{true}\right)q_{true} \otimes q_i^{-1} = \frac{1}{2}\begin{bmatrix} 0 \\ \dot{\vec{\theta}}_{true} \end{bmatrix} \otimes \delta q \quad \text{eq. 4-60.}$$

To deduce the second term from Equation 4-59 and Equation 4-57 we can write:

$$\frac{d}{dt}(q_i^{-1}q_i) = 0 \Rightarrow \frac{d}{dt}q_i^{-1} = -q_i^{-1}\left(\frac{dq_i}{dt}\right)q_i^{-1}$$
$$q_{true}\frac{d}{dt}q_i^{-1} = -\frac{1}{2}q_{true} \otimes q_i^{-1} \otimes \begin{bmatrix} 0 \\ \dot{\vec{\theta}}_i \end{bmatrix} \otimes q_i \otimes q_i^{-1} \quad \text{eq. 4-61.}$$

From the previous equation we have the result:

$$q_{true}\frac{d}{dt}q_i^{-1} = -\frac{1}{2}\delta q \otimes \begin{bmatrix} 0 \\ \dot{\vec{\theta}}_i \end{bmatrix} \quad \text{eq. 4-62.}$$

Substituting Equation 4-60 and Equation 4-62 in Equation 4-59 we obtain:

$$\frac{d}{dt}\delta q = \frac{1}{2}\begin{bmatrix} 0 \\ \dot{\vec{\theta}}_{true} \end{bmatrix} \otimes \delta q - \frac{1}{2}\delta q \otimes \begin{bmatrix} 0 \\ \dot{\vec{\theta}}_i \end{bmatrix} \quad \text{eq. 4-63.}$$

The true orientation is noted by $\dot{\theta}_{true}$, the orientation estimate obtained by integrating the gyro signal is noted by $\dot{\theta}_i$, whereas ω_m is the angular rate measured by the gyro:

$$\dot{\vec{\theta}}_{true} = \vec{\omega}_m + \vec{b}_{true} + \vec{n}_r$$
$$\dot{\vec{\theta}}_i = \vec{\omega}_m + \vec{b}_i \quad \text{eq. 4-64.}$$

Substituting this in Equation 4-63 and rearranging it we obtain:

$$\frac{d}{dt}\delta q = \frac{1}{2}\begin{bmatrix} 0 \\ \vec{\omega}_m + \vec{b}_{true} + \vec{n}_r \end{bmatrix} \otimes \delta q - \frac{1}{2}\delta q \otimes \begin{bmatrix} 0 \\ \vec{\omega}_m + \vec{b}_i \end{bmatrix} \quad \text{eq. 4-65.}$$
$$= \frac{1}{2}\left(\begin{bmatrix} 0 \\ \vec{\omega}_m \end{bmatrix} \otimes \delta q - \delta q \otimes \begin{bmatrix} 0 \\ \vec{\omega}_m \end{bmatrix}\right) + \frac{1}{2}\left(\begin{bmatrix} 0 \\ \vec{b}_{true} + \vec{n}_r \end{bmatrix} \otimes \delta q - \delta q \otimes \begin{bmatrix} 0 \\ \vec{b}_i \end{bmatrix}\right)$$

At this point we need to use knowledge about combining two quaternion vectors. With:

$$\mathbf{a} = \begin{bmatrix} a_0 \\ \vec{a} \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} b_0 \\ \vec{b} \end{bmatrix} \qquad \text{eq. 4-66.}$$

then their product is:

$$\mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_0 b_0 - \vec{a}^T \vec{b} \\ a_0 \vec{b} + b_0 \vec{a} - \vec{a} \times \vec{b} \end{bmatrix} \qquad \text{eq. 4-67.}$$

Now we can simplify the first term in Equation 4-65 and knowing that $\vec{b} \times \vec{a} = -\vec{a} \times \vec{b}$,we have:

$$\begin{bmatrix} 0 \\ \vec{\omega}_m \end{bmatrix} \otimes \begin{bmatrix} \delta q_0 \\ \delta \vec{q} \end{bmatrix} - \begin{bmatrix} \delta q_0 \\ \delta \vec{q} \end{bmatrix} \otimes \begin{bmatrix} 0 \\ \vec{\omega}_m \end{bmatrix} = \qquad \text{eq. 4-68.}$$

$$\begin{bmatrix} 0(\delta q_0) - \vec{\omega}_m^T \delta \vec{q} \\ 0(\delta \vec{q}) + \vec{\omega}_m \delta \vec{q} - \vec{\omega}_m \times \delta \vec{q} \end{bmatrix} - \begin{bmatrix} (\delta q_0)0 - \delta \vec{q}^T \vec{\omega}_m \\ (\delta \vec{q})0 + \delta \vec{q} \vec{\omega}_m - \delta \vec{q} \times \vec{\omega}_m \end{bmatrix} = -2 \begin{bmatrix} 0 \\ \vec{\omega}_m \times \delta \vec{q} \end{bmatrix}$$

Similarly the second part of Equation 4-65 can be written as:

$$\begin{bmatrix} 0 \\ \vec{b}_{true} + \vec{n}_r \end{bmatrix} \otimes \begin{bmatrix} \delta q_0 \\ \delta \vec{q} \end{bmatrix} - \begin{bmatrix} \delta q_0 \\ \delta \vec{q} \end{bmatrix} \otimes \begin{bmatrix} 0 \\ \vec{b}_i \end{bmatrix} = \qquad \text{eq. 4-69.}$$

$$\begin{bmatrix} 0(\delta q_0) - (\vec{b}_{true} + \vec{n}_r)^T \delta \vec{q} \\ 0(\delta \vec{q}) + \delta q_0 (\vec{b}_{true} + \vec{n}_r) - (\vec{b}_{true} + \vec{n}_r) \times \delta \vec{q} \end{bmatrix} - \begin{bmatrix} (\delta q_0)0 - \delta \vec{q}^T \vec{b}_i \\ (\delta \vec{q})0 + \delta q_0 \vec{b}_i - \delta \vec{q} \times \vec{b}_i \end{bmatrix} =$$

$$\begin{bmatrix} -(\vec{b}_{true} - \vec{b}_i + \vec{n}_r)^T \delta \vec{q} \\ \delta q_0 (\vec{b}_{true} - \vec{b}_i + \vec{n}_r) - (\vec{b}_{true} - \vec{b}_i + \vec{n}_r) \times \delta \vec{q} \end{bmatrix} = \begin{bmatrix} 0 \\ \delta \vec{\omega} \end{bmatrix} \delta q_0 - \begin{bmatrix} \delta \vec{\omega}^T \delta \vec{q} \\ \delta \vec{\omega} \times \delta \vec{q} \end{bmatrix}$$

with:

$$\delta \vec{\omega} = \vec{b}_{true} - \vec{b}_i + \vec{n}_r = \Delta \vec{b} + \vec{n}_r \qquad \text{eq. 4-70.}$$

For very small rotations we can assume that

$$\delta q_0 \cong 1 \qquad \delta \vec{q} \cong \vec{0} \qquad \text{eq. 4-71.}$$

Then, Equation 4-69 can be written as:

$$\begin{bmatrix} 0 \\ \vec{b}_{true} + \vec{n}_r \end{bmatrix} \otimes \begin{bmatrix} \delta q_0 \\ \delta \vec{q} \end{bmatrix} - \begin{bmatrix} \delta q_0 \\ \delta \vec{q} \end{bmatrix} \otimes \begin{bmatrix} 0 \\ \vec{b}_i \end{bmatrix} \cong \begin{bmatrix} 0 \\ \delta \vec{\omega} \end{bmatrix} \qquad \text{eq. 4-72.}$$

Now substituting Equation 4-72 and Equation 4-68 into Equation 4-65 we have:

$$\frac{d}{dt}\delta q = \begin{bmatrix} 0 \\ \vec{\omega}_m \times \delta\vec{q} \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 \\ \delta\vec{\omega} \end{bmatrix} \qquad \text{eq. 4-73.}$$

Now if we know that:

$$\vec{\omega}_m \times \delta\vec{q} = -[[\vec{\omega}_m]]\delta\vec{q} = \begin{bmatrix} 0 & -\omega_{m3} & \omega_{m2} \\ \omega_{m3} & 0 & -\omega_{m1} \\ -\omega_{m2} & \omega_{m1} & 0 \end{bmatrix}\begin{bmatrix} \delta q_1 & \delta q_2 & \delta q_3 \end{bmatrix} \qquad \text{eq. 4-74.}$$

Then from Equation 4-70, Equation 4-73 and Equation 4-74, and separating the vector from the scalar terms, we obtain the equation that is used in the Kalman filter:

$$\begin{aligned} \frac{d}{dt}\delta q_0 &= 0 \\ \frac{d}{dt}\delta\vec{q} &= [[\vec{\omega}_m]]\delta\vec{q} + \frac{1}{2}(\Delta\vec{b} + \vec{n}_r) \end{aligned} \qquad \text{eq. 4-75.}$$

By using the infinitesimal angle assumption in Equation 4-53, that is *sin*(*a*) = *a*, we can write the difference of the quaternion vector part as:

$$\delta\vec{q} = \frac{1}{2}\delta\vec{\theta} \qquad \text{eq. 4-76.}$$

Now, using Equation 4-75 and Equation 4-76, we are ready to write the equation that it is used to describe the Kalman error state in terms of Euler angles:

$$\frac{d}{dt}\delta\vec{\theta} = [[\vec{\omega}_m]]\delta\vec{\theta} + (\Delta\vec{b} + \vec{n}_r) \qquad \text{eq. 4-77.}$$

4.4.2.3 The quaternion error in an indirect Kalman filter

The continuous differential equation expressed in quaternion error is derived from Equation 4-75. The matrix form is useful to determine the F matrix.

$$\begin{bmatrix} \delta\dot{q}_0 \\ \delta\dot{q}_1 \\ \delta\dot{q}_2 \\ \delta\dot{q}_3 \\ \Delta\dot{b}_x \\ \Delta\dot{b}_y \\ \Delta\dot{b}_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -wz & wy & \frac{1}{2} & 0 & 0 \\ 0 & wz & 0 & -wx & 0 & \frac{1}{2} & 0 \\ 0 & -wy & wx & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \delta q_0 \\ \delta q_1 \\ \delta q_2 \\ \delta q_3 \\ \Delta b_x \\ \Delta b_y \\ \Delta b_z \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ n_{rx} \\ n_{ry} \\ n_{rz} \end{bmatrix} \qquad \text{eq. 4-78.}$$

4.4.2.4 The Euler angle error in an indirect Kalman filter

The continuous differential equation with quaternion errors is derived from Equation 4-77. The matrix form is useful to determine the F matrix.

$$\begin{bmatrix} \delta\dot{\theta}_x \\ \delta\dot{\theta}_y \\ \delta\dot{\theta}_z \\ \Delta\dot{b}_x \\ \Delta\dot{b}_y \\ \Delta\dot{b}_z \end{bmatrix} = \begin{bmatrix} 0 & -wz & wy & 1 & 0 & 0 \\ wz & 0 & -wx & 0 & 1 & 0 \\ -wy & wx & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta\theta_x \\ \delta\theta_y \\ \delta\theta_z \\ \Delta b_x \\ \Delta b_y \\ \Delta b_z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ n_{rx} \\ n_{ry} \\ n_{rz} \end{bmatrix} \qquad \text{eq. 4-79.}$$

The Euler angles are observable if we have inclinometer information. The angle observation equation, and observation matrix are:

$$\begin{aligned} \delta z_{\theta_{xyz} b_{xyz}} &= H_{\theta_{xyz} b_{xyz}} \delta x + \varepsilon_{XYZ} \\ H_{\theta_{xyz} b_{xyz}} &= \left[\mathbf{I}^{3x3} \ \mathbf{0}^{3x3} \right] \end{aligned} \qquad \text{eq. 4-80.}$$

Ideally, Q_k is supposed to reflect the magnitude of a white noise sequence. If all error sources in the inertial attitude system are taken care of (i.e. modeled in the state propagation matrix), then w_k should be entirely due to the noise floors of the angular rate sensors. In this case, it should be possible to calculate the optimal value of Q_k by measuring the noise covariance, Q, of the stationary gyros in advance, then at each time step compute $Q_k = G_k Q G_k^T$.

However, there are many nonwhite error sources besides bias, such as nonlinearity, hysteresis, misalignment, g-sensitivity, and a scale factor temperature coefficient, none of which are modeled in the state propagation matrix. This is due to the fact that for low cost silicon gyros the most important source of errors is the bias. A simpler approach to the implementation of those errors in the state propagation matrix, which sometimes works almost as well [88], is to just ignore the unmodeled states, but make the Q and R matrices account for the noise in the states that were discarded.

The model of gyro dynamics for a long duration is exponential, but τ is in the order of 30 - 60 minutes. For the run time of the Kalman filter we can consider that the drift error is linear. We assume the following error sources in the process equation:

- gyro noise: from digital analysis, for stationary gyros, we found $\sigma = 0.01$ rad/s. The covariance per step is $(0.01\Delta t)^2$.
- integration rule error: zero because the integration is achieved by an exact formula.
- scale factor error: This is a composite of a nonlinearity and a temperature-dependent scale factor variation. Assuming a scale factor accuracy of 1% full scale, $\sigma = 0.01\omega$ rad/s. The covariance per step is $(0.01\omega\Delta t)^2$.

Q_k is constructed as follows:

$$Q_k = \begin{bmatrix} \sigma_w^2 & 0 & 0 \\ 0 & \sigma_w^2 & 0 \\ 0 & 0 & \sigma_w^2 \end{bmatrix} \qquad \text{eq. 4-81.}$$

where $\sigma_w = 10^{-8}(1+\omega^2)$, assuming $\Delta t = 0.01$ sec.

R_k is modeled in a similar experimental way. The measurement noise is extremely nonwhite. The major source of measurement noise for the fluid inclinometers is "slosh" caused by transverse linear accelerations [77]. Linear motion is not included in the state vector, and therefore, this error cannot be modeled in the measurement matrix. Furthermore, the magnitude of the low-frequency "slosh" errors are sometimes extremely large: up to 20 degrees (0.35 radian). On the other hand, when the head is still, there is no slosh and the attitude angles measured by the inclinometer are very accurate. The algorithm for R_k is therefore designed in a heuristic way to force the Kalman filter to take advantage of the measurements, when they are likely to be meaningful, and to ignore them when they are likely to be erroneous. The basic principle is that σ_v should approach 1 when slosh is likely, and approach the static accuracy of the inclinometer/compass measurements, about 0.2 degree (0.004 radian), when slosh is very unlikely. In the absence of a model for human head motion, we use the information from gyros and accelerometers. The longer the period of time that the head has 1) zero angular velocity, and 2) the square root of the sum of square accelerations approximates 1, the higher the probability that the head is still. Based on this, the algorithm used to set R_k is:

1. compute the "stilltime" τ since the last non-zero gyro reading OR the last time when $\sqrt{a_x^2 + a_y^2 + a_z^2} \gg 1$, and when the magnetic alarm from the magnetometer (TCM2) is set, set σ_r to 1.
2. set $\sigma_v = 1/(1+400\tau)$, and if magnetic alarm is 0 set $\sigma_r = \sigma_v$.
3. if $\sigma_v < 0.004$, set $\sigma_v = 0.004$, and $\sigma_r < 0.002$, set $\sigma_r = 0.002$.
4. set R_k as in Equation 4-82.

$$R_k = \begin{bmatrix} \sigma_v^2 & 0 & 0 \\ 0 & \sigma_v^2 & 0 \\ 0 & 0 & \sigma_r \end{bmatrix} \qquad \text{eq. 4-82.}$$

According to this algorithm, the measurement error covariance for inclinometer roll and pitch ranges from 1, during periods of likely slosh, down to 10^{-4}, during periods of likely stillness. The covariance of the compass yaw error is 1 when the magnetic alarm is set. It varies with the inclinometer information because it is used in the computation of the yaw angle and comes down only to 0.002, corresponding to 1.5 degrees, the heading accuracy.

Typical input signals from gyros and inclinometer and magnetometer (TCM2 sensor) are:

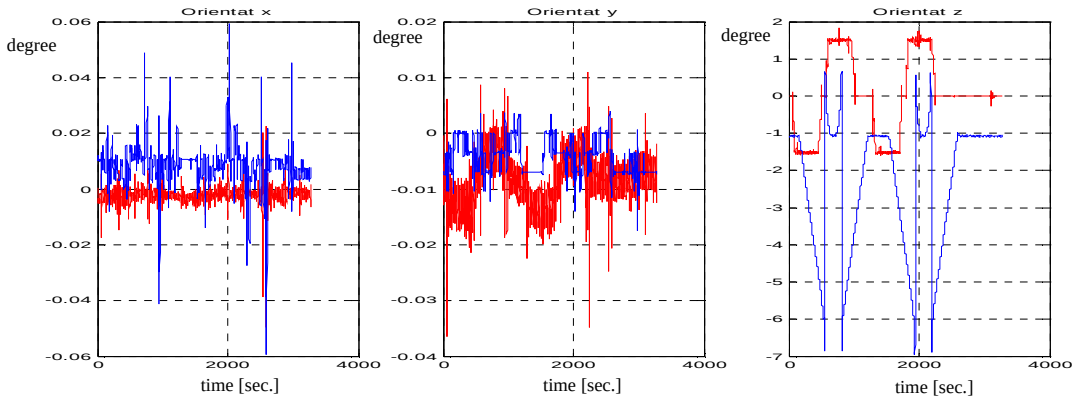

Figure 4-6. The input signals for the Kalman filter

To demonstrate the behavior of the Kalman filter, two datasets were collected. In the first dataset, the complementary Kalman filter block is disabled by setting K, the Kalman gain, equal to zero. During a test period of approximately 35 seconds, the sensor was repeatedly turned through $+180^o$ and -180^o around the z axis (yaw). The yaw angle is plotted against time in blue in Figure 4-7, which demonstrates the problem with unaided inertial integration: the accumulated drift error by the end of the run is about 7^o. This dataset was acquired after the sensors warmed up for 1 hour, and a zero scale calibration of 1 second. The second dataset is created by a similar motion sequence, but with the Kalman filter in use; the results are plotted in red. The filter incorporates the drift-free but noisy measurement from inclinometers and compass, and effectively compensates the drift of the inertial system.

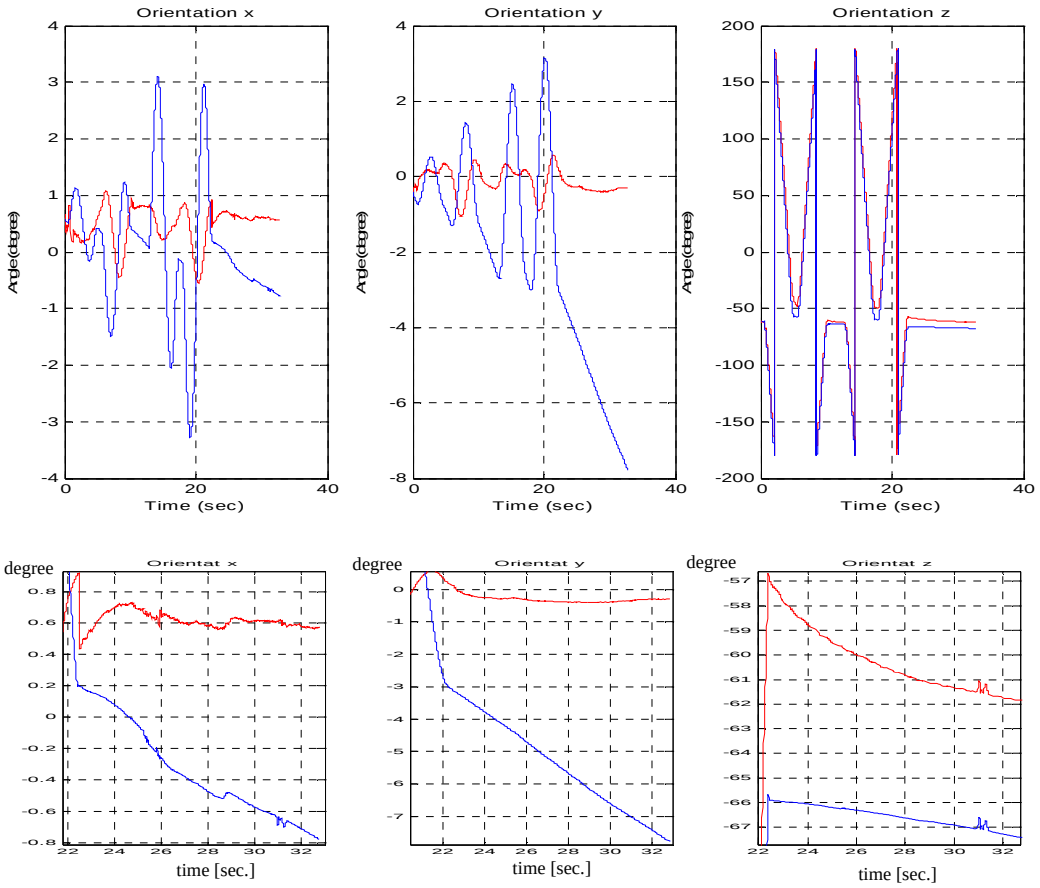

Figure 4-7. Kalman filter results

Due to the time-varying R_k strategy which shuts out the measurements during extensive motion, a certain amount of error accumulates each time the sensor is rolled over and back, and the Kalman filter corrects it once the sensor returns to a stationary pose. The graph in Figure 4-8 shows the drift estimated signals over time. This signal is estimated by a Kalman filter and subtracted from gyro measurements in order to obtain the corrected angle after integration.

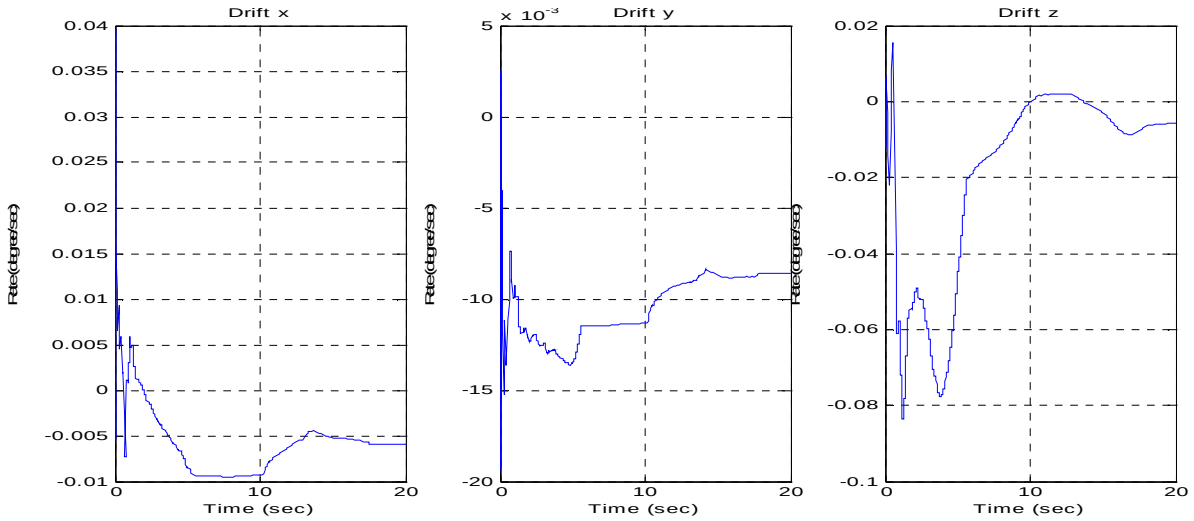

Figure 4-8. Drift estimated by Kalman filter

:

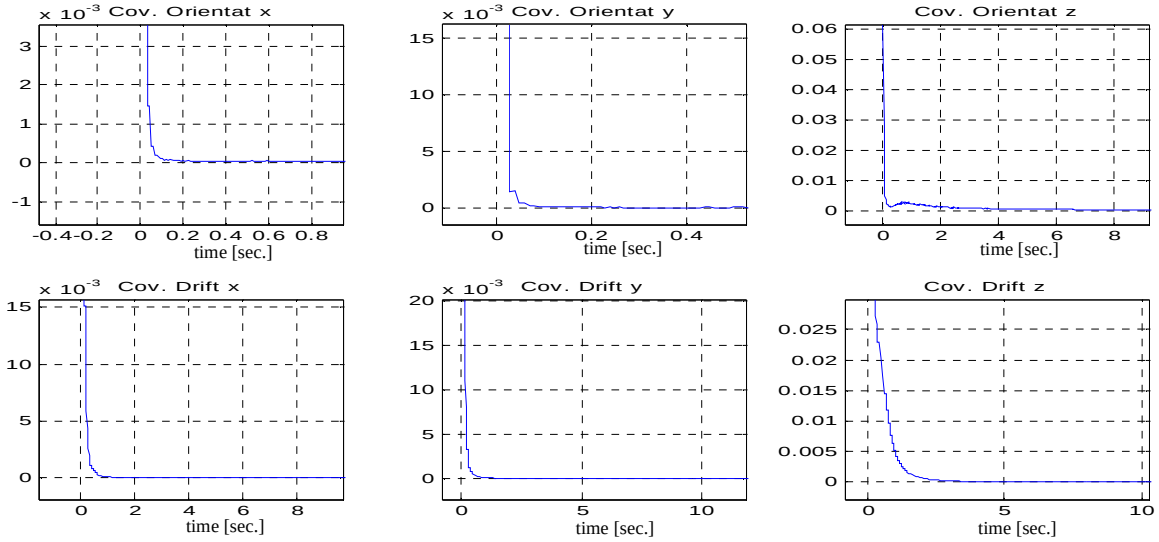

Figure 4-9. Kalman filter Covariance

Figure 4-9 displays plots of the error variance as a function of time. The variance grows between the time instants (time updates) at which angle measurements are available. The angle measurements decrease the error variance of each state estimates at an 0.1 sec. (measurement update) interval.

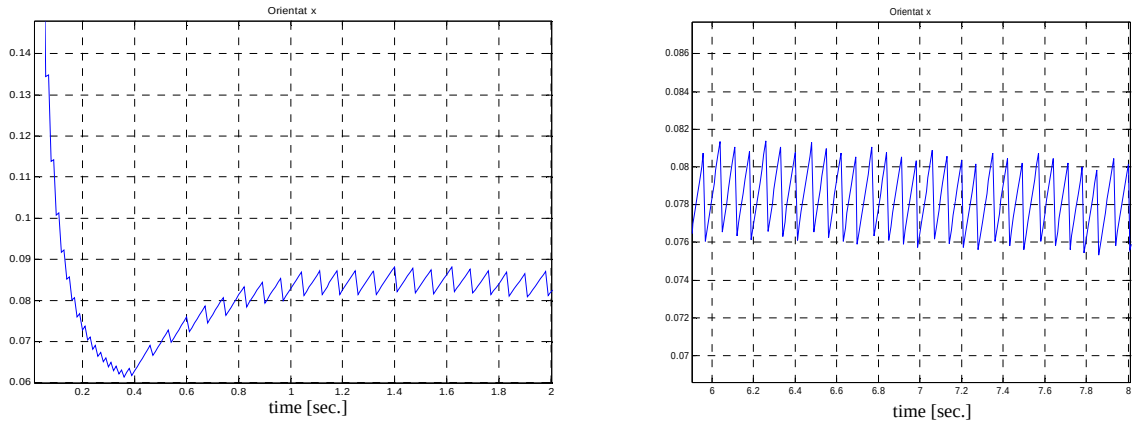

Figure 4-10. Covariance variation over time

Results showed that the orientation Kalman filter converges quickly (Figure 4-10). The stability of the gyros provides very accurate orientation values during motion, but due to the resolution of the inclinometer (0.2 degrees) the overall accuracy cannot be higher than 0.2 degrees, even when the 0.5 degrees accuracy in the heading has no systematic error.

4.4.2.5 A linear error model

A very common linear error model is found in [87].

$$\begin{bmatrix} \delta\dot{r}_N \\ \delta\dot{r}_E \\ \delta\dot{r}_D \\ \delta\dot{v}_N \\ \delta\dot{v}_E \\ \delta\dot{v}_D \\ \delta\dot{\theta}_x \\ \delta\dot{\theta}_y \\ \delta\dot{\theta}_z \end{bmatrix} = \begin{bmatrix} 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&a_D&-a_E \\ 0&0&0&0&0&0&-a_D&0&a_N \\ 0&0&0&0&0&0&a_E&-a_N&0 \\ 0&0&0&0&0&0&0&wz&-wy \\ 0&0&0&0&0&0&-wz&0&wx \\ 0&0&0&0&0&0&wy&-wx&0 \end{bmatrix} \begin{bmatrix} \delta r_N \\ \delta r_E \\ \delta r_D \\ \delta v_N \\ \delta v_E \\ \delta v_D \\ \delta\theta_x \\ \delta\theta_y \\ \delta\theta_z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ n_{aN} \\ n_{aE} \\ n_{aD} \\ n_{\theta x} \\ n_{\theta y} \\ n_{\theta z} \end{bmatrix} \qquad \text{eq. 4-83.}$$

We explain it using Figure 4-11, in which we have the accelerations [a_N, a_E, a_D] in a navigation frame. The IMU is misaligned with small angles [ϕ, θ, ψ]. We want to know the resulting acceleration on the frame axes caused by the misalignment.

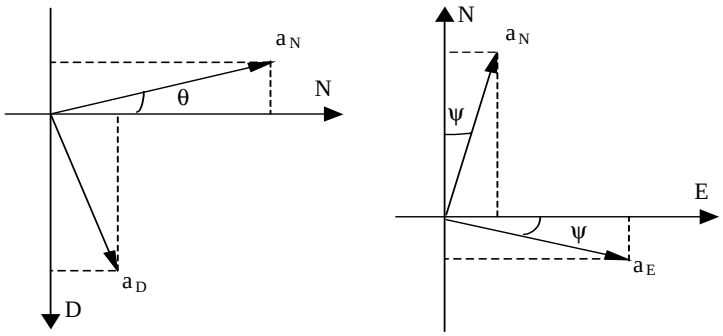

Figure 4-11. IMU misalignment

From the figure we compute the resulting acceleration a_N and we see that the misalignment angles ψ and θ will project the acceleration a_D and a_E to the *x*-axis (*N*). From the figure we set up the equation:

$$\dot{v}_N = a_D \sin(\theta) - a_E \sin(\psi) \qquad \text{eq. 4-84.}$$

If ψ and θ are small we can approximate the equation with:

$$\dot{v}_N = a_D \theta - a_E \psi \qquad \text{eq. 4-85.}$$

We get $\delta\dot{V} = A_n \times \delta\Psi$, were A_n is a skew symmetric matrix.

$$A_N \times = \begin{bmatrix} 0 & a_D & -a_E \\ -a_D & 0 & a_N \\ a_E & -a_N & 0 \end{bmatrix} \qquad \text{eq. 4-86.}$$

This model works fine if the trigonometric functions can be approximated with a first order Taylor expansion. We can use this model in the leveling process. The misalignment angles can be found if the IMU is stationary and if the readouts from IMU say that the person is moving. We can notice that the misalignment on a stationary person will project the acceleration caused by the gravity forward if we have a small roll angle, and to the left if we have a small pitch angle.

4.4.2.6 A linear model for position estimation

We consider that the delta speed and the delta position are represented in the navigation coordinate system, and the delta acceleration (the drift) is in the body reference coordinate system.

$$\Phi_k = \begin{bmatrix} I & I \cdot \Delta t & R \cdot \frac{\Delta t^2}{2} \\ 0 & I & R \cdot \Delta t \\ 0 & 0 & I \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & r_{11}\frac{\Delta t^2}{2} & r_{12}\frac{\Delta t^2}{2} & r_{13}\frac{\Delta t^2}{2} \\ 0 & 1 & 0 & 0 & \Delta t & 0 & r_{21}\frac{\Delta t^2}{2} & r_{22}\frac{\Delta t^2}{2} & r_{23}\frac{\Delta t^2}{2} \\ 0 & 0 & 1 & 0 & 0 & \Delta t & r_{31}\frac{\Delta t^2}{2} & r_{32}\frac{\Delta t^2}{2} & r_{33}\frac{\Delta t^2}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 & r_{11}\Delta t & r_{12}\Delta t & r_{13}\Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 & r_{21}\Delta t & r_{22}\Delta t & r_{23}\Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 & r_{31}\Delta t & r_{32}\Delta t & r_{33}\Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{eq. 4-87.}$$

$$Q_k = \begin{bmatrix} \frac{W}{20}\Delta t^5 & \frac{W}{8}\Delta t^4 & \frac{W}{6}\Delta t^3 \\ \frac{W}{8}\Delta t^4 & \frac{W}{3}\Delta t^3 & \frac{W}{2}\Delta t^2 \\ \frac{W}{6}\Delta t^3 & \frac{W}{2}\Delta t^2 & W\Delta t \end{bmatrix} \quad \text{eq. 4-88.}$$

Figure 4-12 shows how the accelerometer bias is modeled as a random walk process. The bias is modeled as an integrated white noise with a power spectral density (PSD) of W. The complete process is modeled by three integrators in cascade. From this model, the exact expressions for Φ_k and Q_k can be worked out to the form as shown above. The power spectral density of the input white noise W is 1 $(m/s^2)^2/(rad/s)$, and the sampling time Δt equals to 1/100 s. The value of W was obtained after performing some experiments aimed to provide better results for the bias estimation.

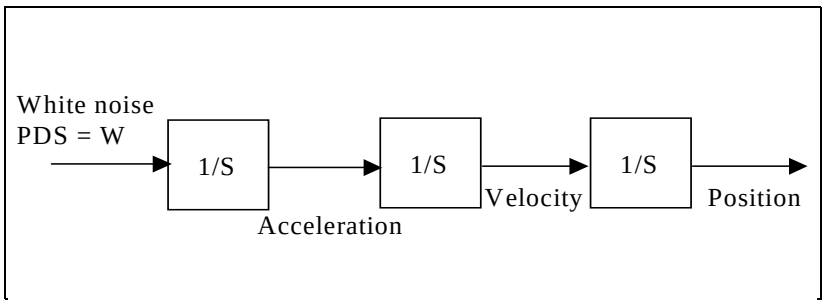

Figure 4-12. The process model of the accelerometer for Kalman Filter

The result for the Kalman filter simulation for accelerometer signals with drift are:

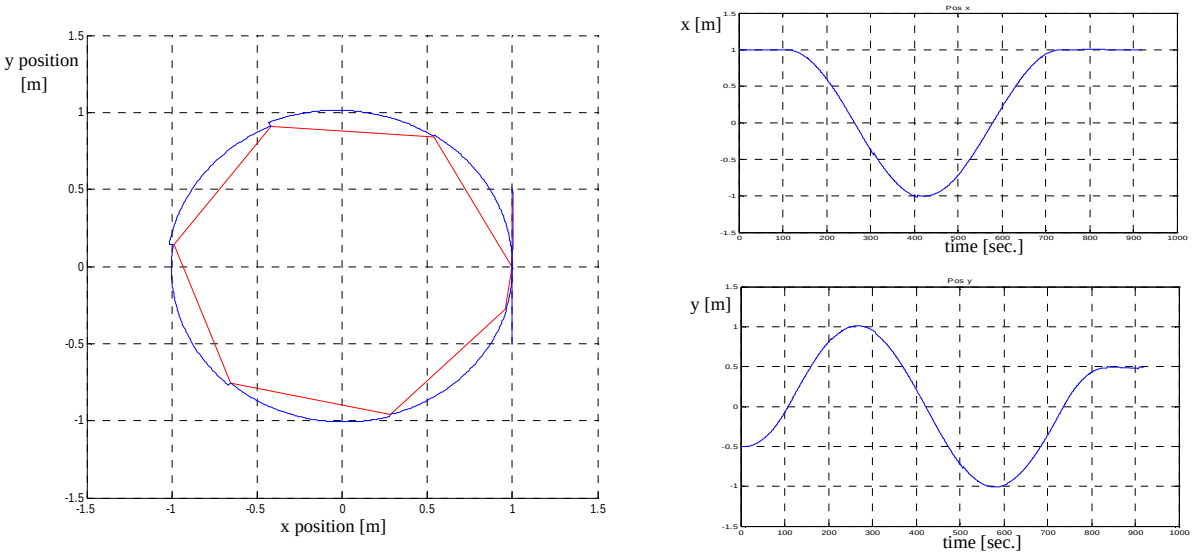

Figure 4-13. Position Kalman filter simulation results

As we can see, the filter is able to correct the accelerometer bias and the estimated trajectory is maintained closer to the actual one. Another conclusion is that the accuracy of the estimated trajectory depends on the absolute measurement updates and also on the accuracy of those updates.

4.4.2.7 A nonlinear model with quaternions

The nonlinear state transition matrix is:

$$f(x,u) = \begin{bmatrix} (q_0^2+q_1^2-q_3^2-q_2^2)U+(-2q_0q_3+2q_2q_1)V+(2q_3q_1+2q_0q_2)W \\ (2q_2q_1+2q_0q_3)U+(-q_3^2-q_1^2+q_0^2+q_2^2)V+(-2q_0q_1+2q_3q_2)W \\ (2q_3q_1-2q_0q_2)U+(2q_0q_1+2q_3q_2)V+(-q_2^2-q_1^2+q_3^2+q_0^2)W \\ VR-WQ+g(2q_3q_1-2q_0q_2)+a_x \\ WP-UR+g(2q_0q_1+2q_3q_2)+a_y \\ UQ-VP+g(-q_2^2-q_1^2+q_3^2+q_0^2)+a_z \\ \frac{-1}{2}q_1P+\frac{-1}{2}q_2Q+\frac{-1}{2}q_3R \\ \frac{1}{2}q_0P+\frac{-1}{2}q_3Q+\frac{1}{2}q_2R \\ \frac{1}{2}q_3P+\frac{1}{2}q_0Q+\frac{-1}{2}q_1R \\ \frac{-1}{2}q_2P+\frac{1}{2}q_1Q+\frac{1}{2}q_0R \end{bmatrix} \quad \text{eq. 4-89.}$$

The state vector of this nonlinear model has the elements position, velocity in the body frame and attitude expressed by the quaternions, $x = [X, Y, Z, U, V, W, q_0, q_1, q_2, q_3]$. The quaternions have the advantage that we use polynomials of order two in the continuous time transition state matrix.

The nonlinear state transition function is linearized to obtain the matrix for the extended Kalman filter. The matrix contains the partial derivatives of f with respect to x.

```
t1 = q0*q0;     t2 = q1*q1;     t3 = q3*q3;     t4 = q2*q2;
t6 = q0*q3;     t7 = q2*q1;     t9 = q3*q1;     t10 = q0*q2;
t15 = q0*U-q3*V+q2*W;       t19 = q1*U+q2*V+q3*W;
t23 = -q2*U+q1*V+q0*W;    t27 = -q3*U-q0*V+q1*W;
t30 = q0*q1;    t31 = q3*q2;
t37 = 2.0*g*q2;          t39 = 2.0*g*q3;          t41 = 2.0*g*q0;          t43 = 2.0*g*q1;
t44 = P/2.0;     t45 = Q/2.0;     t46 = R/2.0;
dfx[0][0...9] = [0.0; 0.0; 0.0; t1+t2-t3-t4; -2.0*t6+2.0*t7; 2.0*t9+2.0*t10; 2.0*t15; 2.0*t19;
2.0*t23; 2.0*t27];
dfx[1][0...9] = [0.0; 0.0; 0.0; 2.0*t7+2.0*t6; -t3-t2+t1+t4; -2.0*t30+2.0*t31; -2.0*t27; -2.0*t23;
2.0*t19; 2.0*t15];
dfx[2][0...9] = [0.0; 0.0; 0.0; 2.0*t9-2.0*t10; 2.0*t30+2.0*t31; -t4-t2+t3+t1; 2.0*t23; -2.0*t27;
-2.0*t15; 2.0*t19];
dfx[3][0...9] = [0.0; 0.0; 0.0; 0.0; R; -Q; -t37; t39; -t41; t43];
dfx[4][0...9] = [0.0; 0.0; 0.0; -R; 0.0; P; t43; t41; t39; t37];
dfx[5][0...9] = [0.0; 0.0; 0.0; Q; -P; 0.0; t41; -t43; -t37; t39];
dfx[6][0...9] = [0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; -t44; -t45; -t46];
dfx[7][0...9] = [0.0; 0.0; 0.0; 0.0; 0.0; 0.0; t44; 0.0; t46; -t45];
dfx[8][0...9] = [0.0; 0.0; 0.0; 0.0; 0.0; 0.0; t45; -t46; 0.0; t44];
dfx[9][0...9] = [0.0; 0.0; 0.0; 0.0; 0.0; 0.0; t46; t45; -t44; 0.0];
```

The derivative matrix was computed with Maple, which was also used to generate the C code. In a later stage we used a dedicated C library to compute the matrix exponential to get the Φ matrix. For a complicated matrix it is impossible to obtain an analytic form for the matrix exponential.

4.4.2.8 Observation Models

- *Position Observation Model*

The position is observable if we have GPS position information. The position observation equation and observation matrix are:

$$\begin{aligned} \delta z_{XYZ} &= H_{XYZ}\delta x + \varepsilon_{XYZ} \\ H_{XYZ} &= \left[\mathbf{I}^{3x3}\ \mathbf{0}^{7x3}\right] \end{aligned} \qquad \text{eq. 4-90.}$$

- *Velocity Observation Model*

Some GPS receivers are able to compute velocity information as well. When we have velocity information then the observation matrix for speed is nonzero. The velocity in the state vector x is represented in body axis. The velocities (u, v, w) need to be transformed to the navigation frame, and this is done with:

$$H_{XYZ} = \left[\mathbf{0}^{3x3}\ C_b^n\ \mathbf{0}^{3x3}\right] \qquad \text{eq. 4-91.}$$

where C_b^n is the rotation matrix from body to navigation frame.

- *Attitude Observation Model*

Because we use quaternions in the state vector and the observations are in an Euler angle representation we need to do a conversion. The *h* function is presented below (see Equation A-18), and we can see that it is nonlinear.

$$h_{\psi,\,\theta,\,\phi}(x) = \begin{bmatrix} \arctan\dfrac{(2e_0e_1 + 2e_2e_3)}{(e_0^2 - e_1^2 - e_2^2 + e_3^2)} \\ \arcsin(2e_0e_2 - 2e_3e_1) \\ \arctan\dfrac{(2e_0e_3 + 2e_1e_2)}{(e_0^2 + e_1^2 - e_2^2 - e_3^2)} \end{bmatrix} \qquad \text{eq. 4-92.}$$

It needs to be linearized along the trajectory. It gives the expression for the attitude observation matrix

$$H_{\psi,\,\theta,\,\phi} = \begin{bmatrix} \mathbf{0}^{3x6} & \dfrac{\partial h_{\psi,\,\theta,\,\phi}(x)}{\partial e_0} & \dfrac{\partial h_{\psi,\,\theta,\,\phi}(x)}{\partial e_1} & \dfrac{\partial h_{\psi,\,\theta,\,\phi}(x)}{\partial e_2} & \dfrac{\partial h_{\psi,\,\theta,\,\phi}(x)}{\partial e_3} \end{bmatrix} \qquad \text{eq. 4-93.}$$

where the partial derivatives are:

$$\frac{\partial h_{\psi,\,\theta,\,\phi}(x)}{\partial e_0} = \begin{bmatrix} \dfrac{-2e1e0^2 - 2e1^3 - 2e1e2^2 + 2e1e3^2 - 4e2e0e3}{e0^4 + 2e0^2e1^2 - 2e0^2e2^2 + 2e0^2e3^2 + e1^4 + 2e1^2e2^2 - 2e3^2e1^2 + e2^4 + 2e2^2e3^2 + e3^4 + 8e0e2e3e1} \\ \dfrac{2e2}{\sqrt{1 - 4e0^2e2^2 + 8e0e2e3e1 - 4e3^2e1^2}} \\ \dfrac{-2e3e0^2 + 2e3e1^2 + -2e3e2^2 + -2e3^3 + -4e2e0e1}{e0^4 + 2e0^2e1^2 - 2e0^2e2^2 + 2e0^2e3^2 + e1^4 + 2e1^2e2^2 - 2e3^2e1^2 + e2^4 + 2e2^2e3^2 + e3^4 + 8e0e2e3e1} \end{bmatrix} \qquad \text{eq. 4-94.}$$

$$\frac{\partial h_{\psi,\,\theta,\,\phi}(x)}{\partial e_1} = \begin{bmatrix} \dfrac{2e0^3 + 2e0e1^2 - 2e0e2^2 + 2e0e3^2 + 4e3e1e2}{e0^4 + 2e0^2e1^2 - 2e0^2e2^2 + 2e0^2e3^2 + e1^4 + 2e1^2e2^2 - 2e3^2e1^2 + e2^4 + 2e2^2e3^2 + e3^4 + 8e0e2e3e1} \\ \dfrac{-2e3}{\sqrt{1 - 4e0^2e2^2 + 8e0e2e3e1 - 4e3^2e1^2}} \\ \dfrac{2e2e0^2 - 2e2e1^2 - 2e2^3 - 2e2e3^2 - 4e3e0e1}{e0^4 + 2e0^2e1^2 - 2e0^2e2^2 + 2e0^2e3^2 + e1^4 + 2e1^2e2^2 - 2e3^2e1^2 + e2^4 + 2e2^2e3^2 + e3^4 + 8e0e2e3e1} \end{bmatrix} \qquad \text{eq. 4-95.}$$

$$\frac{\partial h_{\psi,\,\theta,\,\phi}(x)}{\partial e_2} = \begin{bmatrix} \dfrac{2e3e0^2 - 2e3e1^2 + 2e3e2^2 + 2e3^3 + 4e2e0e1}{e0^4 + 2e0^2e1^2 - 2e0^2e2^2 + 2e0^2e3^2 + e1^4 + 2e1^2e2^2 - 2e3^2e1^2 + e2^4 + 2e2^2e3^2 + e3^4 + 8e0e2e3e1} \\ \dfrac{2e0}{\sqrt{1 - 4e0^2e2^2 + 8e0e2e3e1 - 4e3^2e1^2}} \\ \dfrac{2e1e0^2 + 2e1^3 + 2e1e2^2 + -2e1e3^2 + 4e2e0e3}{e0^4 + 2e0^2e1^2 - 2e0^2e2^2 + 2e0^2e3^2 + e1^4 + 2e1^2e2^2 - 2e3^2e1^2 + e2^4 + 2e2^2e3^2 + e3^4 + 8e0e2e3e1} \end{bmatrix} \qquad \text{eq. 4-96.}$$

$$\frac{\partial h_{\psi,\,\theta,\,\phi}(x)}{\partial e_3} = \begin{bmatrix} \dfrac{2e2e0^2 - 2e2e1^2 - 2e2^3 - 2e2e3^2 - 4e3e0e1}{e0^4 + 2e0^2e1^2 - 2e0^2e2^2 + 2e0^2e3^2 + e1^4 + 2e1^2e2^2 - 2e3^2e1^2 + e2^4 + 2e2^2e3^2 + e3^4 + 8e0e2e3e1} \\ \dfrac{-2e1}{\sqrt{1 - 4e0^2e2^2 + 8e0e2e3e1 - 4e3^2e1^2}} \\ \dfrac{2e0^3 + 2e0e1^2 - 2e0e2^2 + 2e0e3^2 + 4e3e1e2}{e0^4 + 2e0^2e1^2 - 2e0^2e2^2 + 2e0^2e3^2 + e1^4 + 2e1^2e2^2 - 2e3^2e1^2 + e2^4 + 2e2^2e3^2 + e3^4 + 8e0e2e3e1} \end{bmatrix} \qquad \text{eq. 4-97.}$$

The control matrix *G* has the form:

$$G = \frac{\partial f}{\partial u} = \left[\frac{\partial f}{\partial a_x}\ \frac{\partial f}{\partial a_y}\ \frac{\partial f}{\partial a_z}\ \frac{\partial f}{\partial P}\ \frac{\partial f}{\partial Q}\ \frac{\partial f}{\partial R}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1W & V \\ 0 & 0 & 0 & W & 0 & -1U \\ 0 & 0 & 0 & -1V & U & 0 \\ 0 & 0 & 0 & \frac{-1}{2}e1 & \frac{-1}{2}e2 & \frac{-1}{2}e3 \\ 0 & 0 & 0 & \frac{1}{2}e0 & \frac{-1}{2}e3 & \frac{1}{2}e2 \\ 0 & 0 & 0 & \frac{1}{2}e3 & \frac{1}{2}e0 & \frac{-1}{2}e1 \\ 0 & 0 & 0 & \frac{-1}{2}e2 & \frac{1}{2}e1 & \frac{1}{2}e0 \end{bmatrix} \qquad \text{eq. 4-98.}$$

which results from the simulation.

4.4.3 Alternative Implementations and Improvements for Kalman Filters

4.4.3.1 Checking the covariance symmetry and the positive definiteness

The error covariance matrix $P = E\langle \delta x, \delta x^T \rangle$ is, by its definition symmetric. However, numerical errors can result in a representation of P that becomes nonsymmetric. This affects the performance and stability of the Kalman filter.

Square-root filtering is designed to ensure that the covariance matrix of estimation uncertainty remains symmetric and positive definite. Otherwise, the fidelity of the solution of the Riccati equation can degrade to the point that it corrupts the Kalman gain, and that in turn corrupts the estimate. The implementation of square-root filtering is more complex and computationally more expensive because it involves replacing the covariance matrix P with its Cholesky factor (see Chapter A.1).

If we do not choose square-root filtering, then we need some assurance that the algorithm is numerically stable. It was shown[90] that asymmetry of P is one of the factors contributing to numerical instability of the Riccati equation. The covariance matrix can be symmetrized occasionally by adding it to its transpose and rescaling it:

$$P = \frac{1}{2}(P + P^T) \qquad \text{eq. 4-99.}$$

The error covariance matrix $P = E\langle \delta x, \delta x^T \rangle$ is, also by definition positive semidefinite. Computed values of P can lose this positive semidefinite property. When this occurs, the Kalman filter gains for corresponding states have the wrong sign, and the state may temporarily diverge. Even if the sign eventually corrects itself, subsequent performance will suffer, since the covariance matrix is no longer accurate.

Any symmetric matrix, in particular P, can be factored as:

$$P = UDU^T \qquad \text{eq. 4-100.}$$

Special-purpose algorithms have been derived that propagate the factor U and D instead of P itself. The factors contains the same information as P, the factorized algorithm can be shown to be theoretically equivalent to the Kalman filter, and the algorithm automatically maintains both the positive semidefiniteness and the symmetry of the covariance matrix.

4.4.3.2 Serial measurement processing

It is shown in [90], that it is more efficient to process the components of a measurement vector serially, one component at a time, than to process them as a vector. This may seem counterintuitive, but it is true even if its implementation requires a transformation of measurement variables to make the associated measurement noise covariance R a diagonal matrix, which means that noise is uncorrelated from one component to another.

It is possible to make the measurements uncorrelated. If the covariance matrix R of the measurement noise is not a diagonal matrix, then it can be made so by UDU^T decomposition and changing the measurement variables,

$$\begin{aligned} R_{corr} &= U_R D_R U_R^T \\ R_{decorr} &= D_R \\ y_{decorr} &= U_R / y_{corr} \\ H_{decorr} &= U_R / H_{corr} \end{aligned} \qquad \text{eq. 4-101.}$$

where R_{corr} is the nondiagonal measurement noise covariance matrix, and the new measurement vector y_{decorr} has a diagonal measurement noise covariance matrix R_{decorr} and measurement sensitivity matrix H_{decorr}.

The components of y_{decorr} can now be processed one component at a time using the corresponding row of H_{decorr} as its measurement sensitivity matrix and the corresponding diagonal element of R_{decorr} as its measurement noise variance. The implementation of serial measurement update is presented in the following code:

```
x=x_k[-]
P=P_k[-]
for j=1:l
        y=y_k(j);
        H=H_k(j,:);
        R=R_decorr(j,j);
        K=PH'/(HPH'+R);
        x=K(y-Hx);
        P=P-KHP;
end;
x_k[+] = x;
P_k[+] = (P+P')/2;
```

Figure 4-14. Serial measurement update implementation

The last line from this code is a symmetrizing procedure designed to improve robustness.

4.4.3.3 A separate bias Kalman filter

In the application of Kalman recursive filtering techniques, an accurate model of the process dynamics and observation is required. In our scenario, this model contains parameters which may deviate by constant but unknown amounts from their nominal values. Using nominal values of the parameters in the filter design may lead to large errors in the estimate provided by the filter. It is a

common practice to augment the state vector of the original problem by adding additional components to represent the uncertainty parameters. The filter then estimates the bias terms as well as in the original problem. But when the bias terms are comparable to the number of state variables of the original problem, the new state vector has a substantially higher dimension than the original problem, and the computations required by the filtering algorithm may become excessive. In our particular situation when the algorithm runs on the LART board, the filter does not run in real time and the bias estimates have unacceptably large jumps. One solution is to apply the Friedland's separate bias formulation [91], which will replace a plethora of 6x6 multiplications and one 6x6 inversion by a somewhat larger number of 3x3 multiplications and one 3x3 inversion.

The linear error equation (see Equation 4-79 on page 73) has been written in a form in which the error gyro biases are assumed constant, thus permitting the direct application of the results of Friedland's separate bias Kalman filter.

Switching to Friedland's notation, we define the error state vector $x_k = \delta\theta(t_k)$ and a bias state error vector $b_k = \Delta b(t_k)$, where t_k is the time at the k^{th} iteration of the Kalman filter algorithm. By adjoining b and x we obtain an augmented state vector $z_k = \begin{bmatrix} x_k & b_k \end{bmatrix}^T$, which satisfies the following equation:

$$z_k = F_k z_k + \begin{bmatrix} I \\ 0 \end{bmatrix} w_k \qquad F_k = \begin{bmatrix} A_k & B_k \\ 0 & I \end{bmatrix} \qquad \text{eq. 4-102.}$$

The additive white noise w_k, with variance Q_k, only effects x, since b is assumed constant. The measurement equation is:

$$y_k = L_k z_k + v_k \qquad \text{eq. 4-103.}$$

where v_k is white noise with variance R_k. In Friedland's paper, $L_k = \begin{bmatrix} H_k & C_k \end{bmatrix}$, but in our application the measurement from inclinometers and compass only measure x and not b, so $C_k = 0$. This fact will be used in the following formulas and will result in a simplification of Friedland's formulation.

Now applying a Kalman filter to this model, the optimal estimate of the state z is:

$$\begin{aligned} \hat{z}_{k+1} &= F_k \hat{z}_k + K(k+1)(y_{k+1} - L_k F \hat{z}_k) \\ K(k+1) &= P(k) L^T [L P(k) L^T + R_k]^{-1} \end{aligned} \qquad \text{eq. 4-104.}$$

The Riccati equations for the recursive computation of the estimation error covariance matrix $P(k)$ needed in the Kalman gain equation can be rolled together into a single predictor, the predictor covariance update equation:

$$\left.\begin{aligned} P(k) &= (I - K(k)H)P(k) \\ P(k+1) &= F_k P(k) F_k^T + G Q_{k+1} G^T \end{aligned}\right\}$$

$$P(k+1) = F_k [I - K(k)H] P(k) F_k^T + \begin{bmatrix} I \\ 0 \end{bmatrix} Q_{k+1} \begin{bmatrix} I & 0 \end{bmatrix} \qquad \text{eq. 4-105.}$$

To proceed further we partition matrix $P(k)$ into 3x3 matrices as:

$$P(k) = \begin{bmatrix} P_x(k) & P_{xb}(k) \\ P_{xb}^T(k) & P_b(k) \end{bmatrix} \qquad \text{eq. 4-106.}$$

In this way the expression for Kalman gain, see Equation 4-104, may be rewritten in partitioned form as:

$$K(k) = P(k)\begin{bmatrix} H & 0 \end{bmatrix}^T \left[\begin{bmatrix} H & 0 \end{bmatrix} P(k) \begin{bmatrix} H & 0 \end{bmatrix}^T + R_k \right]^{-1} \qquad \text{eq. 4-107.}$$

$$= \begin{bmatrix} P_x(k) & H^T \\ P_{xb}^T(k) & H^T \end{bmatrix} \left[\begin{bmatrix} HP_x(k) & HP_{xb}(k) \end{bmatrix} \begin{bmatrix} H^T \\ 0 \end{bmatrix} + R_k \right]^{-1}$$

$$= \begin{bmatrix} P_x(k)H^T[HP_x(k)H^T + R_k]^{-1} \\ P_{xb}^T(k)H^T[HP_x(k)H^T + R_k]^{-1} \end{bmatrix} = \begin{bmatrix} K_x(k) \\ K_b(k) \end{bmatrix}$$

We have split the Kalman gain because K_x is used for estimating x and K_b for estimating b. To compute these two Kalman gains, K_x and K_b, covariance submatrices P_x and P_{xb} are needed. These are updated by the partition version of Equation 4-105, and also for further simplification we introduce $H = I$:

$$\begin{bmatrix} P_x(k+1) & P_{xb}(k+1) \\ P_{xb}^T(k+1) & P_b(k+1) \end{bmatrix} = \begin{bmatrix} A_k & B_k \\ 0 & I \end{bmatrix} \left(\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} - \begin{bmatrix} K_x \\ K_b \end{bmatrix} \begin{bmatrix} H & 0 \end{bmatrix} \right) \begin{bmatrix} P_x & P_{xb} \\ P_{xb}^T & P_b \end{bmatrix} \begin{bmatrix} A_k & B_k \\ 0 & I \end{bmatrix}^T + \begin{bmatrix} Q_k & 0 \\ 0 & 0 \end{bmatrix} \qquad \text{eq. 4-108.}$$

$$= \begin{bmatrix} A_k - A_kK_x - B_kK_b & B_k \\ -K_b & I \end{bmatrix} \begin{bmatrix} P_xA_k^T + P_{xb}B_k^T & P_{xb} \\ P_{xb}^TA_k^T + P_bB_k^T & P_b \end{bmatrix} + \begin{bmatrix} Q_k & 0 \\ 0 & 0 \end{bmatrix}$$

We considered $H = I$ because the inclinometer and compass measurements are pre-processed inside the TCM2 magnetometer board to give direct measurements of the Euler angles. The values for P_x, P_b, P_{xb} are computed from Equation 4-108 and we obtain the expression:

$$T_1 = A_k - A_kK_x - B_kK_b \qquad T_2 = P_xA_k^T + P_{xb}B_k^T \qquad T_3 = P_{xb}^TA_k^T + P_bB_k^T$$

$$\begin{aligned} P_b(k+1) &= P_b - K_bP_{xb} \\ P_{xb}(k+1) &= T_1P_{xb} + B_kP_b \\ P_x(k+1) &= T_1T_2 + B_kT_3 \end{aligned} \qquad \text{eq. 4-109.}$$

After simplification the fastest possible code for measurement updates is:

```
            T1 = ( Px + R );
            T1.invert();
      //Kx = ( Px * H3.transpose() ) / ( H3* Px * H3.transpose() + Rx );
            Kx =  Px  * T1;
      //Kb = (Pxb.transpose()*H3.transpose())/(H3*Px*H3.transpose()+Rx);
            T3 = Pxb.transpose();
            Kb =  T3 * T1;
      //My derivation based on Friedland paper
            T1 = A - A*Kx - B*Kb;
            T2 = Pb - Kb*Pxb;
            T3 = B*T3;
            Pxb = T1*Pxb + B*Pb;
            Pb = T2;
            Px = Pxb*B.transpose()+(T1*Px + T3)*A.transpose() + Q3;
```

and for time update:

```
            T1 = A*Pxb + B*Pb;
            Px=T1*B.transpose()+(A*Px+B*Pxb.transpose()*A.transpose();
            Pxb = T1;
```

Figure 4-15. Code for Separate Bias Kalman Filter

The timing evaluation of both implementations, using 100000 runs, reveals that the speed approximately doubled, from 890 μs to 550 μs on a PIII 500 MHz. The implementation on the LART embedded system is running almost in real time, at about 80 Hz. In order to make the system run in real time, the Kalman filter should be implemented using integer arithmetic.

4.5 Results and Simulation

The Euler angle and quaternion models are used to implement the two extended Kalman filters. For analysis of the output, the error metric used is root-sum-square (RSS) of the difference between true and estimated value of bias.

Estimates of bias error will be compared by examining how fast the error converges below certain percentages of bias. The simulations run with different bias values showing that both Euler angle and quaternion EKF are asymptotically stable.

The results of the Kalman filter simulation for gyro bias error:

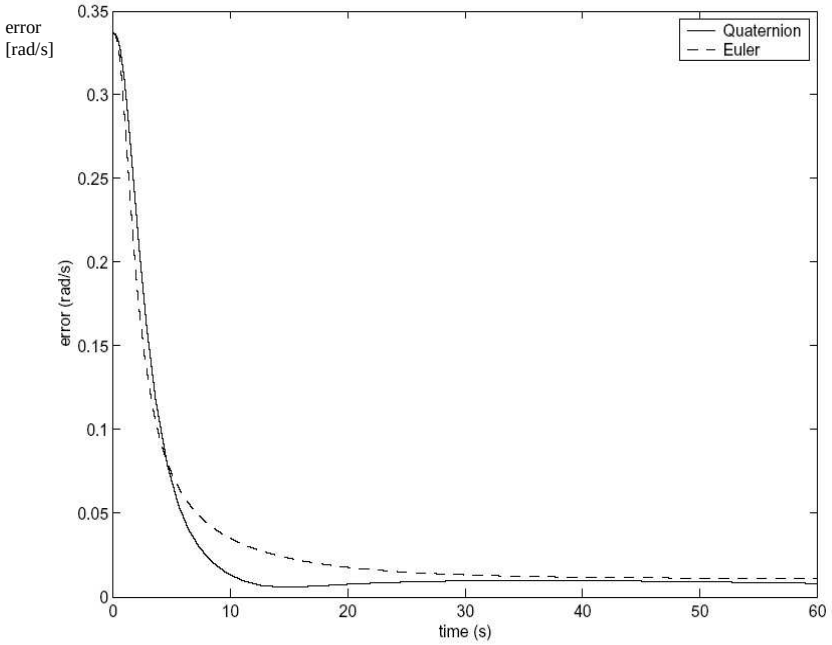

Figure 4-16. Comparison of bias RSS error between Quaternion and Euler angle EKF

The simulation results for Euler angle and quaternion EKF show that the two implementations are comparable, as expected. The quaternion implementation appears to be performing better than Euler angle implementation. Quaternion implementation converges to 1% from the maximum error range in 6.4s compared with 8.6s for the Euler angle. The steady state bias error for quaternion EKF is also slightly lower than the steady state for Euler angle EKF.

The same quaternion EKF runs on LART system, operating under Linux. The implementation on the LART embedded system is running at about 80 Hz. Results showed that the orientation Kalman filter converged in less then 1 sec. The stability of the gyros provided accurate orientation values during normal head motion. The overall accuracy is about 0.2 degrees.

4.6 Conclusions

Aided strapdown systems are receiving renewed interest in applications requiring high accuracy and high rate outputs, while also being inexpensive, small, and low power. A much more critical cause of error in position measurement is error in the orientation determined by the gyros. An error of $d\theta$ in tilt angle will result in an error of $1g \cdot \sin(d\theta)$ in the horizontal components of the acceleration calculated by the navigation computer. In practice, it is the gyroscopes, not the accelerometers which limit the positional navigation accuracy of most INS systems, since the effects of gyroscopic tilt error will soon overtake any small accelerometer biases.

A great difficulty in all attitude estimation approaches that use gyros is the low frequency noise component, also referred to as bias or drift, that violates the white noise assumption required for standard Kalman filtering. This problem has attracted the interest of many researchers since the early days of the space program. Inclusion of the gyro noise model in a Kalman filter by suitably augmenting the state vector has the potential to provide estimates of the sensor bias when the observability requirement is satisfied. An estimate of the attitude would imply the derivation of the dynamics of the moving body, which we wish to avoid because of the complexity. In order to do

so we relate the gyro output signal to the bias and the angular velocity of the vehicle, using the simple and realistic model.

In this chapter we decompose the localization problem into attitude estimation and, subsequently, position estimation. We focus on obtaining a good attitude estimate without building a model of the vehicle dynamics. The dynamic model was replaced by gyro modeling. An Indirect (error state) Kalman filter that optimally incorporates inertial navigation and absolute measurements was developed for this purpose. The linear form of the system and measurement equations for the planar case derived here allowed us to examine the role of the Kalman filter as a signal processing unit. The extension of this formulation to the 3D case shows the same benefits. A tracking example in the 3D case was also shown in this chapter.

In order to extract the actual body acceleration during motion, the local projection of the gravitational acceleration vector has to be subtracted from the accelerometer signals. Even small errors in the attitude estimate $\hat{q}$ can cause significant errors in the calculation of the actual body accelerations. This is more prevalent during slow motions with small body accelerations, such as the ones used in this experiment. The estimated velocities and positions through the integration of the IMU are susceptible to large errors due to the magnitude of the gravitational acceleration compared to the minute body accelerations that the vehicle experiences during its actual motion.

Another advantage is that the filter is very efficient with computer memory requirements. Everything it needs to know about the initial conditions and all the past measurements and motion are contained in the covariance matrix Pk. An outstanding benefit of the Kalman filter is that it is very flexible about timing and the order in which it receives and processes measurements. There is no requirement for periodic time updates or measurement updates. In a typical run-time implementation, the KF program will have a main loop that processes time updates at a high update rate, and slightly interrupts this cycle to squeeze in a measurement update whenever one becomes available from a sensor. This enables flexible integration of data from disparate sensors that are not even synchronized.

The advantage of quaternion representation is that since the incremental quaternion corresponds very closely to a small rotation, the first component will be close to unity and thus the attitude information of interest is contained in the three vector components of the quaternion. The equations used to update the Euler rotations of the body with respect to the chosen reference frame are implemented using a 4th order Runge-Kuta integration algorithm, since a closed form solution does not exist. However, their use is limited since the solutions of $\dot{\phi}$ and $\dot{\psi}$ become undetermined when $\theta = \pm 90^{o}$. By comparison, quaternion update equations do have a closed form solution and do not have singularities.

The quaternion implementation appears to be performing better than the Euler angle implementation. Quaternion implementation converged to 1% from the maximum error range faster than the Euler angle. The steady state bias error for quaternion EKF is also slightly lower than the steady state for Euler angle EKF.

Quaternion EKF implementation in real time represents another issue. After the filter implementation was proven to be stable and running on an Intel PC platform, the next step was to reduce filter computation complexity, but try to maintain filter performance. A separate bias filter formulation was implemented and run on the LART platform almost in real time, providing an update rate of about 80Hz.

Chapter 5

Vision-based Pose Tracking

5.1 Introduction

The pose tracking system for outdoor augmented reality, encompasses the following subsystems:

- A DGPS system that tracks the human position within meters, heading within steps of 45 degrees, and update rates of within a minute
- A Vision system that tracks the head position within centimeters, the orientation within degrees, and an update rate of within a second.
- An Inertial Tracking system that tracks the head within millimeters and its orientation within tenths of degrees and an update rate of within 10 msec.

This chapter describes the algorithms that are necessary to obtain a robust vision system for tracking the motion of a camera based on its observations of the physical world. We will look at feature detection algorithms, camera calibration issues, and pose determination algorithms. Traditional approaches to tracking planar targets are based on finding correspondences in successive images. This can be achieved by computing optical flow or by matching a sparse collection of features [67].

5.2 Feature Extraction

Our tracking system is based on the real-time (video-rate) tracking of features, such as corners, line segments and circles / ellipses in the image. From these features, pose determination algorithms can calculate the motion of the camera.

5.2.1 Corner Detection

Intuitively, a corner is a location in the image where, locally, the intensity varies rapidly in both X and Y directions. The gradients of the image I_x and I_y measure the variation of intensity. If we look at the typical distribution of I_x and I_y in a small window when it is a corner we have large variations in both directions.

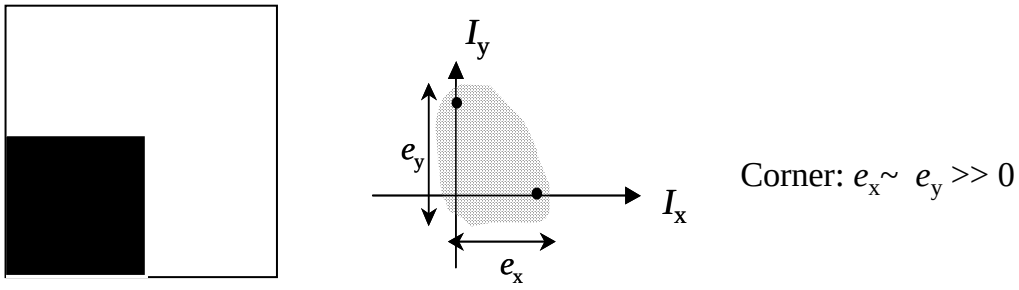

Figure 5-1. Corner feature characteristics

For an ideal corner with sides parallel to the *x* and *y* axis, some of the gradients are (I_x, 0) (for the part of the corner on the vertical edge) (0,I_y) (the horizontal edge) and (0,0) in the uniform regions of the window. More realistically, the values will be spread out in between those three points. The key observation is that the set of (I_x,I_y) values is elongated in both directions.

The corners are selected as basic primitives for the calibration and pose recovery processes for two reasons:

- *Simplicity:* The corners encode particular geometric relationships, thus matching them involves less ambiguity than matching individual image features, since there are fewer possible alternatives and more information to be able to judge a match. In addition, there are fewer corners than image features.
- *Robustness:* Corners are less likely to be completely occluded as opposed to line features. Due to the target geometry, the corners are relatively stable against partial occlusions. Even if the corner point is occluded, its position can be found by intersecting the corresponding line segments. Also, the position of a corner is generally measured with more precision than the position of an isolated feature.

Sobel first derivative operators are used to take the derivatives *x* and *y* of an image, after which a small region of interest is defined for corner detection. A 2x2 matrix of the sums of the derivatives *x* and *y* is created as follows:

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \qquad \text{eq. 5-1.}$$

The eigenvalues of C are found. For the 2x2 matrix of the equation above, the solutions may be written in a closed form:

$$\lambda = \frac{\sum I_x^2 + \sum I_y^2 \pm \sqrt{(\sum I_x^2 + \sum I_y^2)^2 - 4(\sum I_x^2 \sum I_y^2 - (\sum I_x \sum I_y)^2)}}{2} \qquad \text{eq. 5-2.}$$

If λ_1, $\lambda_2 > t$, where *t* is some threshold, then a corner is found at that location. The threshold, *t*, can be estimated from the histogram of λ_2, as the latter has often an obvious valley near zero.

Foerstner's [42] corner detection is based on the same gradient method. It uses the eigenvalues of the same matrix *C*. If the two eigenvalues of the matrix *C* are large, then there is an important change of gray level. The trace of this matrix can be used to select image pixels which correspond to image features.

$$tr(C) = \lambda_1 + \lambda_2 \qquad \text{eq. 5-3.}$$

If *tr(C)* is higher than a threshold, the pixel is considered a corner candidate pixel. Next, the interest points have to be distinguished from pixels belonging to edges. This can be done using the ratio $v = \lambda_2/\lambda_1$ which describes the degree of orientation or isotropy. $\lambda_2 = 0$ indicates a straight edge, while $v = 1$, thus $\lambda_2 = \lambda_1$ indicates a gradient isotropy caused by a corner or a blob. The corner response function is given by:

$$q = \frac{4 \cdot det(C)}{tr^2(C)} = \frac{4 \cdot \lambda_1 \cdot \lambda_2}{(\lambda_1 + \lambda_2)^2} \qquad \text{eq. 5-4.}$$

This value has to be larger than a threshold for a pixel representing a corner. Sub-pixel precision is achieved through a quadratic approximation (fit second order polynomial to 9 points). To avoid

corners due to noise, the images can be smoothed with a Gaussian kernel. But instead of doing this smoothing on original images, it is done on the images of the squared image derivatives contained in matrix *C*. Figure 5-2 shows the corners found in two example images. We can see that our feature points include image corners and T-junctions generated by the intersection of object contours, but also corners of the local intensity pattern not corresponding to obvious scene features.

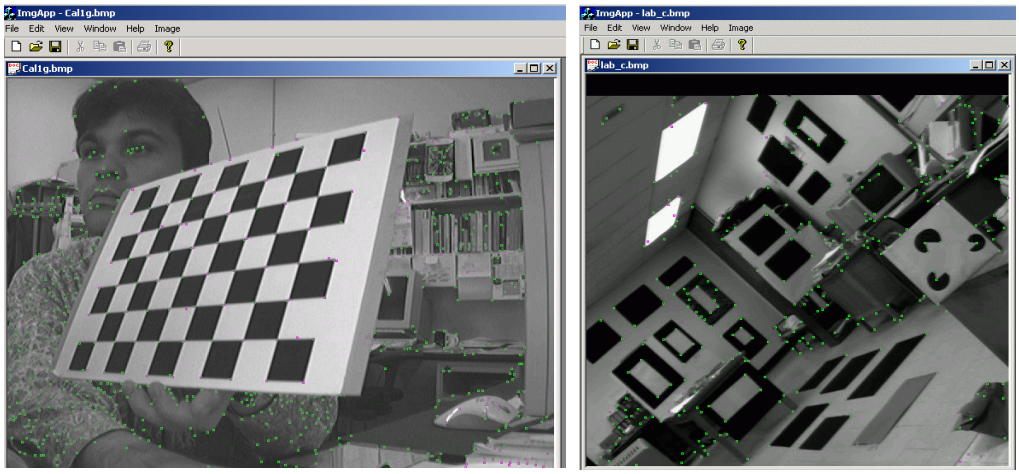

Figure 5-2. Corner detection examples

5.2.2 Target Recognition

A target can be used to track fiducials in an image. A fiducial is a known pattern with a known position in the world. Fiducials are used in the printing industry, and printed circuit board and VLSI production to align layers. A bar or dot code can be considered as a fiducial when used to identify objects and their position. In our case, we assume a target is based on a pattern of squares, which, for instance, can be used for calibration purposes. See Figure 5-2. The target recognition algorithm we used is derived from an algorithm implemented in the Intel OpenCV library, it was enhanced in order to gain robustness. The algorithm works as follow. A video frame from the camera is passed to the function *findTarget*(), which looks for a target pattern. The number of points to look for, and an array in which found point coordinates are stored, are passed as parameters to this function. The way in which *findTarget*() finds the intersections is a bit indirect: it looks for the square, performs an opening operation on them and calculates their intersection using corner detection. Figure 5-3 shows an extreme example of the results of this function. The actual library function only requests that a single iteration is done, so the squares are only slightly opened. Note that the squares maintain their shape after this operation; they are smaller, but still squares.

The image is then passed to a threshold function and subsequently processed by an outline detector from the library. It finds the outlines around connected blocks of pixels such as the quadrangles (projected squares) that we are looking for. The *findTarget*() function is based on the generation of a list of all the contours that can be approximated by a polygon (using the Douglas-Peucker[97] method, see Chapter A.6.2), and discard objects that do not have four vertices, are too small, or just do not look like a projected square. Anything square enough is added to a list.

The function loops through all the found squares and matches up all their intersecting corners. The list of intersecting corners is sorted. Before sorting, we first determine the 4 extreme corners. Based on these points and the 4 corresponding model points we compute the homography matrix *H*. We use the *H* matrix to reproject all the model points in the image and select the points closest to each corresponding detected corner. The *findTarget*() function is able to find the corner intersection points to a precision of a single pixel. If a good set of intersections were found, the number of inter-

sections is returned as a positive value. Otherwise, the number of intersections found is returned as a negative number.

If the function returns a positive value, further refinement of the corner locations can be performed using another function of the CV library, the *cvFindCornerSubPix*() call, which is able to do sub-pixel location of the intersections to a precision of 0.1 pixel. It appeared however that the refined corners were sometimes several pixels away from the true corners. It seemed that as the angles between the edges of adjacent corners approach 90 degrees, the accuracy got worse. At exactly 90 degrees the function is off completely; the refined corners can be located several pixels away from the true corner. When we used the function to do a static camera calibration, we checked the sub-pixel locations visually by plotting the corners in the image. When they were wrong the intrinsic calibration gave bizarre results. Hence, we implemented a new subpixel corner detector algorithm.

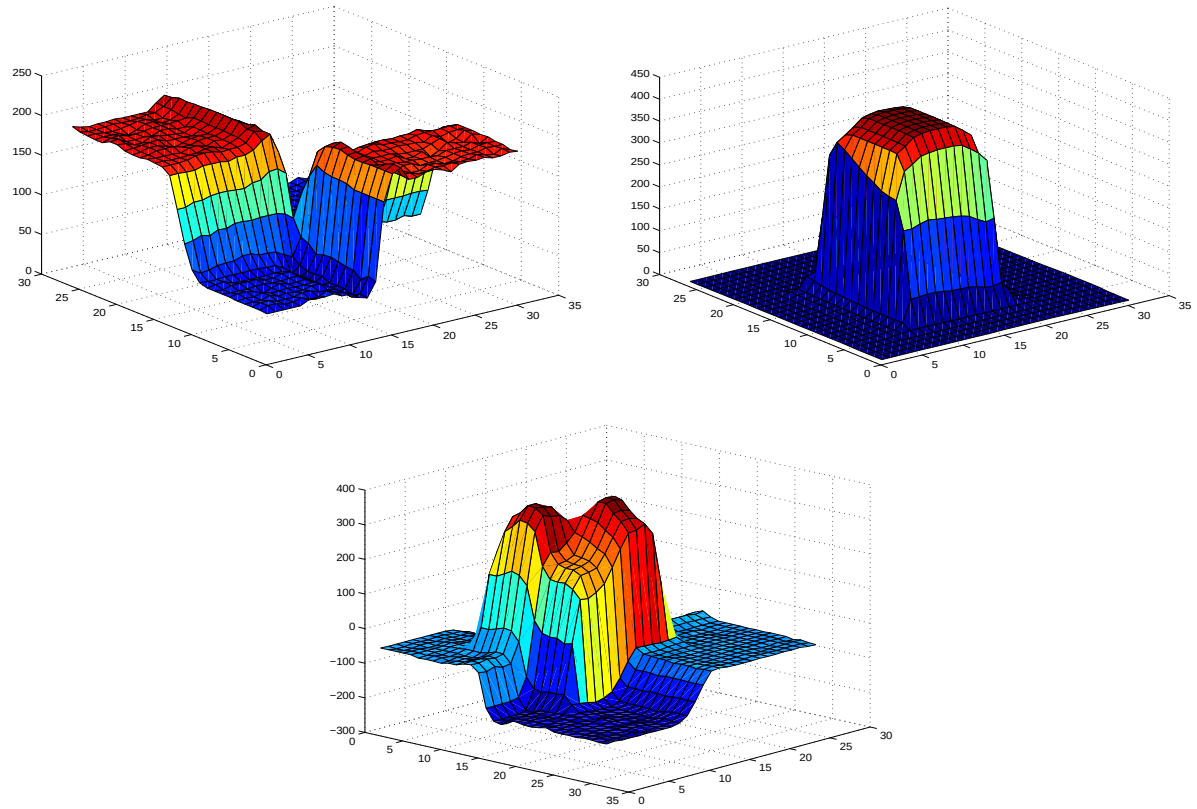

Figure 5-3. Subpixel corner detection

Corners remain relatively stable across a sequence of images in a video. For this reason, a number of corner finding algorithms for tracking purposes have been proposed in the literature. The most popular is the Harris corner finder that computes a corner strength λ_p for each pixel p in a search window W of a grayscale video image. This is determined using a symmetric 2x2 matrix:

$$C_p = \begin{bmatrix} \sum_Q E_x^2 & \sum_Q E_x E_y \\ \sum_Q E_x E_y & \sum_Q E_y^2 \end{bmatrix} \qquad \text{eq. 5-5.}$$

where Q is a $(2N + 1)$x$(2N + 1)$ neighborhood of pixels around p (we use N=3), and E_x and E_y are respectively the x and y spatial image gradients around p using Sobel edge filters. Geometrically, the two eigenvectors of C_p define edge directions at pixel p, and the two eigenvalues λ_1 and λ_2 define the edge strengths. A strong corner is thus denoted by two large eigenvalues, where $\lambda_1 > \lambda_2$. A point is a corner if the smaller of the two eigenvalues $\lambda_p = \lambda_2$ is larger than some predetermined corner strength threshold λ_T.

Note that λ_T is dependent on the size of the chosen neighborhood of p. Therefore, a histogram analysis of the λ_2 values of an entire image can be used to determine a suitable threshold. The best threshold for corners will be located to the right of a large peak in the histogram. Note that this should not be performed for each frame, since the corner finding algorithm is computationally expensive ($O(N^3)$) when performed on an entire image. For our system, a user-defined threshold was used. Since the corner finder computes a λ_2 value for each pixel in W, the strongest corner will be denoted by the pixel with the largest value. The problem with this method of corner finding is the lack of subpixel precision on the computed corner coordinate. This could lead to corner features exhibiting full pixel jitter from frame to frame, which could affect homography stability in subsequent stages of the tracking system.

It has been demonstrated[57] that the approximate location of a grid point can be estimated by fitting a quadratic surface through the points in second eigenvalue image space. Subsequently, the maximum (or minimum) can be calculated analytically from the fitted surface. The pseudocode for the subpixel corner detection is:

```
% fit a 2nd order polynomial to 9 points
% using 9 pixels centered on irow,jcol
u = f(ypeak-1:ypeak+1, xpeak-1:xpeak+1);
u = u(:);
xy = [-1 -1; -1 0; -1 1; 0 -1; 0 0; 0 1; 1 -1; 1 0; 1 1];
x = xy(:,1);
y = xy(:,2);
% u(x,y) = A(1) + A(2)*x + A(3)*y + A(4)*x*y + A(5)*x^2 + A(6)*y^2
X = [ones(9,1),  x,  y,  x.*y,  x.^2,  y.^2];
% u = X*A
A = X\u;
% get absolute maximum, where du/dx = du/dy = 0
x_offset = ( 2*A(2)*A(6) - A(3)*A(4) ) / ( A(4)*A(4) - 4*A(5)*A(6) );
y_offset = -( A(2) + 2*A(5)*x_offset ) / A(4);
% return only one-tenth of a pixel precision
x_offset = round(10*x_offset)/10;
y_offset = round(10*y_offset)/10;
```

Figure 5-4. Subpixel corner detection using a quadratic surface fit

A faster method for finding a subpixel corner is to use λ_2. Non-maximal suppression is first used to locate the strongest corner position $s = (x, y)$ in the pixel neighborhood. A subpixel location is then computed based on weighting the corner strengths of the 4-connected neighbors as follows:

$$x_{subpixel} = x + 0{,}5 + \frac{(\lambda_a - \lambda_b)}{2(\lambda_b - 2\lambda_s + \lambda_a)} \qquad y_{subpixel} = y + 0{,}5 + \frac{(\lambda_c - \lambda_d)}{2(\lambda_d - 2\lambda_s + \lambda_c)} \quad \text{eq. 5-6.}$$

where λ_i represents the corner strength of pixel *i* in Figure 5-5.

	c	
a	s	b
	d	

Figure 5-5. Four-connected neighborhood for a corner pixel s.

The *findTarget*() function works on the original, grayscale image, and the initial guesses of the intersection pointers. For each point, the function iterates until the image gradient at the subpixel intersection location is below a threshold. The measured precision of this subpixel corner location is better than 0.1 pixel [62].

5.3 Camera Calibration

5.3.1 Changing Coordinate System

The purpose of this section is to describe how the perspective projection matrix *P* varies when we change the retinal plane and world coordinate systems, which will reveal more of the interesting structure of the matrix. This information is important since, in practical applications, these changes of coordinates systems occur quite often.

5.3.1.1 Changing coordinates in the retinal plane

Let us consider the effect of changing the origin of the image coordinate system and the units on the *u* and *v* axes on the matrix *P*. These units are determined by the number of sensitive cells for CCD and CMOS cameras.

The corresponding situation is shown in Figure 5-6. We go from the old coordinate system to the new coordinate system, which is centered at a point c_n in the image. For a pixel *m* we have:

$$c_n m = c_n c + cm \qquad \text{eq. 5-7.}$$

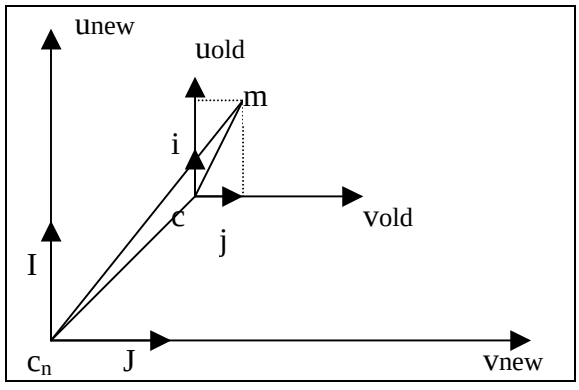

Figure 5-6. The intrinsic parameters and normalized camera

Writing $cm = u_{old} i + v_{old} j$ in the old coordinate system and introducing the scaling from the old coordinate system (*i*, *j*) to the new (*I*, *J*), we have:

$$i = sI \qquad j = sJ$$
$$s = \begin{bmatrix} k_u & 0 \\ 0 & k_v \end{bmatrix} \qquad \text{eq. 5-8.}$$

We can denote $c_n c$ by t in the new coordinate system, and this allows us to rewrite Equation 5-7 in projective coordinates as:

$$\tilde{m}_{new} = \tilde{H} \bullet \tilde{m}_{old}$$
$$\tilde{H} = \begin{bmatrix} s & t \\ 0^T{}_2 & 1 \end{bmatrix} \qquad 0^T{}_2 = \begin{bmatrix} 0 & 0 \end{bmatrix} \qquad \text{eq. 5-9.}$$

Note that matrix H defines a *collineation* of the retinal plane considered as a projective plane. This collineation preserves the line at infinity and is therefore an affine transformation. Since we have $\tilde{m}_{old} = \tilde{P}_{old}\tilde{M}$, we conclude that $\tilde{m}_{new} = \tilde{H}\tilde{P}_{old}\tilde{M}$, and thus:

$$\tilde{P}_{new} = \tilde{H}\tilde{P}_{old} \qquad \text{eq. 5-10.}$$

An $(n+1)$x$(n+1)$ matrix A, such that det(A) is different from 0, defines a linear transformation or a *collineation* from the projective space, P^n, onto itself. The matrix associated with a given collineation is defined up to a nonzero scale factor.

If we denote the coordinates of $\boldsymbol{t}$ by u_0 and v_0, then the most general matrix $\tilde{P}$, when the world reference frame is the standard coordinate system of camera (Equation A-70), can be written as (where f is the focal length of the optical system,):

$$\tilde{P} = \begin{bmatrix} -fk_u & 0 & u_0 & 0 \\ 0 & -fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \text{eq. 5-11.}$$

Let $\alpha_u = -fk_u$ and $\alpha_v = -fk_v$. The parameters α_u, α_v, u_0 and v_0 do not depend on the position and orientation of the camera in space, and they are thus called *intrinsic*.

We now define a special coordinate system that allows us to normalize the retinal coordinates. This coordinate system is called the *normalized coordinate system* of the camera, and it is widely used in motion and stereo applications because it allows us to ignore the specific characteristics of the cameras and to think in terms of ideal system. Given a matrix $\tilde{P}$, we can change the retinal coordinate system so that matrix $\tilde{P}$ and $\tilde{H}$ can be written:

$$\tilde{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\tilde{H} = \begin{bmatrix} \frac{1}{\alpha_u} & 0 & -\frac{u_0}{\alpha_u} \\ 0 & \frac{1}{\alpha_v} & -\frac{v_0}{\alpha_v} \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{eq. 5-12.}$$

Therefore, according to Equation 5-9, the new retinal coordinates are given by:

$$u' = \frac{u - u_0}{\alpha_u} \qquad v' = \frac{v - v_0}{\alpha_v} \qquad \text{eq. 5-13.}$$

If we consider the plane parallel to the retinal plane and at a unit distance from the optical center (Chapter A.5.1), this plane, together with the optical center, defines a normalized camera (see Figure 5-7), and represents the geometric interpretation of a normalized camera.

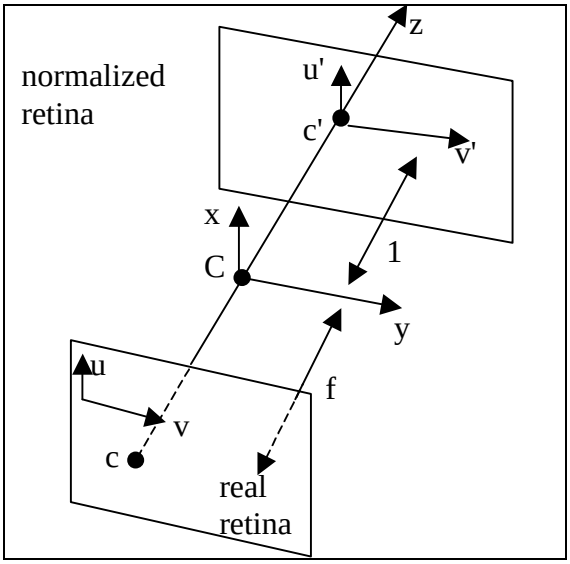

Figure 5-7. Relationship between the real and normalized retinal plane

5.3.1.2 The use of intrinsic parameters

Knowledge of the intrinsic parameters allows us to perform metric measurements with a camera, to compute the angle between rays C_m and C_n determined by two pixels *m* and *n*. The easiest way to see this is to consider the absolute conic Ω. The absolute conic could be seen as an imaginary circle located in the plane at infinity. This conic plays the role of a calibration pattern because its image ω is independent of the camera's position and orientation and depends only upon the intrinsic parameters of the camera.

Let the camera (*C*, *R*) undergo a rigid motion *D*, a combination of rotation and translation; *C* is the optical center and *R* the retinal plane.

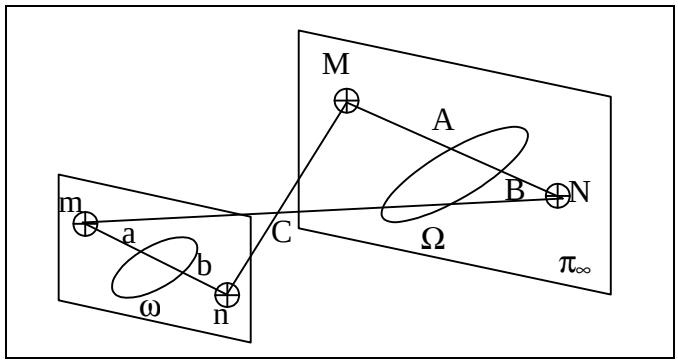

Figure 5-8. How to compute the angle between optical rays <C,m> and <C,n> using the image of the absolute conic

The equation of Ω is [47]:

$$X^2 + Y^2 + Z^2 = 0 = T \qquad (or \quad M^T M = 0) \qquad \text{eq. 5-14.}$$

The image $\tilde{m}$ of a point $\tilde{M}$ of Ω satisfies the equation:

$$\tilde{m} = \tilde{P}\tilde{M} = PM$$

$$\tilde{M} = \begin{bmatrix} M \\ 0 \end{bmatrix} \qquad \text{eq. 5-15.}$$

$$M = P^{-1}\tilde{m}$$

where P is the leftmost 3x3 submatrix of $\tilde{P}$. According to Equation 5-14, M^TM=0, the equation of ω is:

$$\tilde{m}^T P^{-1T} P^{-1} \tilde{m} = 0 \qquad \text{eq. 5-16.}$$

Using Equation 5-11, this is found by simple computation to be equivalent to:

$$\tilde{m}^T \left[\begin{array}{cc:c} \frac{1}{\alpha_u^2} & 0 & -\frac{u_0}{\alpha_u^2} \\ 0 & \frac{1}{\alpha_v^2} & -\frac{v_0}{\alpha_v^2} \\ \hdashline -\frac{u_0}{\alpha_u^2} & -\frac{v_0}{\alpha_v^2} & 1 + \frac{u_0^2}{\alpha_u^2} + \frac{v_0^2}{\alpha_v^2} \end{array}\right] \tilde{m} = 0 \qquad \text{eq. 5-17.}$$

Going to pixel coordinates, this can be rewritten as:

$$\left(\frac{u - u_0}{\alpha_u}\right)^2 + \left(\frac{v - v_0}{\alpha_v}\right)^2 + 1 = 0 \qquad \text{eq. 5-18.}$$

This equation shows that ω contains the intrinsic parameters.

5.3.1.3 Changing the world reference frame

Just as it is important to study how matrix $\tilde{P}$ changes when we change the image coordinate system, it is likewise important for many applications to see how matrix $\tilde{P}$ varies when we change the 3D coordinate system.

If we go from the old coordinate system centered at the optical center C to the new coordinate system centered at O by a rotation R followed by a translation T, in projective coordinates this will be:

$$\tilde{M_{old}} = \tilde{K}\tilde{M_{new}}$$
$$\tilde{K} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ O_3^T & 1 \end{bmatrix} \qquad \text{eq. 5-19.}$$

Matrix $\tilde{K}$ represents a collineation that preserves the plane at infinity and the absolute conic. The matrices R and T describe the position and orientation of the camera with respect to the new world coordinate system. They are called the *extrinsic* parameters of the camera. Using Equation 5-19 we have:

$$\tilde{m} = \tilde{P}_{old}\tilde{M}_{old}$$
$$\tilde{m} = \tilde{P}_{old}\tilde{K}\tilde{M}_{new} \qquad \text{eq. 5-20.}$$

Therefore we have:

$$\tilde{P}_{new} = \tilde{P}_{old}\tilde{K} \qquad \text{eq. 5-21.}$$

This tells us how the perspective projection matrix $\tilde{P}$ changes when we change coordinate systems in three-dimensional space. If we now combine Equation 5-10 and Equation 5-21, we obtain the more general equation:

$$\tilde{P}_{new1} = \tilde{H}\tilde{P}_{old}\tilde{K} \qquad \text{eq. 5-22.}$$

5.3.2 Direct Parameter Calibration and the Tsai Algorithm

Consider a 3D point P, defined by its coordinates $[X_w, Y_w, Z_w]^T$ in the world reference frame. Let $[X_c, Y_c, Z_c]^T$ be the coordinate of P in a camera reference frame (with $Z_c>0$ if P is visible). As usual, the origin of the camera frame is the center of projection, and Z is the optical axis.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R}\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{T}$$

5.3.2.1 Camera Parameters from the Projection Matrix

The method consists of two sequential stages:

1. Estimate the projective matrix linking world and image coordinates
2. Compute the camera parameters as closed-form functions of the entries of the projective matrix

Estimation of the Projection Matrix

The relation between a 3D point in space and its projection in the 2D image is given by:

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X_i^w \\ Y_i^w \\ Z_i^w \\ 1 \end{bmatrix} \qquad \text{eq. 5-23.}$$

with:

$$x = \frac{u_i}{w_i} = \frac{m_{11}X_i^w + m_{12}Y_i^w + m_{13}Z_i^w + m_{14}}{m_{31}X_i^w + m_{32}Y_i^w + m_{33}Z_i^w + m_{34}}$$
$$y = \frac{v_i}{w_i} = \frac{m_{21}X_i^w + m_{22}Y_i^w + m_{23}Z_i^w + m_{24}}{m_{31}X_i^w + m_{32}Y_i^w + m_{33}Z_i^w + m_{34}} \qquad \text{eq. 5-24.}$$

The matrix M is defined up to an arbitrary scale factor and has therefore only 11 independent entries, which can be determined through a homogeneous system of linear equations for at least 6 world-image point matches. However, through the use of calibration patterns like in Figure 5-9:

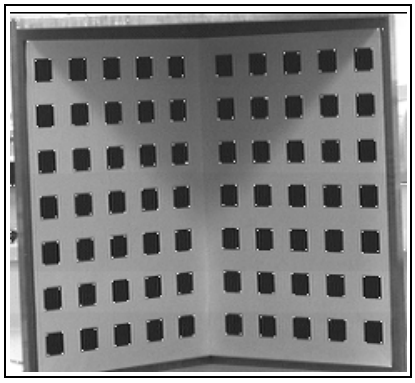

Figure 5-9. Camera Calibration Target

many more correspondences and equations can be obtained and M can be estimated through least squares techniques. For N matches we have the homogeneous linear system: $\mathbf{Am} = 0$, with:

$$A = \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -x_NX_N & -x_NY_N & -x_NZ_N & -x_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -y_NX_N & -y_NY_N & -y_NZ_N & -y_N \end{bmatrix} \qquad \text{eq. 5-25.}$$

and $\mathbf{m} = [m_{11}, m_{12}, m_{13}, \ldots, m_{33}, m_{34}]$.

Since A has rank 11, the vector m can be recovered using SVD-related techniques as the column of V corresponding to the zero (in practice the smallest) singular value of A, with $A = UDV^T$. This means that the entries of M are obtained up to an unknown scale factor [48].

Computing Camera Parameters

We now want to express the intrinsic and extrinsic camera parameters as a function of the estimated projection matrix. Since M is recovered up to a scale factor in the previous approach, the matrix obtained from above through SVD will be noted $\overline{\mathbf{M}}$.

$$\mathbf{M} = \begin{bmatrix} -f_x r_{11} + o_x r_{31} & -f_x r_{12} + o_x r_{32} & -f_x r_{13} + o_x r_{33} & -f_x T_x + o_x T_z \\ -f_y r_{21} + o_y r_{31} & -f_y r_{22} + o_y r_{32} & -f_y r_{23} + o_y r_{33} & -f_y T_y + o_y T_z \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} \quad \text{eq. 5-26.}$$

where o_x, o_y are the coordinates of the image center, f_x, f_y represents the focal length in the x and y directions, and r and T represent the elements of the rotation matrix and translation vector respectively.

In what follows we also need the 3D vectors:

$$\begin{aligned} \mathbf{q}_1 &= [\overline{m}_{11}, \overline{m}_{12}, \overline{m}_{13}]^T \\ \mathbf{q}_2 &= [\overline{m}_{21}, \overline{m}_{22}, \overline{m}_{23}]^T \\ \mathbf{q}_3 &= [\overline{m}_{31}, \overline{m}_{32}, \overline{m}_{33}]^T \\ \mathbf{q}_4 &= [\overline{m}_{14}, \overline{m}_{24}, \overline{m}_{34}]^T \end{aligned} \quad \text{eq. 5-27.}$$

Since M is defined up to a scale factor we can write $\overline{M} = \gamma M$. The absolute value of the scale factor, $|\gamma|$, can be obtained by observing that $\boldsymbol{q}_3$ is the last row of the rotation matrix R. Hence (R is normalized: the squares of the elements in any row or column have the sum of 1),

$$\sqrt{\overline{m}_{31}^2 + \overline{m}_{32}^2 + \overline{m}_{33}^2} = |\gamma| \sqrt{r_{31}^2 + r_{32}^2 + r_{33}^2} = |\gamma| \quad \text{eq. 5-28.}$$

We now divide each entry of the matrix $\overline{\mathbf{M}}$ by $|\gamma|$, and observe that the resulting, normalized projection matrix differs from M by at most a sign change. Then we can recover all intrinsic and extrinsic parameters:

$$
\begin{aligned}
T_z &= \sigma\overline{m}_{34} \\
r_{3i} &= \sigma\overline{m}_{34} \\
o_x &= \mathbf{q}_1^{\,T}\mathbf{q}_3 \\
o_x &= \mathbf{q}_2^{\,T}\mathbf{q}_3 \\
f_x &= \sqrt{\mathbf{q}_1^{\,T}\mathbf{q}_1 - o_x^{\,2}} \\
f_y &= \sqrt{\mathbf{q}_2^{\,T}\mathbf{q}_2 - o_y^{\,2}} \\
r_{1i} &= \frac{\sigma(o_x\overline{m}_{3i} - \overline{m}_{1i})}{f_x} \\
r_{2i} &= \frac{\sigma(o_y\overline{m}_{3i} - \overline{m}_{2i})}{f_y} \\
T_x &= \frac{\sigma(o_xT_z - \overline{m}_{14})}{f_x} \\
T_y &= \frac{\sigma(o_yT_z - \overline{m}_{24})}{f_y}
\end{aligned}
$$

eq. 5-29.

As usual, the estimated rotation matrix is not orthogonal, and we can find the closest orthogonal matrix (in the sense of the Frobenius norm [48]) as follows: $\bar{R} = UDV^T \Rightarrow R = UIV^T$. Using SVD decomposition and since the three singular values of a 3x3 orthogonal matrix are one, we can simply replace *D* by *I*, so that the resulting matrix is exactly orthogonal.

We are left to discuss how to determine the sign σ. The sign σ can be obtained from $T_z = \sigma\overline{m}_{34}$, because we know whether the origin of the world reference frame is in front of ($T_z > 0$) or behind ($T_z < 0$) the camera.

Estimating the Image Center

Vanishing points: Let L_i, $i = 1,..,N$ be parallel lines in 3D space, and $\boldsymbol{l}_i$ the corresponding image lines. Due to the perspective projection, the line L_i appears to meet in a point *p*, called the vanishing point, defined as the common intersection of all the image lines $\boldsymbol{l}_i$.

Orthocenter Theorem [48]: Let *T* be the triangle on the image plane defined by the three vanishing points of the mutually orthogonal sets of parallel lines in space. The image center is the orthocenter of *T*.

What is important is that this theorem reduces the problem of locating the image center to one of intersecting image lines, which can be easily created using a suitable calibration pattern.

The algorithm runs as follows:

1. Compute the three vanishing points [45] *p*1, *p*2 and *p*3, determined by three bundles of lines obtained using a Hough transform.
2. Compute the orthocenter *O*, of the triangle *p*1*p*2*p*3.

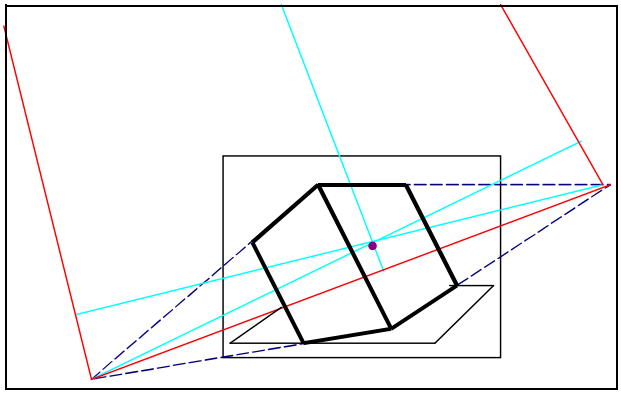

Figure 5-10. Image Center estimation

5.3.2.2 The Tsai Camera Calibration Algorithm

Under the assumption that only a radial distortion occurs in the camera, Tsai [65] proposed a two-stage algorithm to calibrate the camera. Assuming a pixel in the image is only distorted along the radial direction, the radial alignment constraint (RAC) can be employed to make the calibration process linear. This algorithm has been well tested and documented. For the purposes of this thesis, we will only touch on the two stages of the algorithm:

1. Compute the 3D pose: R, T (only x and y are calculated) and the scale factors s_x and s_y;
2. Compute the effective focal length f, radial distortion coefficient kl, and the z component of T.

With these two stages, the whole computation becomes linear and therefore, this algorithm is very efficient and has been adopted extensively in the computer vision field.

Whereas this algorithm requires that the scale factor and the position of the principle point are known in advance, we tried several ways to calculate the scale factor. Moreover, the RAC model requires that the angle of incidence between the optical axis of the camera and the calibration plane is at least 30°. It can be observed that once the rotational matrix is known, the remaining extrinsic and intrinsic parameters can be easily obtained or updated, following the same thought as Tsai, starting from a pose estimation.

5.3.3 Camera Calibration by Viewing a Plane in Unknown Orientations

Camera calibration is a necessary step in 3D computer vision in order to extract metric information from 2D images. Much work has been done, starting in the photogrammetry community ([44], [46] to cite a few) and more recently in the computer vision community ([49],[50],[51],[52], [65], [68], [66] to cite a few). We can classify these techniques roughly into two categories: photogrammetric calibration and self-calibration.

- *Photogrammetric calibration.* Camera calibration is performed by observing a calibration object whose geometry in 3D space is known with very good precision. Calibration can be done very efficiently[47]. The calibration object usually consists of two or three planes orthogonal to each other. Sometimes, a plane undergoing a precisely known translation is also used. These approaches require an expensive calibration apparatus and an elaborate setup [65].
- *Self-calibration.* Techniques in this category do not use any calibration objects. Just by moving a camera in a static scene, the rigidity of the scene provides in general two constraints [17, 15] on the camera's internal parameters from one camera displacement by using image information alone. Therefore, if images are taken by the same camera with fixed internal parameters, correspondences between three images are sufficient to recover both the internal and external

parameters which allow us to reconstruct 3D structure up to a similarity [16, 13]. While this approach is very flexible, it is not yet mature [43]. As there are many parameters to estimate, one cannot always obtain reliable results.

Our proposed technique only requires that a camera observes a planar pattern shown in at least two different orientations. This pattern can be printed on a laser printer and attached to a reasonable planar surface (e.g., a hard book cover). Either the camera or the planar pattern can be moved by hand. The motion does not need to be known. The proposed approach lies between the photogrammetric calibration and self-calibration, because we use 2D metric information rather than 3D or a purely implicit one. Both computer simulation and real data have been used to test the proposed technique, and very good results have been obtained. Compared with conventional techniques, the proposed technique is considerably more flexible. Compared with self-calibration, it acquires a considerable degree of robustness.

5.3.3.1 Basic Equations

We examine the constraints on the camera's intrinsic parameters provided by observing a single plane. A 2D point is denoted by $m = [u, v]^T$. A 3D point is denoted by $M = [X, Y, Z]^T$. We use $\tilde{x}$ to denote the augmented vector by adding 1 as the last element $\tilde{m} = [u, v, 1]^T$ and $M = [X, Y, Z, 1]^T$. A camera is modeled by the usual pinhole model: the relationship between a 3D point M and its image projection m is given by:

$$s\tilde{m} = \mathbf{A}[\mathbf{R} \quad \mathbf{t}]\tilde{M} \qquad \text{eq. 5-30.}$$

where s is an arbitrary scale factor, (R, t), called the extrinsic parameters, are the rotation and translation which relate the world coordinate system to the camera coordinate system, and A, called the camera intrinsic matrix, is given by:

$$\mathbf{A} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{eq. 5-31.}$$

with (u_0, v_0) the coordinates of the principal point, and α and β the scale factors in image u and v axes, and γ the parameter describing the skew of the two image axes. We use the abbreviation A^{-T} for $(A^{-1})^T$ or $(A^T)^{-1}$.

5.3.3.2 The homography between the model plane and its image

Without loss of generality, we assume the model plane is on $Z = 0$ of the world coordinate system. Let us denote the i^{th} column of the rotation matrix R by r_i. From Equation 5-30, we have:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3 \quad \mathbf{t}]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = A[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}]\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \qquad \text{eq. 5-32.}$$

By abuse of notation, we still use M to denote a point on the model plane, but $M = [X, Y]^T$ since Z is always equal to 0. In turn, $\tilde{M} = [X, Y, 1]^T$. Therefore, a model point M and its image m is related by a homography $\boldsymbol{H}$:

$$s\tilde{m} = \mathbf{H}\tilde{M} \quad with \quad \mathbf{H} = A[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] \qquad \text{eq. 5-33.}$$

As is clear, the 3x3 matrix $\boldsymbol{H}$ is defined up to a scale factor.

Let $x = [\mathbf{h}_1^T, \mathbf{h}_2^T, \mathbf{h}_3^T]^2$. Then Equation 5-33 can be rewritten as [48]:

$$\begin{bmatrix} \tilde{M}^T & 0^T & -u\tilde{M}^T \\ 0^T & \tilde{M} & -v\tilde{M}^T \end{bmatrix} \mathbf{x} = 0 \qquad \text{eq. 5-34.}$$

When we are given n points, we have n of these equations, which can be written in a matrix equation as $Lx = 0$, where L is a $2n$x9 matrix. As x is defined up to a scale factor, the solution is well known to be the right singular vector of L associated with the smallest singular value (or equivalently, the eigenvector of L^TL associated with the smallest eigenvalue).

In L some elements are constant 1, some are in pixels, some are in world coordinates, and some are multiplications of both. This makes L numerically poorly conditioned. A much better result can be obtained by performing data normalization.

The reasons for data normalization will be described in Section 5.3.3.3 on page 107. Here we give only the steps and formulas required at each step:

1. The points are translated so that the centroid point is at (0,0)

$$a = \frac{1}{N}\sum_{i=1}^{N} u_i \qquad b = \frac{1}{N}\sum_{i=1}^{N} v_i \qquad \text{eq. 5-35.}$$

2. The coordinate points are scaled so that the mean distance from the points to the center (centroid) is 1. The scale factor is given by:

$$s = \frac{1}{N}\sum_{i=1}^{N} \sqrt{(u_i - u_c)^2 + (v_i - v_c)^2} \qquad \text{eq. 5-36.}$$

In Equation 5-36 we denote with u, v the normalized points coordinates, and u_1, v_1 unscaled ones. Then $u = (u_1 - a)/s, v = (v_1 - b)/s$. We also denote with m_{ij} the elements of the homography matrix $\bar{m}$. Then the coordinates u_1,v_1 have the expression:

$$u_1 = \frac{(a \cdot m_{31} + m_{11} \cdot s)X_w + (a \cdot m_{32} + m_{12} \cdot s)Y_w + a \cdot m_{33} + m_{13} \cdot s}{m_{31}X_w + m_{32}Y_w + m_{33}}$$
$$v_1 = \frac{(b \cdot m_{31} + m_{21} \cdot s)X_w + (b \cdot m_{32} + m_{22} \cdot s)Y_w + b \cdot m_{33} + m_{23} \cdot s}{m_{31}X_w + m_{32}Y_w + m_{33}} \qquad \text{eq. 5-37.}$$

If now we use the normalized points u, v to compute the homography $\bar{m}$, then the resulting homography for the points u_1,v_1 will be:

$$\begin{bmatrix} (a \cdot m_{31} + m_{11} \cdot s) & (a \cdot m_{32} + m_{12} \cdot s) & a \cdot m_{33} + m_{13} \cdot s \\ (b \cdot m_{31} + m_{21} \cdot s) & (b \cdot m_{32} + m_{22} \cdot s) & b \cdot m_{33} + m_{23} \cdot s \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \qquad \text{eq. 5-38.}$$

The solution x (Equation 5-34) can be considered as an initial guess or input for an algorithm based on the maximum likelihood criterion. Let M_i and m_i be the model and image points, respectively. Ideally, they should satisfy Equation 5-33. In practice, they do not because of noise in the extracted image points. Let us assume that m_i is corrupted by Gaussian noise with mean 0 and covariance matrix Λ_{m_i}. Then, the maximum likelihood estimation of **H** is obtained by minimizing the following functional:

$$\sum_i (m_i - \hat{m}_i)\Lambda_{m_i}(m_i - \hat{m}_i) \qquad \text{eq. 5-39.}$$

where

$$\hat{m}_i = \frac{1}{\mathbf{h}_3^T M_i}\begin{bmatrix}\mathbf{h}_1^T M_i \\ \mathbf{h}_2^T M_i\end{bmatrix} \quad with\ \mathbf{h}_i\ the\ i^{th} row\ of\ \mathbf{H}$$

In practice, we simply assume $\Lambda_{m_i} = \sigma^2\mathbf{I}$ for all i. This is reasonable if points are extracted independently with the same procedure. In this case, the above problem becomes a nonlinear least-squares problem: $min_{\mathbf{H}}\sum\|m_i - \hat{m}_i\|^2$. The nonlinear minimization is executed with the Levenberg-Marquard Algorithm as implemented in Minpack [59].

5.3.3.3 Data Normalization

Image coordinates are sometimes given with the origin at the top left of the image, and sometimes with the origin at the center. The question immediately occurs whether this makes a difference to the results of the homography estimation algorithm. More generally, to what extent is the result of the homography computation algorithm dependent on the choice of coordinates in the image. Suppose, for instance, that the image coordinates were changed by some affine or even projective transformation before running the algorithm. This makes a difference.

Condition of the system of equations

The linear method consists in finding the least eigenvector of the matrix A^TA. This may be done by expressing A^TA as a product UDU^T where U is orthogonal and D is diagonal. We assume that the diagonal entries of D are in non-increasing order. In this case, the least eigenvector of A^TA is the last column of U. Denote by k the ratio d_1/d_8 (recalling that A^TA is a 9 x 9 matrix). The parameter k is the condition number of the matrix A^TA, well known to be an important factor in the analysis of stability of linear problems. Its relevance to the problem of finding the least eigenvector is briefly explained next.

The bottom right-hand 2x2 block of matrix D is of the form:

$$\begin{bmatrix} d_8 & 0 \\ 0 & 0 \end{bmatrix}$$

assuming that $d_9 = 0$ because a homography has 8 degrees of freedom, which ideally will be the case. Now, suppose that this block is perturbed by the addition of noise to become:

$$\begin{bmatrix} d_8 & \varepsilon \\ \varepsilon & 0 \end{bmatrix}$$

In order to restore this matrix to diagonal form we need to multiply left and right by V^T and V, where V is a rotation through an angle $\theta = (1/2)\text{atan}(2\varepsilon/d_8)$. If ε is of the same order of

magnitude as d_8 then this is a significant rotation. Looking at the full matrix, $A^TA = UDU^T$, we see that the perturbed matrix will be written in the form $U\bar{V}D'\bar{V}^TU^T$ where:

$$\bar{V} = \begin{bmatrix} I_{7\times 7} & 0 \\ 0 & V \end{bmatrix}$$

Multiplying by $\bar{V}$ replaces the last column of U by a combination of the last two columns. Since the last column of U is the least eigenvector of the matrix, this perturbation will drastically alter the least eigenvector of the matrix A^TA. Thus, changes to A^TA of the order of magnitude of the eigenvalue d_8 cause significant changes to the least eigenvector. Since multiplication by an orthonormal matrix does not change the Frobenius norm of a matrix, we see that:

$$\|A^TA\| = \left(\sum_{i=1}^{9} d_i^2 \right)^{1/2} \qquad \text{eq. 5-40.}$$

If the ratio $k = d_1/d_8$ is very large, then d_8 represents a very small part of the Frobenius norm of the matrix. A perturbation of the order of d_8 will therefore cause a very small relative change to the matrix A^TA, while at the same time causing a very significant change to the least eigenvector. Since A^TA is written directly in terms of the coordinates of the points $u \leftrightarrow u'$, we see that if k is large, then very small changes to the data can cause large changes to the solution. This is obviously very undesirable. The sensitivity of invariant subspaces is discussed in greater detail in [53], where more specific conditions for the sensitivity of invariant subspaces are given.

We now consider how the condition number of the matrix A^TA may be made small. We consider two sorts of transformation, translation and scaling. These methods will be given only an intuitive justification, since a complete analysis of the condition number of the matrix is too complex to undertake here.

Normalizing Transformations

The previous sections concerned the condition number of the matrix A^TA indicating that it is desirable to apply a transformation to the coordinates before carrying out camera calibration algorithm. It was applied with success before carrying out the eight-point algorithm for finding the fundamental matrix [55].

This normalization has been implemented as a prior step in the homography estimation with good results. The condition number of the linear equation for homography computation without normalization is 10^{15}. The same number with normalization is 10^7. The improvement is approximately 10^8.

The effect of normalization is related to the condition number of the set of linear equations. For exact data and infinite precision arithmetic the result will be independent of the normalizing transformation. However, in the presence of noise, the solution will diverge from correct result. The effect of a large condition number is to amplify this divergence. This is true even for infinite precision arithmetic - this is not a round-off error effect [54].

Isotropic Scaling

As a first step, the coordinates in each image are translated (by a different translation for each image) so as to bring the centroid of the set of all points to the origin. The coordinates are also scaled. In the discussion of scaling, it was suggested that the best results will be obtained if the coordinates are scaled, so that, on average, a point u is of the form $u = (u, v, w)^T$, with each of u, v, and w having the same average magnitude. Rather than choose different scale factors for each

point, an isotropic scaling factor is chosen so that the u and v coordinates of a point are scaled equally. To this end, we choose to scale the coordinates so that the average distance of a point u from the origin is equal to $\sqrt{2}$. This means that the "average" point $\bar{m}$ is equal to $(1, 1, 1)^T$.

In summary, the transformation is as follows:

1. The points are translated so that their centroid is at the origin.
2. The points are then scaled so that the average distance from the origin is equal to $\sqrt{2}$.
3. This transformation is applied to each of the two images independently.

5.3.3.4 Constraints on the intrinsic parameters

Given an image of the model plane, a homography can be estimated. Let us denote it by $H = [h_1\ h_2\ h_3]$. From Equation 5-33 on page 106, we have:

$$[\mathbf{h}_1\ \mathbf{h}_2\ \mathbf{h}_2] = \lambda A[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] \qquad \text{eq. 5-41.}$$

where λ is an arbitrary scalar. Using the knowledge that $\mathbf{r}_1$ and $\mathbf{r}_2$ are orthonormal, we have:

$$\begin{aligned} \mathbf{h}_1^T A^{-T} A^{-1} \mathbf{h}_2 &= 0 \\ \mathbf{h}_1^T A^{-T} A^{-1} \mathbf{h}_1 &= \mathbf{h}_2^T A^{-T} A^{-1} \mathbf{h}_2 \end{aligned} \qquad \text{eq. 5-42.}$$

These are the two basic constraints on the intrinsic parameters, given one homography. Because a homography has 8 degrees of freedom and there are 6 extrinsic parameters (3 for rotation and 3 for translation), we can only obtain 2 constraints on the intrinsic parameters [56]. Note that $A^{-T}A^{-1}$ actually describes the image of the absolute conic.

5.3.3.5 Solving Camera Calibration

First, let us see how the matrix $A^{-T}A^{-1}$ it looks in terms of intrinsic parameters:

$$B = A^{-T}A^{-1} = \begin{bmatrix} \frac{1}{\alpha^2} & \frac{-\gamma}{\alpha^2\beta} & \frac{\gamma v_0 - u_0\beta}{\alpha^2\beta} \\ \frac{-\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & \frac{-\gamma(\gamma v_0 - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ \frac{\gamma v_0 - u_0\beta}{\alpha^2\beta} & \frac{-\gamma(\gamma v_0 - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(\gamma v_0 - u_0\beta)^2}{\alpha^2\beta^2} + \frac{v_0^2}{\beta^2} + \lambda \end{bmatrix} \equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \qquad \text{eq. 5-43.}$$

Note that **B** is symmetric, and can be defined by a 6D vector:

$$\mathbf{b} = \begin{bmatrix} B_{11} & B_{12} & B_{22} & B_{13} & B_{23} & B_{33} \end{bmatrix} \qquad \text{eq. 5-44.}$$

If we note with h_i the ith column vector of the homography H, $h_i = [h_{i1}, h_{i2}, h_{i3}]^T$, then we have:

$$\mathbf{h}_i^T B \mathbf{h}_i = \begin{bmatrix} h_{i1} & h_{i2} & h_{i3} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \begin{bmatrix} h_{j1} \\ h_{j2} \\ h_{j3} \end{bmatrix} = \mathbf{v}_{ij}^T \mathbf{b} \qquad \text{eq. 5-45.}$$

where the vector *v* has the expression:

$$\mathbf{v}_{ij}^T = \qquad \text{eq. 5-46.}$$

$$\left[h_{i1} \cdot h_{j1} \;\; h_{i1} \cdot h_{j2} + h_{i2} \cdot h_{j1} \;\; h_{i2} \cdot h_{j2} \;\; h_{i3} \cdot h_{j1} + h_{i1} \cdot h_{j3} \;\; h_{i3} \cdot h_{j2} + h_{i2} \cdot h_{j3} \;\; h_{i3} \cdot h_{j3}\right]$$

Using the above notations, the two fundamental constraints on Equation 5-42 on page 109, from a given homography, Equation 5-45 can be rewritten as two homogeneous equations in **b**:

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \mathbf{b} = 0 \qquad \text{eq. 5-47.}$$

If *n* images of the model plane are observed, by stacking *n* such equations as Equation 5-47, we have:

$$\begin{aligned}
v_{2 \cdot i,1} &= h_{11,i} \cdot h_{22,i} & v_{2 \cdot i+1,1} &= h_{11,i} \cdot h_{11,i} - h_{21,i} \cdot h_{21,i} \\
v_{2 \cdot i,2} &= h_{12,i} \cdot h_{21,i} + h_{11,i} \cdot h_{22,i} & v_{2 \cdot i+1,1} &= h_{12,i} \cdot h_{11,i} - h_{22,i} \cdot h_{21,i} \\
v_{2 \cdot i,3} &= h_{12,i} \cdot h_{22,i} & v_{2 \cdot i+1,1} &= h_{12,i} \cdot h_{12,i} - h_{22,i} \cdot h_{22,i} \\
v_{2 \cdot i,4} &= h_{13,i} \cdot h_{21,i} + h_{11,i} \cdot h_{23,i} & v_{2 \cdot i+1,1} &= 2h_{13,i} \cdot h_{11,i} - h_{23,i} \cdot h_{21,i} \\
v_{2 \cdot i,5} &= h_{13,i} \cdot h_{22,i} + h_{12,i} \cdot h_{23,i} & v_{2 \cdot i+1,1} &= 2h_{12,i} \cdot h_{13,i} - h_{23,i} \cdot h_{22,i} \\
v_{2 \cdot i,6} &= h_{13,i} \cdot h_{23,i} & v_{2 \cdot i+1,1} &= h_{13,i} \cdot h_{13,i} - h_{23,i} \cdot h_{23,i}
\end{aligned} \qquad \text{eq. 5-48.}$$

$$\mathbf{Vb} = 0 \qquad \text{eq. 5-49.}$$

where *V* is a 2*n* x 6 matrix. If *n* =3, we will have in general a unique solution **b** defined up to a scale factor. If *n* = 2, we can impose the skewless constraint $\gamma = 0$ in order to be able to solve the homogeneous equation. This can be done by adding [0, 1, 0, 0, 0, 0] *b* = 0 as an additional equation to Equation 5-49. If *n* = 1, we can only solve two camera intrinsic parameters (that is α and β), by assuming that u_0 and v_0 are known (set at the image center) and $\gamma = 0$. The solution to Equation 5-49 is well known as the eigenvector of V^TV associated to the smallest eigenvalue (equivalently, the right singular vector of *V* associated with the smallest singular value).

Once **b** is estimated, we can compute the intrinsic camera matrix *A*. Knowing **b** we can recompose the matrix *B*. The matrix *B* is estimated only up to a scale factor, $B = \lambda \mathbf{A}^{-\mathbf{T}}\mathbf{A}$ with λ being an arbitrary scale. Solving the system of equations, we can uniquely extract the intrinsic parameters from matrix *B*.

$$\begin{aligned}
v_0 &= (B_{12} \cdot B_{13} - B_{11} \cdot B_{23}) / (B_{11} \cdot B_{22} - B_{12}^2) \\
\lambda &= B_{33} - [B_{13}^2 + v_0 \cdot (B_{12} \cdot B_{13} - B_{11} \cdot B_{23})] / B_{11} \\
\alpha &= \sqrt{\lambda / B_{11}} \\
\beta &= \sqrt{\lambda B_{11} / (B_{11} \cdot B_{22} - B_{12}^2)} \\
\gamma &= -B_{12} \cdot \alpha^2 \beta / \lambda \\
u_0 &= \gamma v_0 / \alpha - B_{13} \cdot \alpha^2 / \lambda
\end{aligned} \qquad \text{eq. 5-50.}$$

Once A is known, the extrinsic parameters for each image can be computed. From Equation 5-41 on page 109, we have [48]:

$$\begin{aligned} \lambda &= 1/\left\|A^{-1}\mathbf{h}_1\right\| = 1/\left\|A^{-1}\mathbf{h}_2\right\| \\ \mathbf{r}_1 &= \lambda A^{-1}\mathbf{h}_1 \\ \mathbf{r}_2 &= \lambda A^{-1}\mathbf{h}_2 \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} &= \lambda A^{-1}\mathbf{h}_3 \end{aligned} \qquad \text{eq. 5-51.}$$

In general, due to the noise in the data, the computed rotation matrix does not normally satisfy the properties of a rotation matrix. The rotation matrix can be orthogonalized as before. We can find the closest orthogonal matrix as follows: $\bar{R} = UDV^T \Rightarrow R = UIV^T$.

This approach also has degenerated configurations, i.e. configurations in which additional images do not provide more constraints on the camera intrinsic parameters. Because Equation 5-42 on page 109 was derived from the properties of the rotation matrix, if **Ri (**rotation matrix for view i**)** is not independent of **Rj**, then image 2 does not provide additional constraints. In particular, if a plane undergoes a pure translation, then **Ri = Rj** and image 2 is not helpful for camera calibration.

Definition of the extrinsic parameters

Consider the calibration grid *#i* (attached to the i^{th} calibration image), and concentrate on the camera reference frame attached to that grid. Without loss of generality, take $i = 1$. The following figure shows the reference frame (O, X, Y, Z) attached to that calibration grid:

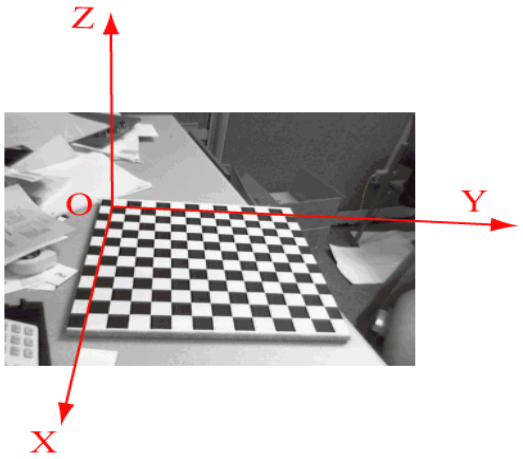

Figure 5-11. Reference frame attached to calibration grid

Let A be a point space of coordinate vector $p_A = [X, Y, Z]$ in the grid reference frame (reference frame shown on the Figure 5-11). Let $p_c = [X_c, Y_c, Z_c]$ be the coordinate vector of A in the camera reference frame. Then p_A and p_c are related to each other through the following rigid motion equation:

$$p_c = R_{c1} \cdot p_A + T_{c1} \qquad \text{eq. 5-52.}$$

In particular, the translation vector T_{c1} is the coordinate vector of the origin of the grid pattern (O) in the camera reference frame, and the third column of the matrix R_{c1} is the surface normal vector of the plane containing the planar grid in the camera reference frame.

5.3.3.6 Lens Distortion

Any camera usually exhibits significant lens distortion, especially radial distortion. The distortion is described by four coefficients: two radial distortion coefficients k_1, k_2, and two tangential ones p_1, p_2. The radial distortion is caused by the fact that objects at different angular distances from the lens axis undergo different magnifications. The tangential distortion is due to the fact that optical centers of multiple lenses are not correctly aligned with the center of the camera. Let (u, v) be true pixel image coordinates, that is, coordinates with ideal projection, and $(\hat{u}, \hat{v})$ be corresponding real observed (distorted) image coordinates. Ideal (distortion-free) and real (distorted) image physical coordinates are the same. Taking into account two expansion terms gives the following:

$$\begin{aligned} \hat{x} &= x + x[k_1 \cdot r^2 + k_2 \cdot r^4] + [2p_1xy + p_2(r^2 + 2x^2)] \\ \hat{y} &= y + y[k_1 \cdot r^2 + k_2 \cdot r^4] + [2p_1xy + p_2(r^2 + 2y^2)] \end{aligned} \qquad \text{eq. 5-53.}$$

where $r^2 = x^2 + y^2$. Second therms in the above relations describe radial distortion and the third ones - tangential. The center of the radial distortion is the same as the principal point. Because $\hat{u} = c_x + f_x u$ and $\hat{v} = c_y + f_y v$, where c_x, c_y, f_x, and f_y are components of the camera intrinsic matrix, the resultant system can be rewritten as follows:

$$\begin{aligned} \hat{u} &= u + (u - c_x)[k_1 \cdot r^2 + k_2 \cdot r^4 + 2p_1y + p_2(r^2/x + 2x)] \\ \hat{v} &= v + (v - c_y)[k_1 \cdot r^2 + k_2 \cdot r^4 + 2p_1x + p_2(r^2/y + 2y)] \end{aligned} \qquad \text{eq. 5-54.}$$

These two relations are used to restore distorted images from the camera ($(\hat{u}, \hat{v})$ are measured and (u, v) are computed image coordinates).

5.3.3.7 Experimental Results

The camera calibration algorithm has been tested on real data obtained from a USB webcamera, and also on a Firewire web camera. The closed-form solution involves finding a singular value decomposition of a small 2nx6 matrix, where n is the number of images. The algorithm continues the refining with the Levenberg-Marquardt non-linear algorithm.

Performance with respect to the number of planes

The orientation of the first three images is with a rotation $r_1 = [30^o, 0, 0]^T$, $r_2 = [0, 30^o, 0]^T$ and $r_3 = [-30^o, -30^o, 15^o]^T$, and a translation so that the fiducial is as big as possible in the camera image, but still entirely visible. From the fourth image, we randomly choose a rotation axis in a uniform sphere so that it is different from all the others (to avoid degenerate configurations). The results are presented in Figure 5-12. On the left we see the influence for α and β, and on the right for u_o and v_o. The error decreases when more images are used. From 2 to 3, the error decreases significantly and as can be seen, it is advisable to use at least 8 images.

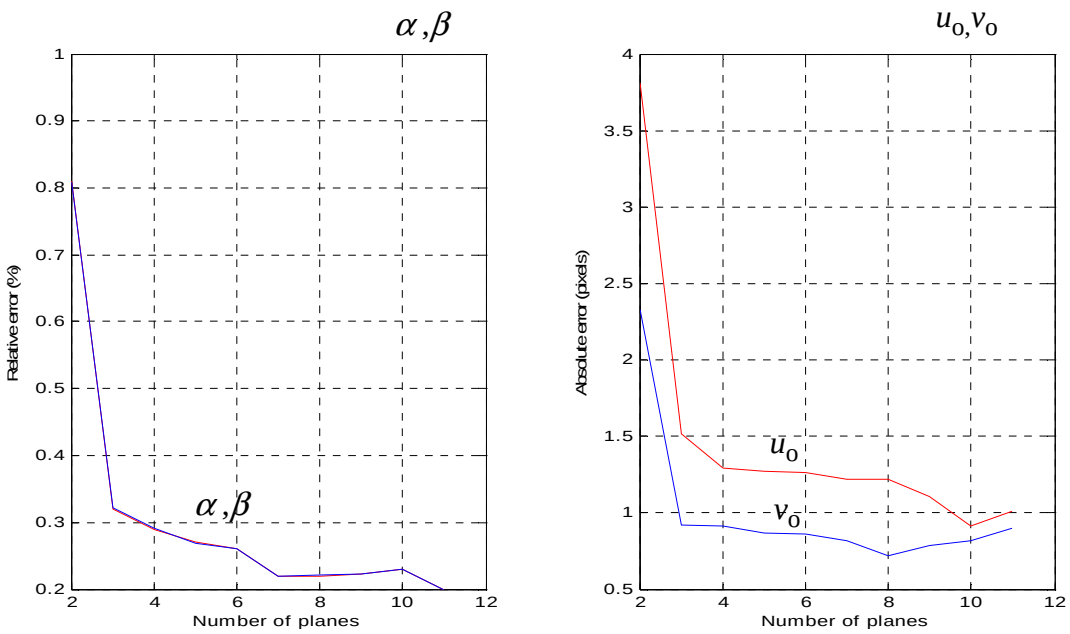

Figure 5-12. Error versus the number of images for Firewire webcamera (ADS Pyro).

Performance with respect to the orientation of the model plane

This experiment was done again to estimate the influence of the orientation of the model plane with respect to the image plane on the α, β, u_0 and v_0 parameters (see Figure 5-12). In this experiment only three images are used. The first one is parallel to the image plane and the other two are rotated with plus and minus the specified angle around a rotation axis. When the angle is smaller than 5^o, about 50% of the time the result is undefined because the planes are almost parallel to each other (degenerate configuration). The best performance seems to be achieved with an angle around 45^o, but in practice, when the angle increases, foreshortening makes corner detection less precise.

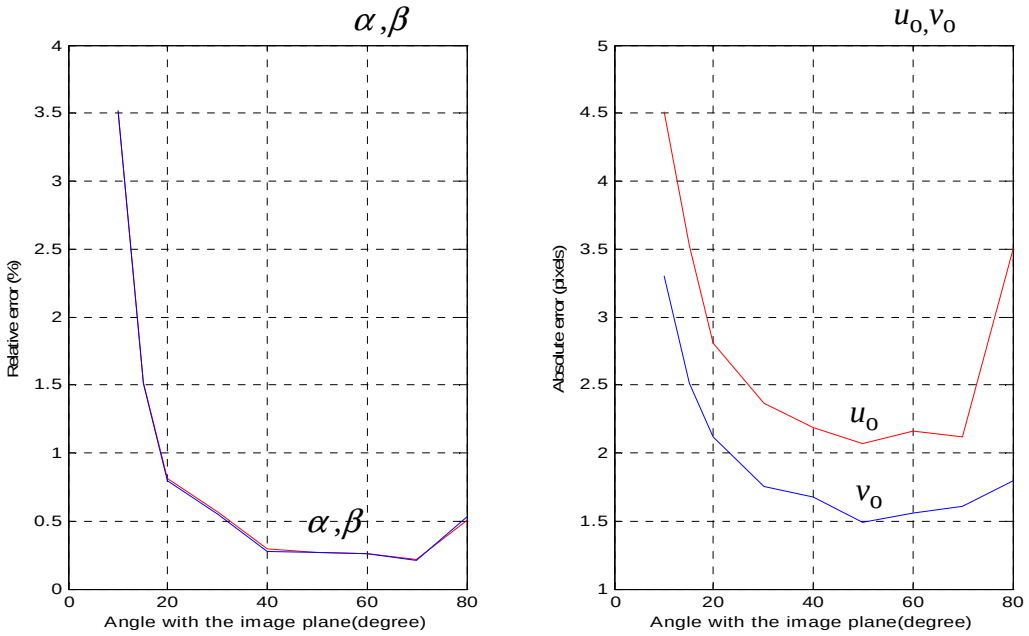

Figure 5-13. Error versus the angle of the model plane with respect to the image plane for Firewire webcamera (ADS Pyro).

Through experiments, we found that Tsai's method yielded the most accurate results when trained on data of low measurement error (Figure 5-14). This, however, is difficult to achieve in practice without an expensive and time-consuming setup. In contrast, our planar calibration method, although sensitive to noise in training data, takes advantage of the calibration pattern's planar constraints and requires only relative measurements between adjacent calibration points, which can be accomplished at very high accuracy with trivial effort. Thus, in the absence of sophisticated measurement apparatus, our planar calibration results easily outperform those of Tsai.

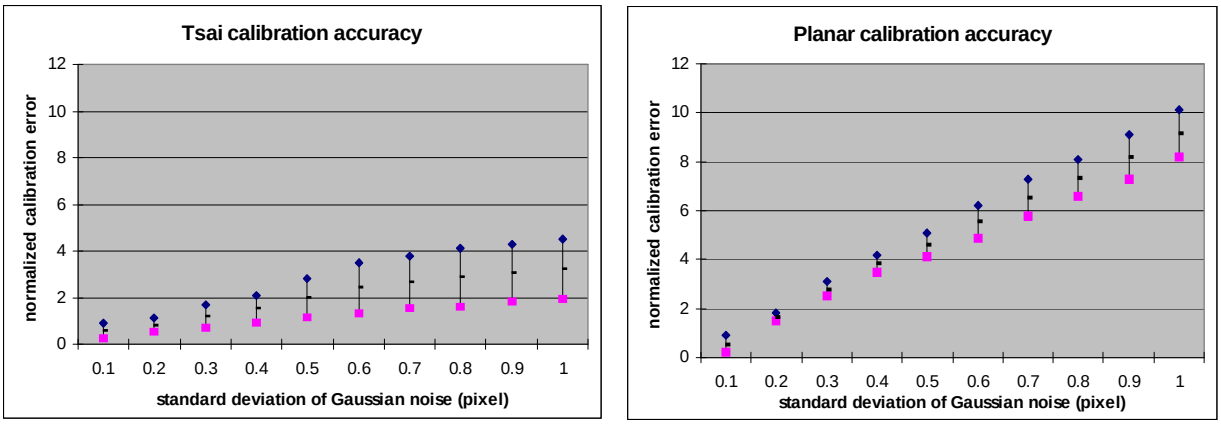

Figure 5-14. Effect of pixel coordinate noise on calibration accuracy

Note that the same camera calibration algorithm can also be used for pose computation. Once we have the calibrated camera we can run the whole algorithm with a new plane. The proposed algorithm will then compute only the extrinsic parameters, the intrinsic ones will remain the same.

5.4 Pose Computation Algorithms

5.4.1 Fiducial System

The fiducial system consists of calibrated landmarks, which are crucial for vision tracking. Different types of landmarks have been suggested in several systems, including corner features, square shape markers, circular markers, and multi-ring color markers [61][63][64]. In our work, we designed landmarks whose appearance simplifies robust detection and recognition. After evaluating many options, we selected the square shape marker similar to [63] with unique patterns inside. This design is easily printed on any BW printer (Figure 5-15). Its simple geometry lends itself to mathematical perspective projection compensation, leading to robust landmark detection and recognition.

To make thresholding operations robust in different lighting conditions, we apply a modified form of homomorphic filtering, which is designed to eliminate the effect of non-uniform lighting in images. In ordinary homomorphic processing, the logarithm of the image is taken in order to separate illumination and reflectance into additive terms. Then apply a high-pass filter designed to attenuate the slow-varying illumination term, and exponentiate to restore a good-looking image. Realizing that we were not concerned with restoring a good-looking image at the end, we used a 3x3 Sobel edge detector and we skippped the final exponentiation that is normally used to restore the image to human-viewable brightness levels. We found that now with one threshold value, at least 90% of the original fiducials are detected.

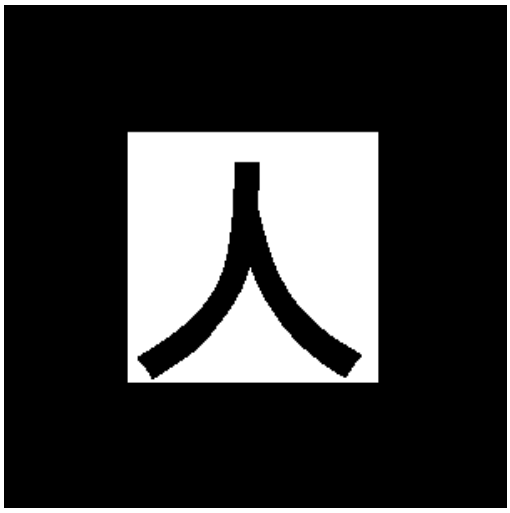

Figure 5-15. Example of symbol to be recognized

Landmark detection

The detection procedure is performed in three steps:

Coarse detection: using a predicted position (from the previous frame) we find the regions of the image where the gradients are high over an expected-size image area. In this way, we detect the four candidates corner points.

Projection compensation: using a perspective-imaging model we compensate for geometric deformations of the extracted regions. To compensate for lighting variations we use the gradient for intensity normalization.

Fine detection: fit the extracted candidate landmarks to the defined models. The best fit determines the detected landmark.

Landmark recognition

Since we want camera pose for each video frame, fast and efficient landmark detection and recognition are crucial. We have developed a contour moments method that robustly detects and recognizes the square landmarks in real-time. First, we detect corner positions of the square pattern. Based on this information we remove the effect of perspective projection and extract the contour of the symbol enclosed by the square pattern. From the contour we compute the 7 Hu moments[69] that will be used for recognition. The task of landmark designs is to choose ones that have as different Hu moments as possible.

Landmark positions

Landmarks printed on paper are easy to stick to walls, ceiling or floors in indoor environments. The best option indoors is to mount those markers on the ceiling because of less interference with other objects. There are few walls outdoors on which to mount fiducials, and they are far away. So the best option is to mount fiducials on the ground.

Careful quantitative experiments comparing this system with the AR Toolkit have not yet been completed, however a qualitative analysis of several sequences containing targets under a variety of scene conditions has been performed. Althought the AR Toolkit performs well in the office, it fails under motion blur and when camera lighting disrupts the binarization. The template matching to identify a specific target does not incorporate any intensity normalization and is therefore very sensitive to lighting changes. This detector demonstrated 95% overall performance through indoor and outdoor scenes.

This detection and recognition approach is robust and fast. It achieves 20 frames/sec on a 450 MHz PC. The system detects and discriminates between dozens of uniquely marked landmarks.

5.4.2 Pose Approximation Method

Using weak-perspective projection, a method for determining approximate pose, termed pose from orthography and scaling (POS) in [70], can be derived. First, a reference point P_0 in the world is chosen from which all other world points can be described as vectors: $\vec{P} = P_i - P_0$ (see Figure 5-16).

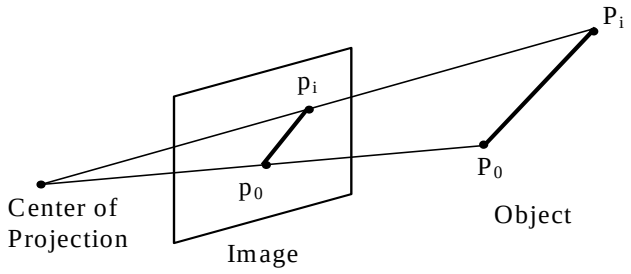

Figure 5-16. Scaling of Vectors in Weak-Perspective Projection

Similarly, the projection of this point, namely p_0, is a reference point for the image points: $\vec{p} = p_i - p_0$. As follows from the weak-perspective assumption, the *x* component of $\vec{p_i}$ is a scaled-down form of the *x* component of $\vec{P}_i$:

$$x_i - x_0 = s(X_i - X_0) = s(\vec{P} \cdot \hat{i}) \qquad \text{eq. 5-55.}$$

This is also true for the *y* component. If *I* and *J* are defined as scaled-up versions of the unit vectors $\hat{i}$ and $\hat{j}$ ($I = s\hat{i}$ and $J = s\hat{j}$), then

$$x_i - x_o = \vec{p}_i \cdot I \qquad y_i - y_o = \vec{p}_i \cdot J \qquad \text{eq. 5-56.}$$

as two equations for each point for which *I* and *J* are unknown. These equations, collected over all the points, can be put into matrix form as:

$$\underline{x} = MI \qquad \underline{y} = MJ \qquad \text{eq. 5-57.}$$

where $\underline{x}$ and $\underline{y}$ are vectors of *x* and *y* components of $\vec{p}_i$ respectively, and *M* is a matrix whose rows are the $\vec{p}_i$ vectors. These two sets of equations can be further joined to construct a single set of linear equations:

$$\left[\underline{x}\ \underline{y}\right] = M\left[I\ J\right] \Rightarrow \underline{\vec{p}_i} C = M\left[I\ J\right] \qquad \text{eq. 5-58.}$$

where $\underline{\vec{p}_i}$ is a matrix whose rows are $\vec{p}_i$. The latter equation is an overconstrained system of linear equations that can be solved for *I* and *J* in a least-squares sense as:

$$\left[I\ J\right] = M^{+}\underline{\vec{p}_i} \qquad \text{eq. 5-59.}$$

where M^{+} is the pseudo-inverse of *M*.

Now that we have *I* and *J*, we construct the pose estimate as follows. First, $\hat{i}$ and $\hat{j}$ are estimated as *I* and *J* normalized, that is, scaled to unit length. By construction, these are the first two rows of the rotation matrix, and their cross-product is the third row:

$$R = \begin{bmatrix} \hat{i}^T \\ \hat{j}^T \\ (\hat{i} \times \hat{j})^T \end{bmatrix} \qquad \text{eq. 5-60.}$$

The average of the magnitudes of I and J is an estimate of the weak-perspective scale[70]. From the weak-perspective equations, the world point in camera coordinates is the image point in camera coordinates scaled by s:

$$P_0 = p_0 / s = \begin{bmatrix} x_0 & y_0 & f \end{bmatrix} / s \qquad \text{eq. 5-61.}$$

which is precisely the translation vector being sought.

The Algorithm

The POSIT algorithm was first presented in the paper by DeMenthon and Davis [70]. In this paper, the authors first describe their POS (Pose from Orthography and Scaling) algorithm. By approximating perspective projection with weak-perspective projection, POS produces a pose estimate from a given image. POS can be repeatedly used by constructing a new weak perspective image from each pose estimate and feeding it into the next iteration. The calculated images are estimates of the initial perspective image with successively smaller amounts of “perspective distortion” so that the final image contains no such distortion. The authors term this iterative use of POS as POSIT (POS with ITerations).

POSIT requires three pieces of known information:

- The object model, consisting of N points, each with unique 3D coordinates. N must be greater than 3, and the points must be non-degenerate (non-coplanar) to avoid algorithmic difficulties. Better results are achieved by using more points and by choosing points as far from coplanarity as possible. The object model is an N x 3 matrix.
- The object image, which is the set of 2D points resulting from a camera projection of the model, points onto an image plane; it is a function of the object current pose. The object image is an N x 2 matrix.
- The camera intrinsic parameters, namely, the focal length of the camera.

Given the object model and the object image, the algorithm proceeds as follows:

1. The object image is assumed to be a weak perspective image of the object, from which a least-squares pose approximation is calculated via the object model pseudoinverse.
2. From this approximate pose the object model is projected onto the image plane to construct a new weak perspective image.
3. From this image a new approximate pose is found using least-squares, which in turn determines another weak perspective image, and so on.

For well-behaved inputs, this procedure converges to an unchanging weak perspective image, whose corresponding pose is the final calculated object pose.

```
POSIT (imagePoints, objectPoints, focalLength) {
        count = converged = 0;
        modelVectors = modelPoints – modelPoints(0);
        oldWeakImagePoints = imagePoints;
        while (!converged) {
                if (count == 0)
                        imageVectors = imagePoints – imagePoints(0);
                else {
                        weakImagePoints = imagePoints .*
                        ((1 + modelVectors*row3/translation(3));
                        imageDifference = sum(sum(abs( round(weakImagePoints) –
                        round(oldWeakImagePoints))));
                        oldWeakImagePoints = weakImagePoints;
                        imageVectors = weakImagePoints – weakImagePoints(0);
                }
                [I J] = pseudoinverse(modelVectors) * imageVectors;
                row1 = I / norm(I);
                row2 = J / norm(J);
                row3 = crossproduct(row1, row2);
                rotation = [row1; row2; row3];
                scale = (norm(I) + norm(J)) / 2;
                translation = [imagePoints(1,1); imagePoints(1,2); focalLength] /scale;
                converged = (count > 0) && (imageDifference < 1);
                count = count + 1;
        }
        return {rotation, translation};
}
```

Figure 5-17. POSIT Algorithm in Pseudo-Code

As the first step assumes, the object image is a weak perspective image of the object. It is a valid assumption only for an object that is far enough from the camera so that “perspective distortions” are insignificant. For such objects the correct pose is recovered immediately and convergence occurs at the second iteration. For less ideal situations, the pose is quickly recovered after several iterations. However, convergence is not guaranteed when perspective distortions are significant, for example, when an object is close to the camera with pronounced foreshortening. DeMenthon and Davis state that “convergence seems to be guaranteed if the image features are at a distance from the image center shorter than the focal length”[70]. Fortunately, this occurs for most realistic camera and object configurations.

Pose Accuracy Evaluation

For the validation of the results we have collected measures in different positions uniformly distributed on the workspace. The camera used in this experiment is a color CCD camera with a resolution of 640x480 (Firewire webcam - ADS Pyro). The camera orientation has been set in each case so that the whole target was in field of view of the camera. For every point we have carried out 50 measures. The target object (Figure 5-18) is composed of several circle markers, which provide at least six non-coplanar feature points.

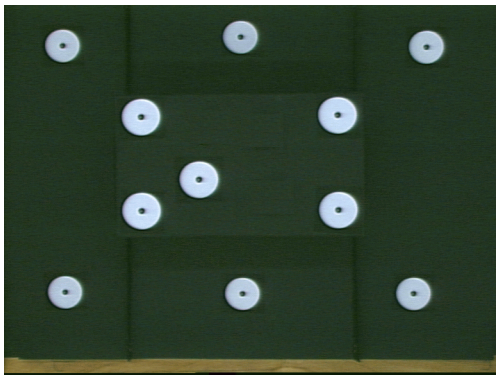

Figure 5-18. Non-coplanar target used in the experiment

In order to verify the pose estimation in regard to the scene range, known positions in the area of 50-400 cm in depth are measured in 20 cm steps (see Figure 5-19). With the vision-based positioning system presented in this chapter, we encountered a maximum error of 2.5 cm in the range of 4 m. The plots depicted in Figure 5-19 show that objects in the closer field of view can be located with a high precision rate. In order to estimate the position of the cameras (user), objects in the lower ranges have to be identified and should be tracked to get relative translation values.

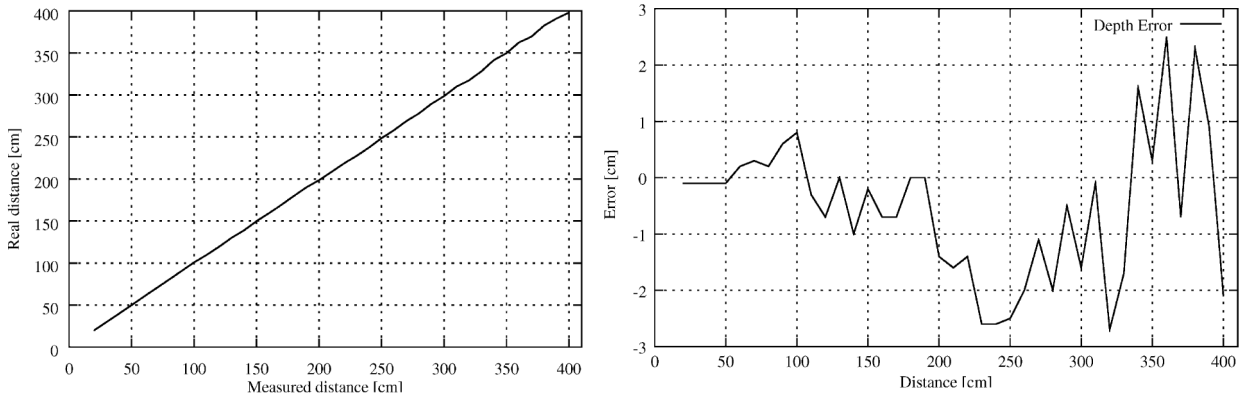

Figure 5-19. POSIT Algorithm results

Distance to tracked features

A crucial aspect is the distance to the objects being tracked. Here, we consider features within 5 meters to be 'nearby', and features farther away to be distant. Tracking more distant features is advantageous in various respects. First, a small error on the stored real-world position of that feature has fewer consequences on the calculated position, as in the projected image of the feature the error scales with the distance to the object. Second, the larger the distance, the larger the choice becomes of known features in the database, and so it will be more likely that a good feature is available. Third, distant features will change less as the observer moves as compared to nearby features. Fourth, nearby features are nearby only a short amount of time, as the user is moving around, and therefore use of nearby features requires more frequent refreshing of the features being used than distant features.

However, we doubt that distant features can be used for centimeter-accurate tracking of the user. Consider the following simple scenario, where a 1024x768 camera with a large-angle lens of 90^{o}

horizontal field of view is used. The average width of a pixel is now $90^o/1024 = 0.09^o$, so the feature direction will be estimated 0.045^o off on the average, assuming we carry out pixel-accurate tracking. We know the real-world location *x* and we want to know the distance to the object *d* (Figure 5-20).

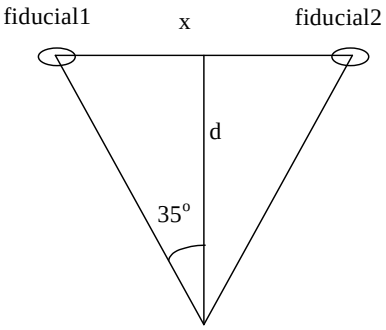

Figure 5-20. Two features within the camera's field of view.

If we optimistically assume that we have two fiducials in the corner of the camera's field of view, the angle between the two will be about 70^o, and so *d* follows from $\tan 35^o = x/d$. But we do not know if this is 35^o exactly - it probably is 0.045^o off - so we may be at another distance *d*' with $x/d' = \tan(35.045^o)$. Hence we have for our uncertainty on the distance $d'/d = \tan(35^o)/\tan(35.045^o) = 0.99833$. So, we have about 2 mm uncertainty per meter distance to the fiducial. If we want centimeter accuracy we have to have $d<5$ m. In less optimal cases, where the angle between the two is smaller, the maximum distance will be even less (Figure 5-21).

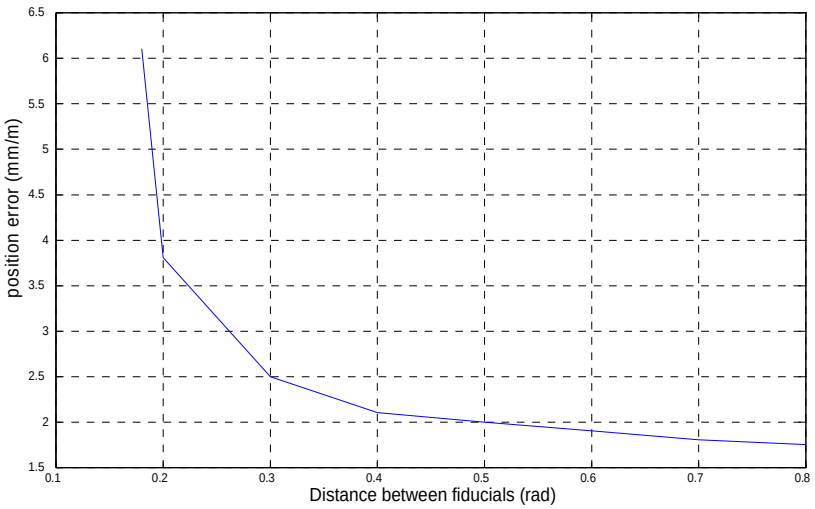

Figure 5-21. Uncertainty in the distance (mm. of error per meter distance to the fiducial) as a function of the angular distance between the fiducials.

With wide-angle lenses one can increase the working volume at the expense of resolution. For example, to cover a 5 X 5 X 2.4 m working volume inside a 6 X 6 X 3 m room using four cameras mounted in the corners of the ceiling would require cameras with 78° horizontal and 76° vertical FOV. Assuming 1000 X 1000 pixel cameras and 0.1-pixel resolution for locating beacons, this would yield a resolution of about 0.7 mm. This is quite an adequate positional resolution for many

applications, but the orientation must be computed from the positions of three beacons mounted on a rigid triangle. A triangle of 15-cm per side would provide orientation resolution of about 0.4° .

5.5 Real-Time Image Processing

In order to make fast object tracking systems with computer vision, the underlaying image processing algorithms must be fast enough to be valuable for a real-time tracker. Fast enough meaning that the entire image processing sequence must be performed within 1-2 video frame time, i.e. 40-80 msec. To realize this there are two possibilities: to use special hardware, such as the NEC Integrated Memory Array Processor (IMAP) system [34] or the Philips Xetal[1] chip [38], or to use the multimedia extensions on general purpose processors [32].

In this chapter we present a study on the impact of MMX technology and PIII Streaming SIMD (Single Instruction stream, Multiple Data stream) extensions in image processing and machine vision applications. A comparison with traditional scalar code and with a parallel SIMD architecture (IMAP-VISION) [75],[82] is discussed, while emphasizing the particular programming strategies for speed optimization. More precisely, we discuss low-level and intermediate-level image processing algorithms that can be used in a parallel SIMD implementation. High-level image processing algorithms are more suitable for implementation on MIMD architectures.

Current general-purpose processors have been enhanced with new features explicitly dedicated to the efficient handling of multimedia (audio and video) data; in order to exploit the high intrinsic spatial parallelism of low-level image processing, a SIMD solution has been adopted. Low-level image processing operators can be classified as point operators, neighborhood operators, global operators and special operators, with respect to the way the output pixels are determined from the input pixels. The simplest cases are the point operators where a pixel from the output image depends only on the value of the same pixel from the input image. More generally, neighborhood operators compute the value of a pixel from the output image as an operation on the pixels of a neighborhood around a corresponding pixel from the input image, possibly using a kernel mask. The values of the output pixels do not depend on each other. The most complex case consists of global operators where the value of a pixel from the output image depends on all pixels from the input image. It can be observed that most of the image processing operators exhibit natural parallelism in the sense that the input image data required to compute a given area of the output is spatially localized. This high degree of natural parallelism exhibited by most of the low-level image processing operators can be easily exploited using SIMD parallel architectures or techniques. SIMD architecture consists of a linear array of simple processors capable of applying in parallel the same instruction on different elements of the data. For MMX implementation we have also considered the exploitation of the instruction level parallelism of the code and data reduction in order to fit into the internal processor cache.

5.5.1 MMX Implementations

A set of MMX routines that allow us to compare the MMX SIMD and IMAP-VISION system has been developed; they include some low-level and some middle-level image processing routines. MMX technology for accelerating multimedia applications has become commonplace in recent years. Many of the core requirements of multimedia processing overlap with industrial machine vision requirements, and so it is natural that the vision community benefits from this computational

1. A paraphrase of the Dutch: “Ik zie het al”, more or less translatable in “Oh, I see!”

capacity. Intel's MMX Technology [31] adds several new data types and 57 new instructions specifically designed to manipulate and process video, audio and graphical data more efficiently. These types of applications often use repetitive loops that, while occupying 10 percent or less of the overall application code, can account for up to 90 percent of the execution time. A process called Single Instruction Multiple data (SIMD) enables one instruction to perform the same function on multiple pieces of data. The new data types allow handling of 64-bit data. This is accomplished by reassigning 64 bits of each of the eight 80-bit floating-point registers as an MMX register. The 64-bit registers may be thought of as eight 8-bit bytes, four 16-bit words, two 32-bit double words, or one 64-bit quadword.

MMX software writing [32] currently still implies the use of assembly language. We will focus in this discussion on a couple of strategies which are imposed by the architecture and that guarantee the highest performance.

5.5.1.1 Data Alignment

Data alignment is very important when writing MMX code as the execution speed can be boosted by more than 30%. When data access to a cachable address misses the data cache, the entire line is brought into the cache from external memory. This is called a line fill. On Pentium and dynamic execution (P6-family) processors, data arrives in a burst composed of four 8-byte sections to match the cache line size of 32 bytes. On the P6 when a write occurs and the write misses the cache, the entire 32-byte cache line is fetched. On the Pentium processor, when the same write miss occurs, the write is simply sent out to memory. Therefore, by aligning data on a 32-byte boundary in memory, one may take advantage of matching the cache line size and avoiding multiple memory-cache transfers. The delay for a cache miss on the Pentium with MMX is 8 internal clock cycles and on a P6 the minimum delay is 10 internal clock cycles. This is a serious performance penalty if we think that an aligned memory access might take only 1 cycle to execute. In order to align data on a 32-byte boundary, padding of data may be required. Many compilers allow the user to specify the alignment.

Moreover, if the code pushes MMX registers onto a stack, it is necessary to replace the entry and exit code of the procedure to ensure that the stack is aligned too. As a matter of convention, compilers allocate anything that is not static on the stack and it may be convenient to make use of 64-bit data quantities that are stored onto the stack. Another reason for aligning the stack may be the following. If inside the MMX routine an argument passed to a routine is accessed several times, it would be better on an aligned boundary to avoid cache misses.

5.5.1.2 Instruction Scheduling

To get the highest speed from MMX, one has to think in terms of instruction scheduling. The most critical scheduling involves output operand collision. The Pentium processor is an advanced superscalar processor. It is built around two general-purpose integer pipelines and a pipelined floating-point unit, allowing the processor to execute two integer instructions simultaneously. The first logical pipe is the U-pipe, and the second is the V-pipe. During decoding of an instruction, the next two instructions are checked, and, if is possible, they are issued such that the first one executes in the U-pipe and the second in the V-pipe. If this is not possible, only one instruction will be issued to the U-pipe and no instruction is issued to the V-pipe. This means that the execution of two instruction in one clock cycle might double the performance. For this reason, it is advised to keep both pipes busy. But at the same time we are limited by the hardware, as the Pentium has only one shift register, one multiplier, and only the U-pipe can execute instructions that access the memory or integer register and hence [32]:

- Two MMX instructions that both use the MMX multiplier unit (*pmull*, *pmulh*, *pmadd*) cannot pair. Multiply operations may be issued in either the U-pipe or the V-pipe but not in the same cycle.
- Two MMX instructions that both use the MMX shifter unit (*pack*, *unpack*, and *shift* instruction) cannot pair.
- MMX instructions that access either memory or integer register can be issued in the U-pipe only.

5.5.1.3 Tuning MMX Code

This section presents loop variables reduction, loop unrolling and loop interleaving techniques. One optimization technique is to consider the elimination of an inner loop. *For* loops are complex structures requiring a counter variable. Counter variables place an additional strain on the CPU register pool, which in the Intel architecture is uncomfortably small. Using a pointer increment with an end-of-line limit greatly increases speed since the incrementing pointer is also the counter loop. Usually the MMX code consists of short loops iterated many times over an array of data. Often there is no relationship between two iterations of the same loop, so the output of the second iteration does not depend on the output of the first. Having calculated the highest number of iterations that can be executed in parallel (*N*), the loop can be unrolled by *N* and then instructions can be moved around to maximize the use of both pipelines:

- MMX instructions that access memory are always executed in the U-pipe, so it is better to spread them around and try to exploit the V-pipe with ALU or shift instructions.
- Two shift or multiply instructions cannot be issued in the same cycle. We can introduce a small delay between the iterations, so that the group of MMX instructions that are physically near belong to different stages of their respective iterations.

For operations on binary images like noise filters, morphological operations or measurements can be performed on a packed image. In this way 64 pixels will fit in a single MMX register and if we can gain advantage from loop interleaving optimization and when issuing 2 operations in one cycle, we can process 128 pixels at once. Since a typical image is 256 x 256, the use of one bit per pixel shrinks the output data down to 8k bytes, allowing a more efficient use of the Pentium's L1 cache memory. Unfortunately this packing requires a lot of instructions and an additional phase of bit swapping. Also careful instruction scheduling is required due to the limitations on the use of shift instructions: only one shift operation can be issued in each clock cycle.

5.5.1.4 The Intel VTune Performance Analyzer

Intel’s VTune is one of the standard performance analyzers for the x86 architectures. It uses Pentium on-chip performance-monitoring hardware counters that keep track of many processor-related events. For each event type, VTune can count the total number of events during an execution of a program and locate the spots in the program’s source code where these events took place (with corresponding frequencies). We used VTune to gather statistics on the following event types: clockticks, total instructions retired, L1 cache line allocated, L1 misses outstanding (roughly this equals the number of L1 cache misses times the average number of cycles to service the miss), L2 Cache Read Misses, L2 Cache Write Misses, L2 Cache Instruction Fetch Misses, L2 Cache Request Misses (equals the sum of the previous three categories), L2 Cache Reads, L2 Cache Writes, L2 Cache Instruction Fetches, L2 Cache Requests (equals the sum of the previous three categories).

The VTune Performance Analyzer runs programs and takes rapid snapshots of the execution path based on time slices (the users can determine the delay) or user events. It does so without requiring

the program to be recompiled or instrumented. Using this run, the VTune Performance Analyzer creates a database-like profile of the executable from the snapshots and presents the information in a variety of ways. These include the views commonly found in execution profilers:

- *Code level:* This view presents source code with hit counts and duration beside each line of code. Optionally, users can request that assembly language is interspersed between lines of high-level code.
- *Call graph:* This is a diagrammatic representation of which functions called which other functions. It is presented as a calling tree with its root on the left side of the screen with calls flowing to the right.
- A *bar graph* showing which program and operating-system modules consume the most execution time.

Figure 5-22 presents on the left a part of assembly code with its execution pipe, pairing and cache misses, and on the right the bar graph showing which routine consumes the most execution time (for pattern matching algorithm).

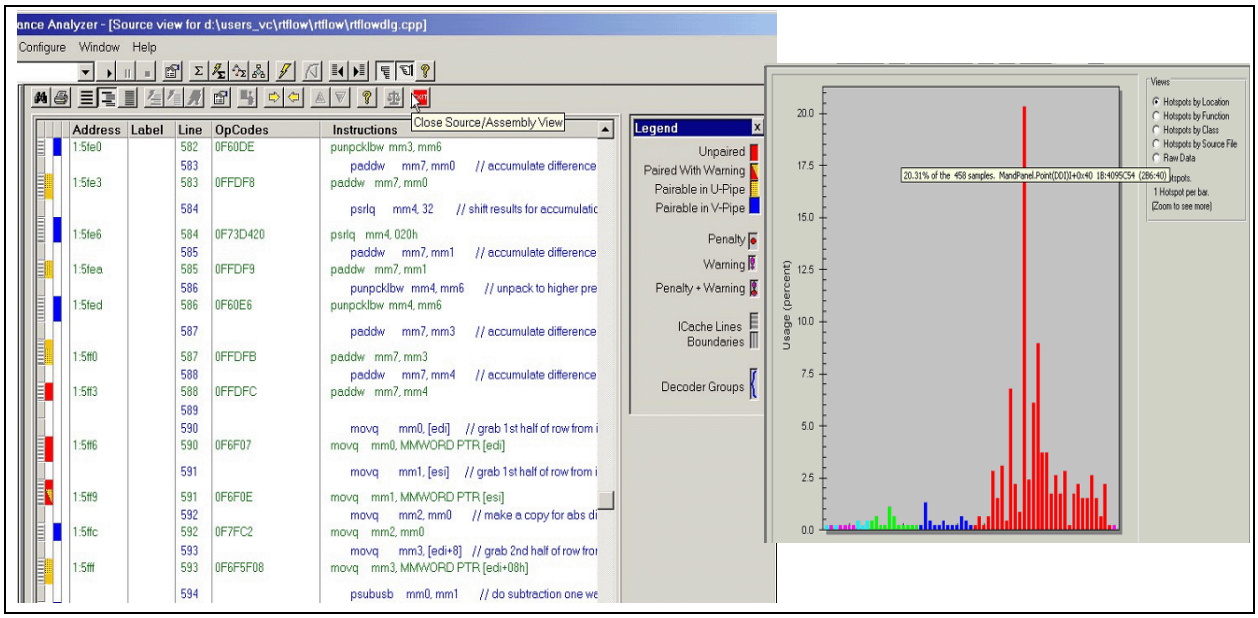

Figure 5-22. Intel VTune Performance Analyzer

VTune performs these measurements by sampling, which is inherently inaccurate. VTune implements a self-calibration mechanism which allows us to set the desired accuracy of the results. Additionally, VTune also can perform a dynamic analysis (simulation) of a portion of the code. The simulation takes a lot of time and is therefore mainly useful for short segments of code. We used dynamic analysis to understand better our program's behavior at hot-spots.

5.5.1.5 The Intel Pentium III Processor

The Intel Pentium III processor provides sufficient processing power for many real-time computer vision applications. In particular, the Pentium III includes integer and floating point Single Instruction Multiple Data (SIMD) instructions, which can greatly increase the speed of computer vision algorithms. The Streaming SIMD Extensions expand the SIMD capabilities of the Intel® Architecture. Previously, Intel MMX instructions allow SIMD integer operations. These new instructions implement floating-point operations on a set of eight new SIMD registers. Additionally, the

Streaming SIMD Extensions provide new integer instructions operating on the MMX registers as well as cache control instructions to optimize memory access. Applications using 3D graphics, digital imaging, and motion video are generally well suited for optimization with these new instructions. Multiprocessor Pentium III systems are relatively inexpensive, and provide similar memory bandwidths and computational power as non-Intel workstations costing significantly more money.

Block matching is essential in motion estimation. Equation 5-62 is used to calculate the Sum of Absolute Differences (also referred to as the Sum of Absolute Distortions), which is the output of the block-matching function (dx and dy represent the displacement in in x and y direction respectively).

$$SAD = \sum_{i=0}^{15} \sum_{j=0}^{15} \left| V_n(x+i, y+j) - V_m(x+dx+i, y+dy+j) \right| \qquad \text{eq. 5-62.}$$

Figure 5-23 illustrates how motion estimation is accomplished, *dx* and *dy* are candidate motion vectors. Motion estimation is accomplished by performing multiple block matching operations, each with a different *dx*, *dy*. The motion vector with the minimum SAD value is the best motion vector

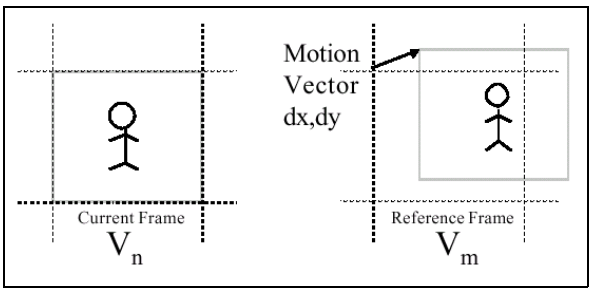

Figure 5-23. Block matching

Streaming SIMD Extensions provide a new instruction, *psadbw*, that speeds up block matching. The code to perform this operation is given below. This code has been observed [80] to provide a performance increase of up to twice that obtained when using MMX technology.

The operation of this new instruction, psadbw is given in Figure 5-24.

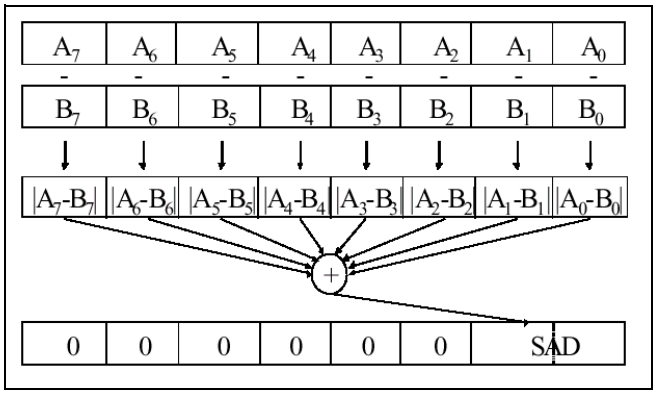

Figure 5-24. PSADBW instruction

5.5.1.6 MMX Image Processing implementation and benchmarks

An interesting problem is how to measure the execution time of an MMX routine. This could be done using the information from the real-time stamp counter (RDTSC), which contains the cycle counter. The detailed description of the RDTSC counter may be found in [33].

The measured timing is approximate and depends on many factors such as OS overheads, number of processes running, cash situation if MMX code contains read/write instructions, etc. Various conditions, such as cache warming prior to reading or writing from/to the same memory blocks, a particular write strategy implemented in the processor and L2 cache, most significantly affect the performance. For these reasons we need to carefully consider the results and run multiple test and average out the results excluding the values far from the mean, which may occur due to a particular running condition.

Figure 5-25. Optical Flow computation using Streaming SIMD instruction.

5.5.2 The NEC IMAP-VISION System

The IMAP-Vision [34] is a SIMD Linear Processor Array (LPA) on a PCI board. It is a parallel architecture for real-time image processing that includes a high-level language for data parallel programming labeled “one dimensional C” (1DC). The IMAP-Vision *card* contains 256 8-bit processing elements (PEs), controlled by a 16-bit control processor (CP) and it has a 10 GIPS peak performance. The IMAP-Vision *chip* integrates 32 8-bit processors and 32 8-Kbit SRAM. Processors are connected in series to create a one-dimensional SIMD processor array. Instructions for the processors are given from an off-chip, on-board dedicated control processor, which is an adapted NEC microprocessor. This control processor (CP) contains a 16-bit Arithmetic and Logic Unit (ALU), multiplier and barrel-shifter, 128 kB local memory and 32 16-bit registers. The PEs contain an 8-bit ALU/shifter. It can work on 1 kB of local, on-chip memory, using sixteen 8-bit registers plus three 1-bit registers. The on-chip memory is accessible in 1 clock cycle. Although the IMAP chip has high levels of computational ability by integrating one-dimensional SIMD processors on a single chip, its on-chip memory is sometime insufficient for flexible execution of complex algorithms [79]. For that reason, the IMAP-Vision board has a high bandwidth external memory with an efficient data transfer mechanism. The bus controller allows the host PC to access data not only in data memory, but also in on-chip memory and the external memory without collisions, even when a real-time vision computation is running. The PEs can load data from an external memory of 64 kB per PE which is accessible in eight clock cycles, plus five cycles overhead for the first line. Data can be copied between in- and external memory in parallel with computations. This mechanism not only allows the host PC to collect the results created by the processor array, but also

allows the host PC to change the parameters in run-time easily. Both internal and external memory are seen as one 256-byte wide, 8-bit deep, two-dimensional memory where each column is being treated by a PE. In 1DC, memory access is always done by means of (*sep*) variables and/or pointers to them, the placement of which is handled by the compiler, transparently for the user. An alignment at an address which is a multiple of 256 is possible for the use of lookup table multiplication. In IMAP-assembly, variables can be used or the memory can be addressed directly. While PEs can only access values in “their own” column of internal and external memory, the CP can access any part of any memory, in- and external PE memory, program memory and its own data memory.

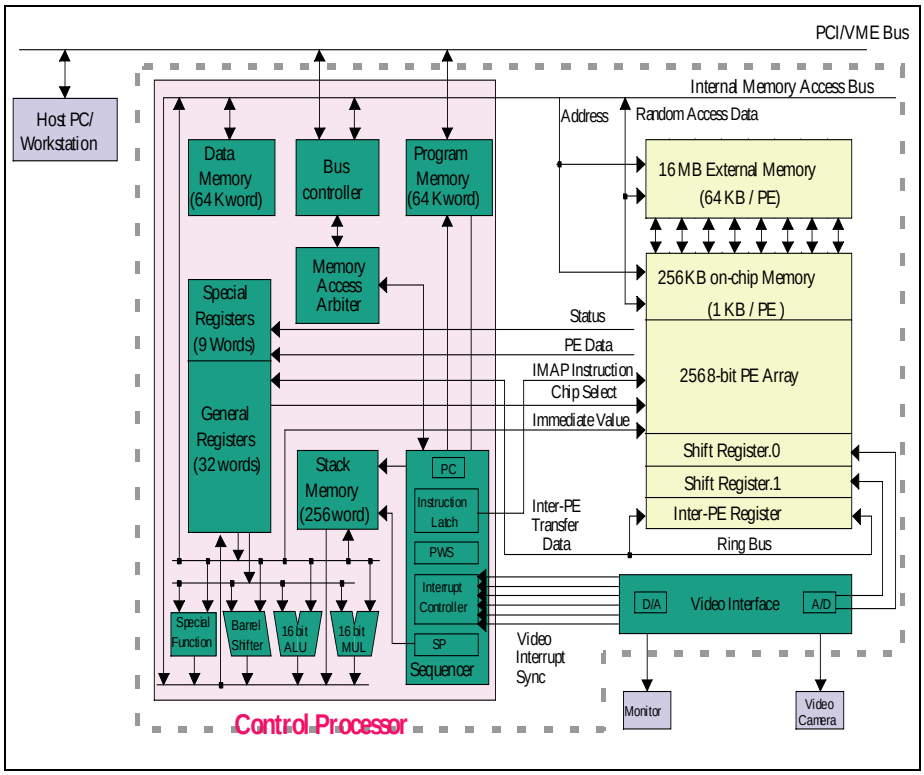

Figure 5-26. The IMAP-Vision board

Figure 5-26 shows the IMAP-Vision board. The IMAP-Vision board integrates eight IMAP Vision chips, eight 16 Mbit synchronous DRAMs (one for each IMAP-Vision chip), a control processor, video interface and a PCI interface in a single slot PCI bus board. The board can be used with a PC or a workstation that has a PCI slot, thus the IMAP Vision board constitutes a compact cost-effective system. Communication on the IMAP-VISION can be done in two ways: via the CP, and over the inter-PE bus. A single PE can be selected to write a value to the CP using a masking mechanism. By masking out all PEs except for one, this PE can write a value to the data bus where it can be read by the CP. When multiple PEs try to write data, the result is unpredictable. Data in a CP register or CP memory can be made globally available to all PEs (broadcast) by a special CP instruction. Each PE has access to its neighbor PEs' registers over the inter-PE bus in one clock-cycle. PEs further away can be reached through repetitive shifting. Because this does not work if one of the PEs between the source and the destination PE is masked out, a special shift register exists which is not maskable. This means data is always written to it, even by PEs which are not active. When, after shifting the value is copied to a work register or to memory, this is only done by active PEs.

5.5.2.1 The IMAP assembly and 1DC language

The IMAP-assembly language

The IMAP-assembly language comprises the RISC instruction sets of the IMAP-VISION chips and the control processor. It supports variables and macros. Every line of code can contain both a CP and a PE instruction, which will be executed simultaneously. Also two flags can be specified indicating if interrupts are allowed and if and how the PE mask register should be taken into account for the execution of the PE instruction. The CP instructions are distinguished from the PE instructions by a leading “c”. Also CP registers are called c0-c31 and PE registers r0-r18.

Besides the normal ALU instructions, the hardware and the corresponding instructions for the PEs have support for software multiplication using a lookup table (LUT), later versions of the chip have a multiplier. Normally it takes several ANDs and ORs to find the table indices given by 4 bits of one byte and 4 bits of the other, but the IMAP-VISION PEs can calculate the index and load the indexed value in one clock cycle. Also there are instructions for shifting a byte over 4-bits before adding it to another byte (with or without carry) in one cycle.

As always, it is clear that assembly language is not easy to use for everyday programming. It should be used only to optimize the computationally expensive part of (library) code.

The 1DC language

1DC is, as the name indicates, a C-like language [37]. It follows the ANSI-C syntax, and is designed as an enhanced C language to support virtual LPAs (Linear Processor Array). The enhancement of 1DC from C is straightforward:

- extended declaration of entities which are associated with the PE array (*sep* or distributed variables),
- extended constructors for selecting active processor groups,
- and extended operators (like *mif*) for manipulating data on the PE array.

Entities are declared as either *sep* (or *separate*) or *scalar* (*scalar* is the same as declarations in C language) as shown in Figure 5-27. A *sep* variable is associated with the processor array and stored on the on-chip memory or on external memory, while the *scalar* variables are associated with the control processor and stored either in the CP, the on-board memory, or on the external memory. For example, a 256x240 image, distributed column-wise over the PEs is defined as: separate unsigned char img[240].

The PEs have a mask register which, when set to 0, deactivates the processor. Extended instructions exist for selecting active processors:

- *mif* (condition) ... *melse* construction
- *mfor* (init; stop_cond; instruction) construction
- *mwhile* (condition) construction
- *mdo* ... *mwhile* (condition) construction

These instructions act just as their common counterparts, except for the fact that in the condition a separate expression has to occur, so that the condition may be true on some processors and false on others. The instructions will save the current mask on the processors, and in the case of the active processors for which the condition is false, zero the mask (make the processor inactive). Thus the instructions which are controlled by the condition are only executed by the processors for which the condition is true. After that the previous mask is restored. Saving and restoring the mask makes it possible to nest the masking instructions.

To make the value of a *sep* variable on a specific processor globally available it can be addressed using the :[*pe_num*:] operator. The operators for accessing variables on other processors, :< and :>,

can be used in two ways. As a unary operator, *sep_var* = :< *sep_var*; will rotate the values of *sep_var* one processor to the left. As a binary operator *sep_var* = *sep_var* :< *n*; will rotate the values of *sep_var n* processors to the left.

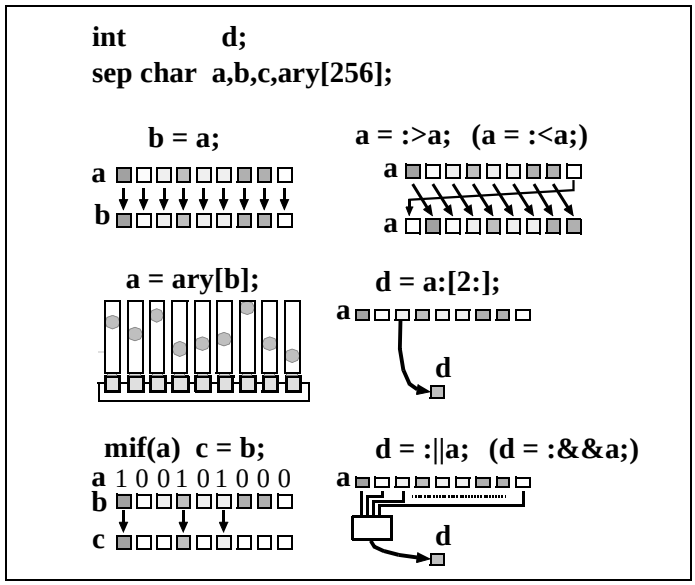

Figure 5-27. 1DC language extension

The IMAP-VISION board comes with an MS-Windows or X-Window (e.g. Linux) programming environment based on 1DC, which makes this parallel board a powerful tool. The X-Window based debugger provides not only assembly and source level debugging facilities, but also functionalities such as interactive adjustment of variables and constants, which is useful for parameter tuning in real-time imaging applications. The real-time adjustment of variables is possible due to the use of a bus controller that regulates memory accesses from the host PC to all memories. This controller allows the host PC to access data not only in data memory but in the on-chip memory and the external memory without collision. This mechanism not only allows the host PC to collect computational results created by the IMAP Vision, but also allows the host PC to change the parameter in run-time easily. When developing an application program which handles real-world images, trial and error parameters tuning and algorithm testing for a variety of data are essential. The integrated debugger provides GUI tools such as select buttons, 1D sliders and 2D scroll maps. These tools are associated with variables by clicking variable names in the source window and buttons associated with these tools. After that, in real-time processing, every time the programmer touches or drags these tools, corresponding values are written onto the associated variables.

The integrated debugger features are designed to:

- set up breakpoints by clicking lines in the source program
- display values by clicking variables in the source program (updated dynamically)
- display disassembled code of any function
- display image in any scale, also in character format (updated dynamically)
- change a value at run-time through variable setting windows
- display execution time profile in each line or function

Implementations and optimizations in 1DC

Writing a program in 1DC is not very difficult, but writing a good program is. A program that works is not necessarily efficient. In order to write efficient, i.e. fast programs, certain guidelines should be followed. The first is to always try to exploit parallel processing to the maximum. If one has 256 processors, one should use all of them. The second is to avoid using nested loops.

This sometimes necessitates a work around, which is not trivial. A simple but very expressive example is the histogram computation, which is a global image processing operation. First, every processor makes a local histogram of the image column it has in its memory. Probably most people would write something like (PENO in the total number of processors):

```
for(i=0;i<256;i++)
   for(j=0;j<PENO;j++)
      histo:[i:] += tmp[i]:[j:];
```

But in that case, we have a nested loop in which only 1 processor is working at a time. A more efficient use of processors would be to do a prefix addition, one of the most basic forms of parallel programming. It uses only log(256) = 8 steps:

```
for(i=0;i<256;i++){
        for(j=0;j<8;j++)
                tmp[i] += tmp[i] :< (1<<j);
                histo:[i:] = tmp[i]:[i:];}
```

But we can do better. Every processor could add its local value and then pass it on to its neighbor, which will add its local value while the first one adds the next value. Since the :< operator takes only one clock cycle we do not lose any time at all. Then we find the fastest code possible (PENUM is the processor number/location in the LPA):

```
for(i=0;i<256;i++)
        histo=:<(histo + tmp[(PENUM+i)& 255] );
```

Performing execution timing in the debugger gives results that speak for themselves:

- classical histogram: 60.10 ms
- using prefix addition: 2.91 ms
- optimal program: 0.29 ms

SIMD systems are made for local neighborhood operation (LNOs). North and south bound neighbors are retrieved from the PE's own memory, East and West bound neighbors are reached by shifting left and right.

Recursive neighborhood operations (RNOs) are also quite doable on the IMAP-VISION system, as it is equipped with indirect addressing. For updating each pixel, RNOs refer to the pixel value of a neighboring pixel, which was already updated. By delaying each PE, one iteration, each column is shifted and hence the already processed pixels come into reach without abandoning the SIMD principle. Examples of RNOs are morphological operations like thinning or the distance transform. A parallel method for efficient implementation of RNO can be used [35]. A distance transform implementation in 1DC will take 8ms on an image of dimension 256 x 240 with a 40 MHz system. Also the distributed bucket processing [40] (stack-based) method is extensively used. It consists of two processing phases during which every PE simulates a software stack in its local memory. In the first phase all pixels are visited once in order to find points with specific features such as contour pixels or peak pixels and push them onto the stack. In the second phase they are popped and processed. In the processing phase output pixels can be pushed back onto stacks, e.g., corresponding to an object contour where they belong for contour tracking or to accumulate votes

as in Hough transforms. All the pixels from the stacks are processed until all stacks on all PEs involved in the operation are empty. A circular Hough transform implemented using this method performs in 6 to 11ms. This image transform was used to detect, in real time, the ball in a robot soccer game [72] and a finger detection system for gesture recognition in human-computer interaction [76], Figure 5-28.

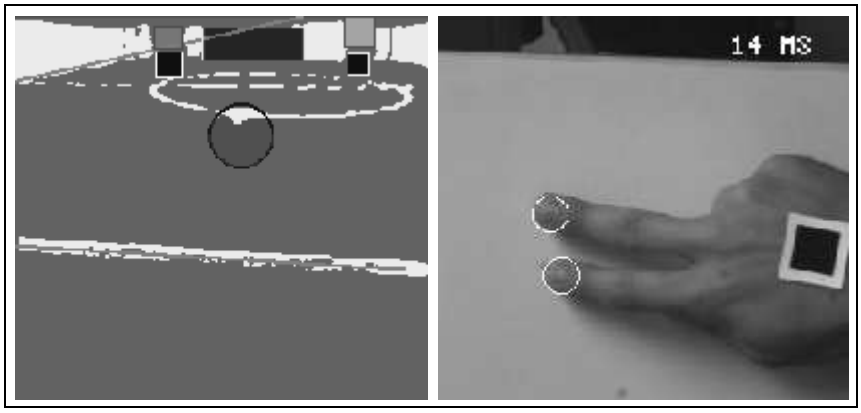

Figure 5-28. Examples of the Circular Hough Transform

The IMAP-CE chip

The IMAP-CE [36] is the fourth generation of a series of SIMD linear processor arrays based on the IMAP architecture. The aim of IMAP-CE is to provide a compact, cost-effective and yet high-performance solution for various embedded real-time vision applications, especially for vision-based driving assistance applications in the intelligent transportation system (ITS) fields. The IMAP-CE integrates 128 VLIW processing elements with one RISC control processor, which provides the single instruction stream for the processor array, on a single chip. The peak performance of IMAP-CE is up to 51.2 GOPS operating at 100 MHz.

Each PE of the processor array consists of a 2K byte local memory, 24 8-bit registers, two 1-bit mask registers, an 8-bit ALU and multiplier. PEs are connected in series to configure a one-dimensional SIMD processor array, to which instructions are provided by the internal Control Processor (CP), which is itself a 16-bit RISC processing unit. IMAP-CE supports the connection of a maximum of sixteen chips, which forms a 2048 PE system with a peak performance of 819.2 GOPS. The control processors operate either in lockstep or independently, making all kind of other configurations possible.

Refinement of the instruction set of IMAP-CE for accelerating SIMD operations has been done based on the analysis result of compiler-generated codes of 1DC for some intelligent transportation system applications. Code patterns which frequently appear are identified, and are checked for the possibility of replacement by instructions. As a result, special instructions for PE indexed address generation, PE conditional branch, and label propagation, are inserted on top of the existing RISC instructions. Wired-logic instructions are also introduced for accelerating label propagation operations, which is a major sub-operation for connected component labeling.

The IMAP-CE (128 PEs, 100MHz) is about 5 times faster than the P3 (Pentium III 1GHz), and 3 times faster than its predecessor, IMAP-Vision (256 PEs, 40MHz). Due to the enhanced instructions for label propagation, the corresponding performance of IMAP-CE is about 7 times faster than that of IMAP-Vision.

5.5.3 Experimental Results and Conclusions

Due to the language design and the RISC-like instruction-set of IMAP-VISION, the 1DC compiler has achieved codes competitive with hand-written assembly code as can be seen in Table 5-1. Because of this we will use code written in 1DC. The assembly should be used only to optimize the most time-consuming parts of the code. Also 1DC hides almost all the limitations of the hardware. Unsupported operations are replaced by corresponding macros by the compiler.

Table 5-1. 1DC compiler performance

Algorithm	**Assembly code steps**	**Compiler code steps**	**Ratio**
Average filter	5600	6430	1.15
Histogram	4039	4872	1.21
Rotation 90 degree	20696	23326	1.13

Also for complicated algorithms we tried out MMX technology intrinsics, developed by Intel. Intrinsics are highly optimized routines written in assembly, for which the C compiler generates inline assembler code. The intrinsics allow for relatively quick prototyping of code and is much easier to maintain than assembly code. On the other hand, only the Intel compiler supports this technique and the performance of this compiler code is about 15-25% slower than equivalent assembly code.

In Table 5-2 we make a comparison of the execution times between MMX code on a single Pentium II 300MHz processor and 1DC on the IMAP-VISION system. We used in our measurements a 256 x 256, 8 bits per pixel image.

Table 5-2. MMX versus IMAP-VISION timings

Operation Type	**MMX PII300MHz**	**IMAP-VISION**
Image Binarization	0.3ms	0.036ms
Images add	0.5ms	0.1ms
Image mean	0.9 ms	0.04ms
Image Multiplication	1.1ms	0.09ms
Images Bit And	0.8ms	0.07ms
Convolution 3x3 kernel	5.5ms	0.42ms
Sobel Edge Detection	2.4ms	0.4ms
Image Variance	14.8ms	0.65 ms
Histogram	10,6ms	0.29ms
Dilation (binary)	12,4ms	0.31 ms

Low-level image processing performs very well on a single MMX processor architecture and on PIII (with Streaming SIMD Extensions). The IMAP-VISION still performs on average 10 times faster, mostly because it has 256 PEs. The drawback of the IMAP-VISION system is the 40MHz operating frequency. Also, both systems have a major drawback. The IMAP-VISION system has no floating point (FP) operations, and within MMX code we cannot use floating point operations either (in fact we can, but it costs 50 processor cycles to switch between FP and MMX mode). With the new PIII Streaming SIMD Extensions this is not a problem any more.

In this study we did not take into account the acquisition and the transfer time. We can expect that the IMAP will perform far better because the bandwidth between on-chip and external memory or frame grabber is 1.28 GByte/s, which enables the transfer of 644 frames in 33.3ms (the NTSC frame rate).

While the IMAP-Vision system performs better because of the large number of processing elements, the MMX processor and PIII (with Streaming SIMD Extensions) remain a good candidate for low-level image processing, if we also take the price into account. However with IMAP-CE chips on the market, targeted at 10 dollars/unit, this should not be a problem any more.

A clear negative point is that programming MMX code is cumbersome, even using standard libraries, in contrast with the 1DC programming of the IMAP vision system.

5.5.4 Conclusions on Real-Time Image Processing

The pose tracking system for outdoor augmented reality, is partly based on a vision system that tracks the head position within centimeters, the head orientation within degrees, and has an update rate of within a second. The algorithms that are necessary to obtain a robust vision system for tracking the automotion of a camera based on its observations of the physical world contains feature detection algorithms, camera calibration routines and pose determination algorithms. To achieve our goal we need a system that is based on the real-time (video-rate) tracking of features, such as corners, line segments and circles / ellipses in the image. In order to achieve this we have investigated how these real-time constraints can be resolved. We have investigated two alternatives, both based on special hardware. One is based on special multimedia instructions in general purpose processors, and the other on special purpose image processing hardware, based on a solid paradigm, SIMD. We conclude that from these features, pose determination algorithms can calculate the automotion of the camera on a general purpose wearable computing platform, such as the LART system. To test our pose tracking system we have used throughout the project MMX-like multimedia extensions as well as the IMAP-Vision system.

Our conclusion is that the IMAP-Vision system performs better than the multimedia extensions, because of its large number of processing elements, its programmability, its memory interface, its direct interface to the camera, and finally, it leaves the general purpose processor (LART/Pentium) it is attached to free for other jobs, such as pose estimation or inertia tracking.

A problem however is the availability of the IMAP chips and boards, a problem that should be solved when the IMAP-CE chips come onto the market (2005?). What is left then is the power consumption of the chip. It is already far less than the power consumption of the old IMAP chips, and considerably better than the Pentium's, but the IMAP-CE's prime target is to be an embedded vision processor for industrial inspection, robotics and the automotive industry, and not for wearable computing. As such, the Philips XETAL chip becomes a good candidate. Although it has a less powerful architecture, it was targeted from the beginning on the low-power market and is thus suitable for wearable computing. As with the IMAP-CE, the design of architectures is alive and ongoing.

The MMX-type Streaming SIMD multimedia extensions to general purpose processors remain good candidates for low-level image processing, however provisionally not in low-power embedded systems for wearable computing.

5.6 Conclusions

Camera Calibration, Data Normalization

In this chapter, we have presented a flexible technique to easily calibrate a camera. The technique only requires the camera to observe a planar pattern from a few (at least two but in practice one should use about 10) different orientations. We can move either the camera or the planar pattern. The motion does not need to be known. Radial lens distortion is modeled. The proposed procedure consists of a closed-form solution, followed by a nonlinear refining based on a maximum likelihood criterion. Both computer simulation and real data have been used to test the proposed technique, and very good results have been obtained. Compared with conventional techniques which use expensive equipment such as two or three orthogonal planes, the proposed technique offers considerable flexibility. The proposed preprocessing (data normalization) of data points before calibration makes the algorithm stable to pixel noise.

Through experiments, we found that Tsai's method yielded the most accurate results when trained on data of low measurement error. This, however, is difficult to achieve in practice without an expensive and time-consuming setup. In contrast, our planar calibration method, although sensitive to noise in training data, takes advantage of the calibration pattern’s planar constraints and requires only relative measurements between adjacent calibration points, which can be accomplished at very high accuracy with trivial effort. Thus, in the absence of sophisticated measurement apparatus, our planar calibration results may easily outperform those of Tsai.

Vision-based trackers require a wide field-of-view (FOV) in order to overcome the geometric dilution of precision and to achieve sufficient sensitivity to distinguish small rotations from small translations. The use of either multiple narrow-FOV cameras or a single wide-angle camera allows the desirable precision to be achieved. One lightweight low-power camera is a cost effective solution that offers the additional benefit of leaving out the difficult mutual alignment required by multiple cameras. The focal length of the lenses used in our experiments is in the range of 2.2mm-3.6mm, yielding a FOV in the range of 50-80 degrees for a ¼ inch CCD (ADS Pyro webcam).

Calibration is essential when a combination of wide-angle lenses and a small CCD are used in a camera. The wide-angle lenses require accurate calibration to compensate for a considerable amount of radial distortion. Additionally, the miniature chip with a pixel size of about one micron is impossible to mount exactly lined up with the optical axis of the lens. A shift of 10-30 pixels in the image plane is typically observed between the optical axis of the lens and the center of the CCD chip. To compensate both the shift of CCD center and the lens radial distortion, a high quality calibration procedure for intrinsic parameters is required. With the calibration procedure described in this chapter we were able to obtain good calibration results.

Landmark recognition

For each detected landmark, we first extract the contour and compute Hu moments. M. Hu[69] has shown that a set of seven features derived from the second and third moments of contours is an invariant to translation, rotation, and scale change. Unfortunately Hu moments are not invariant to affine transformation (perspective transformation). So, before contour extraction we perform an inverse perspective projection. Then the extracted coefficients are compared with the stored set saved in the database by using Euclidean distance measure. The best match determines the recognized marking.

This detection and recognition approach is robust and fast. It achieves 20 frames/sec on a 450 MHz PC. The system detects and discriminates between dozens of uniquely marked landmarks.

Distance to tracker, Fiducial System

The theoretical uncertainty per meter distance is dependent on the system setup: camera resolution and lens horizontal field of view. For a 1024x768 camera with a large-angle lens of 90^o horizontal field of view we have about 2 mm uncertainty per meter distance to the fiducial. If we want centimeter accuracy we have to have $d<5$ m. In less optimal cases, where the angle between the two is smaller, the maximum distance will be even less.

Pose Computation

Compared with other optical tracking systems which use conventional calibration techniques and tracking mechanisms that need re-initalization, the proposed system is easy to use and flexible. It advances 3D computer vision one step from laboratory environments to real world use. The proposed 6-DOF tracking system is scalable, and has registration errors of about 1cm for distances within 2m between camera and landmarks. Maximum update rate obtained on a system with P3 at 600MHz was 15 Hz.

It is well known that the rotational acceleration of head motion can reach high gradients. Thus, using only one camera for pose estimation will be acceptable if only small changes between two images occur. Integration with an inertial measurement unit will overcome speed problems.

Using binocular images will improve accuracy and speed, because pose estimation can be done in almost every frame. Stereo pose estimation makes position estimation easier and more stable because it adds one more constraint, that is the known geometric relation between the cameras. In future work we intend to study system performance improvement when using a stereo configuration, compared to one camera configuration.

Chapter 6

Conclusion

In this work we have thoroughly discussed the problem of Mobile User Positioning in the context of Augmented Reality applications, the problem of state estimation of noisy dynamic systems using Kalman Filters, and finally how to apply Kalman Filter techniques to solve the Mobile System Localization problem. In order of appearance, we discussed the following topics.

In **Chapter 2** we reviewed a variety of existing techniques and systems for position determination. From a practical point of view, we discussed the need for a mobile system to localize itself while navigating through an environment. We identified two different localization instantiations, position tracking and global localization. In order to determine what information a mobile system has access to regarding its position, we discussed different sources of information and pointed out advantages and disadvantages. We concluded that due to the imperfections in actuators and sensors due to noise sources, a navigating mobile system should localize itself using information from different sensors.

In **Chapter 3**, based on the analysis of the technologies presented in the Chapter 2 and the sensors described in this chapter, we selected a set of sensors from which to acquire and fuse the data in order to achieve the required robustness and accuracy. We selected for the inertial system three accelerometers (ADXL105) and three gyroscopes (Murata ENC05). To correct for gyro drift we use a TCM2 sensor that contains a two-axis inclinometer and a three-axes magnetometer (compass). Indoors we use a Firewire webcam to obtain the position and orientation information. Outdoors we use, in addition, a GPS receiver in combination with a radio data system (RDS) receiver to obtain DGPS correction information.

The calibration of accelerometers, angular rate sensors, and inertial measurement unit increase the accuracy of their measurement. In order for the system to operate properly, it is imperative that the null point and scale factor of each individual component of the IMU sensor (accelerometers and gyros) are determined. Temperature compensation uses a look-up table based on the apparent relation between drift rate and sensor temperature. This compensation provides a notable improvement. From an initial drift of 9 degree/second the drift dropped down to 5-10 degree/minute. For this reason, the system comes with a digital A/D channel for a temperature sensor that can be read during operation. However, we found out that even with a drift look-up table the resulting bias-drift estimate is not accurate enough. The thermal bias-drift rate of the accelerometer placed at room temperature was found by experiments to be 0.108μg/s. The effect of the random bias drift on the velocity and position error is quite important. The bias deviation is about 2-3mg for the entire accelerometer measurement range.

Chapter 4 was concerned with development of inertial equations required for the navigation of a mobile system. To understand the effect of error propagation, the inertial equations were linearized. In this chapter we decompose the localization problem into attitude estimation and, subsequently, position estimation. We focus on obtaining a good attitude estimate without building a model of

the vehicle dynamics. The dynamic model was replaced by gyro modeling. An Indirect (error state) Kalman filter that optimally incorporates inertial navigation and absolute measurements was developed for this purpose. The linear form of the system and measurement equations for the planar case derived here allowed us to examine the role of the Kalman filter as a signal processing unit. The extension of this formulation to the 3D case shows the same benefits. A tracking example in the 3D case was also shown in this chapter.

In this chapter we thoroughly discussed the basics of the Kalman Filter. We looked at the assumptions that the Kalman Filter poses on the system whose state it estimates: a linear dynamic system with linearly related measurements, corrupted by Gaussian distributed, white, zero-mean noise. There are a number of remarkable advantages one will notice just in the form of the filter. First off all, the Kalman gain is computed from scratch each time you wish to incorporate a new measurement. This makes it very easy to track systems with time-varying dynamics or measurement processes. The ability to handle time-varying models is also the key to using the Kalman filter with nonlinear systems or nonlinear measurement models.

Derivation of EKF using quaternions is a novel approach. The advantage of quaternion representation is that since the incremental quaternion corresponds very closely to a small rotation, the first component will be close to unity and thus the attitude information of interest is contained in the three vector components of the quaternion. The equations used to update the Euler rotations of the body with respect to the chosen reference frame are implemented using a 4th order Runge-Kuta integration algorithm, since a closed form solution does not exist. The quaternion implementation appears to be performing better than the Euler angle implementation. Quaternion implementation converged to 1% from the maximum error range faster than the Euler angle. The steady state bias error for quaternion EKF is also slightly lower than the steady state for Euler angle EKF.

Quaternion EKF implementation in real time represents another issue. After the filter implementation was proven to be stable and running on an Intel PC platform, the next step was to reduce filter computation complexity, but try to maintain filter performance. A separate bias filter formulation was implemented and run on the LART platform almost in real time, providing an update rate of about 80Hz.

Chapter 5 details all the necessary steps for implementing a vision positioning system. The pose tracking system for outdoor augmented reality is partly based on a vision system that tracks the head position within centimeters, the head orientation within degrees, and has an update rate of within a second. The algorithms that are necessary to obtain a robust vision system for tracking the automotion of a camera based on its observations of the physical world contain feature detection algorithms, camera calibration routines and pose determination algorithms.

In this chapter, we have presented a flexible technique to easily calibrate a camera. Compared with conventional techniques, which use expensive equipment such as two or three orthogonal planes, the proposed technique offers considerable flexibility. The proposed preprocessing (data normalization) of data points before calibration makes the algorithm stable to pixel noise.Through experiments, we found that Tsai's method yielded the most accurate results when trained on data of low measurement error. Thus, in the absence of sophisticated measurement apparatus, our planar calibration results may easily outperform those of Tsai.

For each detected landmark, we first extract the contour and compute Hu moments. Then the extracted coefficients are compared with the stored set saved in the database by using Euclidean distance measure. The best match determines the recognized marking. This detection and recognition approach is robust and fast. It achieves 20 frames/sec on a 450 MHz PC. The system detects and discriminates between dozens of uniquely marked landmarks. The proposed 6-DOF tracking

system is scalable, and has registration errors of about 1cm for distances within 2m between camera and landmarks. Maximum update rate obtained on a system with P3 at 600MHz was 15 Hz.

In order to achieve the real-time (video-rate) tracking of features, we have investigated how these real-time constraints can be resolved. We have investigated two alternatives, both based on special hardware. One is based on special multimedia instructions in general purpose processors, and the other on special purpose image processing hardware, based on a solid paradigm, SIMD. Our conclusion is that the IMAP-Vision system performs better than the multimedia extensions. The MMX-type Streaming SIMD multimedia extensions for general purpose processors remain good candidates for low-level image processing, however provisionally not in low-power embedded systems for wearable computing.

6.1 Contributions

The main contributions presented in this thesis are as follows:

- We present an overview of position measurement technology, with both advantages and disadvantages.
- We present sensors that are often used in pose determination with their advantages and disadvantages. Based on the requirements formulated for Augmented Reality Applications, we select some and combine them in an Inertial Measurement Unit.
- Since existing technology or sensor alone cannot solve the pose problem, we combine information from multiple sensors to obtain a more accurate and stable system. The integration is achieved using a Kalman filter. We present the formulation for a new Kalman filter implementation based on quaternions.
- We present the development of an entire position determination system using off-the-shelve existing sensors integrated using separate Kalman filters. Where the research and implementation were not complete due to the time constraint we provide simulations to prove the validity of the concept. Still, a unified solution is presented: inertial measurement integration for orientation and GPS in combination with a differential correction unit for positioning. The accuracy obtained is 0.5 degrees for orientation, at an update rate of 100 Hz, and 5 m accuracy for positioning at 1 Hz.
- We present all the necessary steps for implementing a vision positioning system. The integration with the other sensors system is left to future research.

6.2 Future Research Directions

This work can be used as a theoretical basis for further studies in a number of different directions. First, in the localization chapters we discussed some positioning methods implementing the localization formula. They give a starting point for further readings into the different positioning methods. Second, also the KF chapters form a basis for further studies. With the basic derivations of the KF and some of the extensions, it is interesting to look more detailed into other extensions, relaxing more assumptions. Third, the experiments we performed in the chapters on KFs and localization were performed in a simulator and merely meant as illustrations of concepts relating to the KF. In order to draw conclusions on the practical utility of the KF, the theory discussed in these chapters can be applied in practice.

Throughout the progress of this work, interesting ideas for future work came up. One of these is to look for ways to make the KF applicable in dynamic environments, since the real world is not static.

In this work we discussed the use of KFs and positioning of a mobile system using static landmarks whose location it knows with some uncertainty. It is interesting to look at what would happen if the landmarks are not static, but move through the environment according to some motion model.

Position tracking using object and scene recognition remains for future research. The idea was that, given a 3D description of the environment (e.g. a CAD-model) and an initial position estimate, an accurate position could be calculated iteratively. In order to locate a 3D object in an image, two general strategies can be applied. Firstly, the inherent 3D representation of the object can be matched, which leads to a 3D to 2D matching problem. Secondly, a 2D representation of the object can be applied, which leads to a 2D-to-2D mapping problem. Object recognition requires the extraction of suitable features, which have to be available both in image and model data. A 3D GID geo-database must be developed that is capable of storing complex geometric and/or topological models.

It is well known that the rotational acceleration of head motion can reach high gradients. Thus, using only one camera for pose estimation will be acceptable if only small changes between two images occur. Integration with an inertial measurement unit will overcome speed problems. Using binocular images will improve accuracy and speed, because pose estimation can be done in almost every frame. Stereo pose estimation makes position estimation easier and more stable because it adds one more constraint, i.e. the known geometric relation between the cameras. In future work we intend to study system performance improvement when using a stereo configuration, compared to one camera configuration.

Bibliography

[1] Interactive Imaging Systems. http://www.iisvr.com/99iis/Products/vfx3d/VFX3D.htm.

[2] Gyro Point, Inc. http://www.gyropoint.com.

[3] Analog Devices, Inc. http://content.analog.com/pressrelease/prprintdisplay/

[4] Feiner, S., MacIntyre, B., H.llerer, T., Webster, A. (1997). A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. Proceedings ISWC'97 (International Symposium on wearable computing (Cambridge,MA, October 13-14), www.cs.columbia.edu/graphics/publications/ISWC97.ps.gz.

[5] Azuma, R., Baillot, Y., Behringer,R., Feiner, S., Julier, S., MacIntyre, B. (2001). Recent Advances in Augmented Reality. IEEE Computer Graphics and Applications , pp. 34–47.

[6] Azuma, R. A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments, 6(4), pp. 355 - 385.

[7] Caudell, T. P. and Mizell, D. W. (1992). Augmented reality: An application of heads-up display technology to manual manufacturing processes. In Proceedings of the 25th Hawaii International Conference on Systems Sciences, pp. 7–10, Kauai, HI.

[8] Ubicom (1997), http://www.ubicom.tudelft.nl.

[9] Janin, A. L., Mizell, D. W., and Caudell, T. P. (1993). Calibration of head-mounted displays for augmented reality applications. In Proceedings of IEEE Virtual Reality Annual Intrinsic Symposium, pp. 246–255, New York, NY.

[10] Taylor, R., Gruben, K., LaRose, D., and Gomory, S. (1994). Image guided command and control of a surgical robot. In Proceedings of Medicine Meets Virtual Reality II, San Diego, CA.

[11] Meyer, K., Applewhite, H. L., and Biocca, F. A. (1992). A survey of position trackers. Presence: Teleoperators and Virtual Environments,1(2), pp. 173–200.

[12] Christine Youngblut, Rob E. Johnson Sarah H. Nash, Ruth A. Wienclaw, Craig A.Will, Review of Virtual Environment Interface Technology, INSTITUTE FOR DEFENSE ANALYSES

[13] J. Borenstein, H.R. Everett, L. Feng, (1996). Where am I? Sensors and Methods for Mobile Robot Positioning. University of Michigan. http://www-personal.engin.umich.edu/~johannb/position.html

[14] Yohan Baillot, Jannick Rolland, (1996). Fundamental Principles of Tracking Technology for Virtual Environments. TR96-004,November, Center for Research and Education in Optics and Lasers, Department of Electrical Engineering and Department of Computer Sciences, University of Central Florida.

[15] Eric Foxlin, Michael Harrington, and George Pfeifer, (1998). Constellation™: A Wide - Range Wireless Motion -Tracking System for Augmented Reality and Virtual Set Applications. InterSense Incorporated.

[16] InterSense IS-600 User Manual. http://www.isense.com/products/prec/is600.

[17] Ascension Technology Corp. http://www.ascension-tech.com/products/miniBird/minibird.htm.

[18] Crossbow Technology, Inc. http://www.xbow.com/html/gyros/dmu6x.htm.

[19] Garmin GPS 25 Technical Specification. http://www.garmin.com/products/gps25.

[20] TCM2 Technical specifications. http://www.pnicorp.com.

[21] Abidi, M. and Chandra, T., (1990). Pose Estimation for Camera Calibration and Landmark Tracking, Proceedings of IEEE International Conference on Robotics and Automation, Cincinnati, OH, May 13-18, pp. 420-426.

[22] Abidi, M. and Gonzalez, R., Editors, (1992). Data Fusion in Robotics and Machine Intelligence.Academic Press Inc., San Diego, CA.Acuna, M.H. and Pellerin, C.J., "A Miniature Two-Axis Fluxgate Magnetometer." IEEE Transactions on Geoscience Electronics, Vol. GE-7, pp. 252-260.

[23] Adams, M.D., (1992). Optical Range Data Analysis for Stable Target Pursuit in Mobile Robotics. Ph.D. Thesis, Robotics Research Group, University of Oxford, U.K.

[24] N. I. Durlach, A. S. Mavon, (1993). Virtual Reality: Scientific and Technological Challenges. Washington, D.C., National Academy Press.

[25] S. Bryson, S.S. Fisher, (1990). Defining, Modeling and Measuring System Lag in Virtual Environments. Proceedings of SPIE - The International Society of Optical Engineering Vol. 1256, Bellington, WA, pp. 98-109.

[26] K. Chervest, et. al., (2000). Developing a Context-aware Electronic Tourist Guide: some Issues and Experiences. Proceedings of CHI, Netherlands, pp. 17-24.

[27] W. W. Smith, (1991). Passive Location of Mobile Cellular Telephone Terminals. IEEE 1991 Proc. on Security Technology, pp. 221-225.

[28] S. Sakagami, S. Aoyama, K. Kuboi, S. Shirota, A. Akeyama, (1992). Vehicle Position Estimates by Multibeam Antennas in Multipath Environments. IEEE Trans. on Vehicular Technology, Vol. 41, No. 1, pp. 63-67.

[29] G. L. Turin, W. S. Jewell, T. L. Johnston, (1972). Simulation of Urban Vehicle-Monitoring Systems. IEEE Trans. on Vehicular Technology, Vol. VT-21, No. 1, pp. 9-16.

[30] W. C. Y. Lee, (1989). Mobile Cellular Telecommunications Systems. McGraw-Hill Book Company, New York.

[31] Intel Corporation, (1997). Intel Architecture MMX Technology Developer's Manual, Intel Corporation. Available at http://www.intel.com.

[32] Intel Corporation, (1997). MMX Technology Programmers Reference Manual, Intel Corporation. Available at http://www.intel.com.

[33] Intel Corporation, Using the RDTSC Instruction for Performance Monitoring, Available at http://developer.intel.com/drg/pentiumII/appnotes/RDTSCPM1.HTM.

[34] Y. Fujita et al., (1996). IMAP-VISION: An SIMD Processor with High-Speed On-chip Memory and Large Capacity External Memory. Proc. of IAPR Workshop on Machine Vision Applications (MVA), pp. 170-173.

[35] S. Kyo and K. Sato, (1996). Efficient Implementation of Image Processing Algorithms on Linear Processor Arrays using the Data Parallel Language 1DC. Proc. of IAPR Workshop on Machine Vision Applications (MVA), pp. 160-165.

[36] S. Kyo, T. Koga, and S. Okazaki, (2001). IMAP-CE: A 51.2 Gops Video Rate Image Processor with 128 VLIW Processing Elements. In Proc. Int. Conf. Image Processing, pp. 294-297.

[37] M. van der Molen and S. Kyo, (1997). Documentation for the IMAP-VISION image processing card and the 1DC language. NEC Incubation Center. http://www.incx.nec.co.jp/doc/englishIMAPand1DC.pdf.

[38] A.A. Abbo, R.P. Kleihorst, L.Sevat, P. Wielage, R. van Veen, M.J.R. op de Beeck, and A. van der Avoird, (2001). A low-power parallel processor IC for digital video cameras. in Pro-

ceedings of the 27th European Solid-State Circuits Conference, Villach, Austria. Carinthia Tech Institute.

[39] P. P. Jonker, (1990). Architectures for Multidimensional Low - and Intermediate Level Image Processing. Proc. of IAPR Workshop on Machine Vision Applications (MVA), pp. 307-316.

[40] J.G.E. Olk and P.P. Jonker, Bucket processing: A paradigm for image processing, ICPR13, Proc. 13th Int. Conf. on Pattern Recognition (Vienna, Austria, Aug.25-29) Vol. 4, IEEE Computer Society Press, Los Alamitos

[41] J.G.E. Olk and P.P. Jonker, (1997). Parallel image processing using distributed arrays of buckets. Pattern Recognition and Image Analysis, vol. 7, no. 1, pp. 114-121.

[42] W. Foerstner, (1994). A framework for low-level feature extraction. In Proceedings of the European Conference on Computer Vision, vol. II, pp. 383-394.

[43] S. Bougnoux, (1998). From projective to euclidean space under any practical situation, a criticism of self-calibration. In Proceedings of the 6th International Conference on Computer Vision, pp. 790–796.

[44] D. C. Brown, (1971). Close-range camera calibration. Photogrammetric Engineering, 37(8), pp. 855–866.

[45] B. Caprile and V. Torre, (1990). Using Vanishing Points for Camera Calibration. The International Journal of Computer Vision, 4(2), pp.127–140.

[46] W. Faig, (1975). Calibration of close-range photogrammetry systems: Mathematical formulation. Photogrammetric Engineering and Remote Sensing, 41(12), pp. 1479–1486.

[47] O. Faugeras, (1993). Three-Dimensional Computer Vision: a Geometric Viewpoint. MIT Press.

[48] E. Trucco and A. Verri: Introductory Techniques for 3-D Computer Vision, Prentice-Hall, 1998, ISBN 0-13-261108-2

[49] O. Faugeras, T. Luong, and S. Maybank, (1992). Camera self-calibration: theory and experiments. In G. Sandini, editor, Proc 2nd ECCV, volume 588 of Lecture Notes in Computer Science, Santa Margherita Ligure, Italy, Springer-Verlag, pp. 321–334.

[50] O. Faugeras and G. Toscani, (1986). The calibration problem for stereo. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, pp. 15–20.

[51] S. Ganapathy, (1984). Decomposition of transformation matrices for robot vision. Pattern Recognition Letters, 2, pp. 401–412.

[52] D. Gennery, (1979). Stereo-camera calibration. In Proceedings of the 10th Image Understanding Workshop, pp. 101–108.

[53] G. Golub and C. van Loan, (1996). Matrix Computations. The John Hopkins University Press, Baltimore, Maryland, 3rd edition.

[54] R. Hartley and A. Zisserman, (2004). Multiple View Geometry in Computer Vision, second edition, Cambridge University Press, ISBN: 0521540518.

[55] R. Hartley, (1995). In defence of the 8-point algorithm. In Proceedings of the 5th International Conference on Computer Vision. Boston, MA. IEEE Computer Society Press, pp. 1064–1070,.

[56] Zhang, Z., (1999). Flexible camera calibration by viewing a plane from unknown orientations. The Proceedings of the Seventh IEEE International Conference on Computer Vision, pp. 666-673.

[57] D. Liebowitz and A. Zisserman, (1998). Metric rectification for perspective images of planes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, California. IEEE Computer Society, pp. 482–488.

[58] Heikkila, J., Silven, O., (1997). A four-step camera calibration procedure with implicit image correction. Computer Vision and Pattern Recognition, pp. 1106-1112.

[59] J. More, (1977). The levenberg-marquardt algorithm, implementation and theory. InG.A.Watson, editor, Numerical Analysis, Lecture Notes in Mathematics 630. Springer-Verlag.

[60] I. Shimizu, Z. Zhang, S. Akamatsu, and K. Deguchi, (1998). Head pose determination from one image using a generic model. In Proceedings of the IEEE Third International Conference on Automatic Face and Gesture Recognition, Nara, Japan, pp. 100–105.

[61] A. State, G. Hirota, D. T. Chen, B. Garrett, M. Livingston. Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking. Proc. of SIGGRAPH'96, pp. 429-438, 1996.

[62] S. Brantner, T. Auer, A. Pinz. Real-Time Optical Edge and Corner Tracking at Subpixel Accuracy. Proceedings of 8th International Conference on Computer Analysis of Images and Patterns CAIP '99. Springer, F. Solina, A. Leonardis (eds.), 1999. pp. 534-541.

[63] M. Billinghurst and H. Kato. Collaborative Mixed Reality. Prof. of the first international sysposium on Mixed Reality, pp. 261 – 284, 1999.

[64] U. Neumann and Y. Cho. A Self-Tracking Augmented Reality System. Proc. of ACM Virtual Reality Software and Technology, pp. 109-115, 1996.

[65] R.Y. Tsai, (1987). 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, 3(4), pp. 323–344.

[66] G.Wei and S. Ma, (1993). A complete two-plane camera calibration method and experimental comparisons. In Proc. Fourth International Conference on Computer Vision, Berlin, pp. 439–446.

[67] P.H.S. Torr and A. Zisserman, (1999). Feature Based Methods for Structure and Motion Estimation. In International Workshop on Vision Algorithms, pp. 278-295.

[68] J. Weng, P. Cohen, and M. Herniou, (1992). Camera calibration with distortion models and accuracy evaluation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(10), pp. 965–980.

[69] M. Hu. Visual Pattern Recognition by Moment Invariants, IRE Transactions on Information Theory, 8:2, pp. 179-187, 1962.

[70] Daniel F. DeMenthon and Larry S. Davis, (1992). Model-Based Object Pose in 25 Lines of Code. In Proceedings of ECCV '92, pp. 335-343.

[71] S. Persa and P.P. Jonker, (2001). Real Time Image Processing Architecture for Robot Vision, Best Presentation Paper Award, in: David P. Casasent (eds.), Intelligent Robots and Computer Vision XX: Algorithms, Techniques, and Active Vision (Proc. Conf. Boston, USA, 28 October - 2 November 2001), Proc. SPIE, vol. 4572, IBSN 0-8194-4300X, pp. 105-115.

[72] S. Persa and P.P. Jonker, (2001). Multisensor Robot Navigation Systems, in: Douglas W. Cage (eds.), Mobile Robots XVI (Proc. Conf. Boston, USA, 28 October - 2 November 2001), Proc. SPIE, vol. 4573.

[73] W. Pasman, S. Persa and F.W. Jansen, (2001). Realistic Low-Latency Mobile AR Rendering, in: Proc. International Symposium on Virtual and Augmented Architecture (VAA01) (Proc. Conf. Dublin, Ireland June 21-22), LNCS, IBSN 1-85233-456-8.

[74] S. Persa and P.P. Jonker, (2001). Hybrid Tracking System for Outdoor Augmented Reality, in: R. L. Lagendijk, J.W.J. Heijnsdijk, A.D. Pimentel, M.H.F. Wilkinson (eds.), Proc. ASCI

2001, 7th Annual Conf. of the Advanced School for Computing and Imaging (Heijen, The Netherlands, May 30- June 1), pp. 162-167.

[75] S. Persa and P.P. Jonker, (2000). Evaluation of Two Real Time Image Processing Architectures, in: L.J. van Vliet, J.W.J. Heijnsdijk, T. Kielman, P.M.W. Knijnenburg (eds.), Proc. ASCI 2000, 6th Annual Conf. of the Advanced School for Computing and Imaging (Lommel, Belgium, June 14-16), ASCI, Delft, pp. 387-392.

[76] S. Persa and P.P. Jonker, (2000). Human-computer Interaction using Real Time 3D Hand Tracking, in: J. Biemond (eds.), Proc. 21st Symposium on Information Theory in the Benelux (Wassenaar, NL, May 25-26), pp. 71-75.

[77] S. Persa and P.P. Jonker, (2000). Hybrid Tracking System for Outdoor Augmented Reality, in: R.L. Lagendijk, R. Heusdens, W.A. Serdijn, H. Sips (eds.), Proc. 2nd Int. Symposium on Mobile Multimedia Systems and Applications MMSA2000 (Delft, Nov.9-10), pp. 41-47.

[78] S. Persa and P.P. Jonker, (2000). On Positioning for Augmented Reality Systems, Proc. Signal Processing Symposium 2000 (SPS2000), IEEE Benelux Signal Processing Chapter (Hilvarenbeek, NL, March 23-24).

[79] S. Persa and P.P. Jonker, (2000). Real Time Image Processing Architecture for Robot Vision, in: David P. Casasent (eds.), Intelligent Robots and Computer Vision XIX: Algorithms, Techniques, and Active Vision (Proc. Conf. Boston, USA, Nov.7-8), Proc. SPIE, vol. 4197, pp. 221-228.

[80] S. Persa, C. Nicolescu, and P.P. Jonker, (2000). Evaluation of two real time low-level image processing architectures, Proc. IAPR Workshop on Machine Vision Applications (Tokyo, Japan, Nov.28-30), pp. 295-298.

[81] S. Persa and P.P. Jonker, (1999). On positioning for augmented reality systems, in: H.-W. Gellersen (eds.), Handheld and Ubiquitous Computing (Proc. HUC'99, 1st Int. Symposium, Karlsruhe, Germany, September 27-29), Lecture Notes in Computer Science, vol. 1707, Springer, Berlin, pp. 327-329.

[82] S. Persa and P.P. Jonker, (1999). Real-time image processing using the IMAP-VISION board, in: Z. Houkes (eds.), Abstracts of the 2nd ASCI Imaging workshop (Enschede, Mar.29-31), pp. 47-48.

[83] Jack B. Kuipers: Quaternions and Rotations Sequences, Princeton University Press, Princeton, New Jersey 1998.

[84] Systron Donner, (1998). BEI GyrochipII Product Specifications. http:\\www.systron.com\gyroiids.pdf.

[85] Murata, (1999). Piezoelectric Vibrating Gyroscope ENC Series. Catalog S42E-2. http://www.murata.com/catalog/s42e2.pdf.

[86] Karl. J Astrom, (). Computer Controlled System Theory and Design. ISBN 0-13-314899-8

[87] D.H. Titterton and J.L. Weston, (). Strapdown inertial navigation technology, Peter Peregrinus Ltd

[88] Robert Grover Brown and Patrick Y.C Hwang, (). Introduction to Random Signals and applied Kalman Filtering. Wiley, ISBN 0-471-12839-2

[89] Peter S.Maybeck, (1979). Stochastic Models, Estimation and Control Volume 1. Navateck Books and Software.

[90] Mohinder S. Grewal, Lawrence R. Weill and Angus P. Andrews, (2001). Global Positioning Systems, Inertial Navigation, and Integration. Willey-Interscience, ISBN 0-471-35032-X.

[91] B. Friedland, (1969). Treatment of Bias in Recursive Filtering. IEEE Transactions on Automatic Control, vol. AC-14, pp. 359-367.

[92] P. Zarchan and H. Musoff, (2000). Fundamentals of Kalman Filtering, A Practical Approach. Progress in Astronautics and Aeronautics Series, Published by AIAA, ISBN: 1-56347-455-7.

[93] Farrell, J. A and M. Barth. The Global Positioning System and Inertial Navigation: Theory and Practice. McGraw-Hill, 1999.

[94] Savage, P., (1978). Strapdown Sensors. NATO AGAARD Lecture Series No. 95.

[95] Leick, A., (1995). GPS Satellite Surveying. John Wiley and Sons, Inc., Toronto.

[96] Bakker,J.D., Mouw, E.,Joosen,M., Pouwelse,J., (2000). The LART Pages. Delft University of Technology, Faculty of Information Technology and Systems Available Internet:http://www.lart.tudelft.nl.

[97] Douglas D., Peucker T., (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer 10(2), pp. 112-122

[98] Kaplan, Elliot D., ed., Understanding GPS Principles and Applications, Artech House Publishers, Boston, 1996.

[99] Stergios I. Roumeliotis, Gaurav S. Sukhatme, George A. Bekey. Circumventing Dynamic Modeling Evaluation of the ErrorState Kalman Filter applied to Mobile Robot Localization, 1999 IEEE International Conference on Robotics and Automation

Chapter A

Appendix

A.1 Cholesky Decomposition

This decomposition is named after Andre Louis Cholesky, who discovered the method for factoring a symmetric, positive-definite matrix P as a product of triangular factors.

A Cholesky factor of a symmetric positive-definite matrix P is a matrix C such that:

$$CC^T = P \qquad \text{eq. A-1.}$$

Cholesky factors are not unique. If C is a Cholesky factor of P, then for any conformable orthogonal matrix M, the matrix:

$$A = CM \qquad \text{eq. A-2.}$$

satisfies the equation:

$$AA^T = CM(CM)^T = CMM^TC^T = CC^T = P \qquad \text{eq. A-3.}$$

One possible disadvantage of the CC^T decomposition is the need to take square roots. This is easily avoided by using the LDL^T factorization, where L is a lower triangular with 1's on the diagonal, and D is a diagonal matrix. In terms of the Cholesky factor P, this is done by defining, for i > j,

$$L_{i,j} = P_{i,j}/P_{j,j} \qquad D_{i,i} = P_{i,i}^2 \qquad \text{eq. A-4.}$$

which can be calculated as follows:

For j:=1 to n do:

$$D_{j,j} = A_{j,j} - \sum_{k=1}^{j-1} L_{j,k}^2 D_{k,k} \qquad \text{eq. A-5.}$$

For i:=j+1 to n do:

$$L_{i,j} = \left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k} D_{k,k} \right) / D_{j,j} \qquad \text{eq. A-6.}$$

A.2 Direction Cosine Matrix

A.2.1 Propagation of a DCM with time

In order to update a DCM it is necessary to solve a matrix differential equation of the form:

$$\dot{C}_b^n = C_b^n \Omega \qquad \Omega = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \qquad \text{eq. A-7.}$$

Over a single computer cycle, the solution of the differential equation may be written as follows:

$$C_b^n(t_{k+1}) = \int_{t_k}^{t_{k+1}} \dot{C}_b^n(t_k)dt = C_b^n(t_k)\exp\left(\int_{t_k}^{t_{k+1}} \Omega dt\right) \qquad \text{eq. A-8.}$$

If the orientation of the turn rate vector, $\omega = [p, q, r]$ remains fixed in space over the update interval T, we can write the integral as ($[\sigma\times]$ is a notation for the following matrix):

$$\int_{t_k}^{t_{k+1}} \Omega dt = [\sigma\times] = \begin{bmatrix} 0 & -\sigma_z & \sigma_y \\ \sigma_z & 0 & -\sigma_x \\ -\sigma_y & \sigma_x & 0 \end{bmatrix} \qquad where \qquad \begin{cases} \sigma_x = \int_{t_k}^{t_{k+1}} pdt \\ \sigma_y = \int_{t_k}^{t_{k+1}} qdt \\ \sigma_z = \int_{t_k}^{t_{k+1}} rdt \\ \sigma = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} \end{cases} \qquad \text{eq. A-9.}$$

Expanding the exponential term in Equation A-8 gives ($[\sigma\times]$ depends on k):

$$B_k = \mathbf{I} + [\sigma\times] + \frac{[\sigma\times]^2}{2!} + \frac{[\sigma\times]^3}{3!} + \frac{[\sigma\times]^4}{4!} + \ldots \qquad \text{eq. A-10.}$$

Using the fact that $[\sigma\times]$ is a skew symmetric matrix and using Equation A-9 it can be shown [87]:

$$[\sigma\times]^2 = \begin{bmatrix} -(\sigma_y^2 + \sigma_z^2) & \sigma_x\sigma_y & \sigma_x\sigma_z \\ \sigma_x\sigma_y & -(\sigma_x^2 + \sigma_z^2) & \sigma_y\sigma_z \\ \sigma_x\sigma_z & \sigma_y\sigma_z & -(\sigma_x^2 + \sigma_y^2) \end{bmatrix} \qquad \text{eq. A-11.}$$

$$[\sigma\times]^3 = -(\sigma_x^2 + \sigma_y^2 + \sigma_z^2)[\sigma\times]$$

$$[\sigma\times]^4 = -(\sigma_x^2 + \sigma_y^2 + \sigma_z^2)[\sigma\times]^2$$

Then the exponential expansion can be written [87]:

$$B_k = \mathbf{I} + \left(1 - \frac{\sigma^2}{3!} + \frac{\sigma^4}{5!} - \ldots\right)[\sigma\times] + \left(\frac{1}{2!} - \frac{\sigma^2}{4!} + \frac{\sigma^4}{6!} - \ldots\right)[\sigma\times]^2 + \ldots \qquad \text{eq. A-12.}$$

$$= \mathbf{I} + \frac{\sin\sigma}{\sigma}[\sigma\times] + \frac{(1 - \cos\sigma)}{\sigma^2}[\sigma\times]^2$$

Provided that σ is the angle vector as defined above, Equation A-12 provides an exact representation of the attitude matrix which relates to body attitude at times t_{k+1} and t_k.

Normalization of DCM

If C_b^n matrix is updated a great deal this introduces rescaling of the rows vectors. The row vectors will after a while not be orthogonal to each other. This can be caused by the floating point arithmetic. The matrix can be made orthogonal using:

$$C_b^n = \tilde{C}_b^n - 0.5(\tilde{C}_b^n \tilde{C}_b^{nT})\tilde{C}_b^n \qquad \text{eq. A-13.}$$

A.3 Quaternion

A.3.1 Multiplication

The product of two quaternions,

$$\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \qquad \mathbf{e} = e_0 + e_1\mathbf{i} + e_2\mathbf{j} + e_3\mathbf{k} \qquad \text{eq. A-14.}$$

may be derived by applying the usual rules for products of complex numbers (if we assume $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$):

$$\begin{aligned}\mathbf{q} \bullet \mathbf{e} &= (q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k})(e_0 + e_1\mathbf{i} + e_2\mathbf{j} + e_3\mathbf{k}) \\ &= (q_0e_0 - q_1e_1 - q_2e_2 - q_3e_3) \\ &+ (q_0e_1 + q_1e_0 + q_2e_3 - q_3e_2)\mathbf{i} \\ &+ (q_0e_2 - q_1e_3 + q_2e_0 + q_3e_1)\mathbf{j} \\ &+ (q_0e_3 + q_1e_2 - (q_2e_1 + q_3e_0))\mathbf{k}\end{aligned} \qquad \text{eq. A-15.}$$

Alternatively, the quaternion product may be expressed in matrix form as:

$$\mathbf{q} \bullet \mathbf{e} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} \qquad \text{eq. A-16.}$$

A.3.2 Quaternion from DCM

For small angular displacements, the quaternion elements may be derived [83] using the following relationships:

$$\begin{aligned} q_0 &= \frac{1}{2}\sqrt{1 + c_{11} + c_{22} + c_{33}} \\ q_1 &= \frac{1}{4q_0}(c_{32} - c_{23}) \\ q_2 &= \frac{1}{4q_0}(c_{13} - c_{31}) \\ q_3 &= \frac{1}{4q_0}(c_{21} - c_{12}) \end{aligned} \qquad \text{eq. A-17.}$$

A more comprehensive algorithm for the extraction of quaternion parameters from the direction cosines, which takes into account the relative magnitudes of direction cosines elements, is described in the following pseudocode [83]:

```
if M11 => 0 and (M22 + M33)=> 0 then
        Q1      =       1 + M11 + M22 + M33
        Q2      =       M32 - M23
        Q3      =       M13 - M31
        Q4      =       M21 - M12
if M11 =>0 and (M22 + M33) 0 then
        Q1      =       M32 - M23
        Q2      =       1 + M11 - M22 - M33
        Q3      =       M21 + M12
        Q4      =       M13 + M31
if M11 0 and (M22 - M33) =>0 then
        Q1      =       M13 - M31
        Q2      =       M21 + M12
        Q3      =       1 - M11 + M22 - M33
        Q4      =       M32 + M23
if M11 0 and (M22 - M33) 0 then
        Q1      =       M21 - M12
        Q2      =       M13 + M31
        Q3      =       M32 + M23
        Q4      =       1 - M11 - M22 + M33
```

A.3.3 DCM from Quaternion

The following pseudocode describes the conversion [83]:

```
TX      =       Q2*Q2
TY      =       Q3*Q3
TZ      =       Q4*Q4
TQ      =       TY+TZ

if (TQ + TX + Q1*Q1) is not 0 then
        TK = 2 / (TQ + TX + Q1*Q1)
else
        TK      =       0

M11     =        1 - TK*TQ
M22     =        1 - TK*(TX + TZ)
M33     =        1 - TK*(TX + TY)

TX      =       TK*Q2
TY      =       TK*Q3
TQ      =       (TK*Q4)*Q1
TK      =       TX*Q3

M12     =       TK - TQ
M21     =       TK + TQ
```

```
TQ    =    TY*Q1
TK    =    TX*Q4

M13   =    TK+TQ
M31   =    TK-TQ

TQ    =    TX*Q1
TK    =    TY*Q4

M23   =    TK - TQ
M32   =    TK + TQ
```

A.3.4 Euler angles expressed using Quaternions

The Euler angles may be derived directly from quaternion elements. For conditions where θ is not equal to 90°, the Euler angles can be determined [83] using:

$$\phi = \operatorname{atan}\left\{\frac{2(q_2q_3 + q_0q_1)}{q_0^2 - q_1^2 - q_2^2 + q_3^2}\right\}$$
$$\theta = \operatorname{asin}\{-2(q_1q_3 - q_0q_2)\} \qquad \text{eq. A-18.}$$
$$\psi = \operatorname{atan}\left\{\frac{2(q_1q_2 + q_0q_3)}{q_0^2 + q_1^2 - q_2^2 - q_3^2}\right\}$$

A.3.5 Quaternion expressed in terms of Euler angles

$$\begin{aligned} q_0 &= \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ q_1 &= \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ q_2 &= \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\ q_3 &= \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \end{aligned} \qquad \text{eq. A-19.}$$

A.3.6 Propagation of Quaternion with time

In order to update the Quaternion it is necessary to solve a matrix differnetial equation of a form:

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega_q \cdot \mathbf{q} \qquad \Omega_q = \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \qquad \text{eq. A-20.}$$

The vector $\omega = [p, q, r]$ represents the angular speed or the turn rate vector. Over a single computer cycle, the solution of the differential equation may be written as follows:

$$\mathbf{q}(t_{k+1}) = \int_{t_k}^{t_{k+1}} \mathbf{q}(t_k)dt = \exp\left(\frac{1}{2}\int_{t_k}^{t_{k+1}} \Omega dt\right) \cdot \mathbf{q}(t_k) \qquad \text{eq. A-21.}$$

If the orientation of the turn rate vector, $\omega = [p, q, r]$ remains fixed in space over the update interval T, we can write:

$$\int_{t_k}^{t_{k+1}} \Omega_q dt = [\sigma_q \times] = \begin{bmatrix} 0 & -\sigma_x & -\sigma_y & -\sigma_z \\ \sigma_x & 0 & \sigma_z & -\sigma_y \\ \sigma_y & -\sigma_z & 0 & \sigma_x \\ \sigma_z & \sigma_y & -\sigma_x & 0 \end{bmatrix} \qquad \begin{cases} \sigma_x = \int_{t_k}^{t_{k+1}} p dt \\ \sigma_y = \int_{t_k}^{t_{k+1}} q dt \\ \sigma_z = \int_{t_k}^{t_{k+1}} r dt \\ \sigma = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} \end{cases} \qquad \text{eq. A-22.}$$

Expanding the exponential term in Equation A-21 gives:

$$B_k = \mathbf{I} + [0{,}5\sigma_q \times] + \frac{[0{,}5\sigma \times]^2}{2!} + \frac{[0{,}5\sigma \times]^3}{3!} + \frac{[0{,}5\sigma \times]^4}{4!} + \ldots \qquad \text{eq. A-23.}$$

Using the fact that $[\sigma \times]$ is a skew symmetric matrix and using Equation A-9 it can be shown:

$$[0{,}5\sigma_q \times]^2 = \begin{bmatrix} -\frac{(\sigma_x^2 + \sigma_y^2 + \sigma_z^2)}{4} & 0 & 0 & 0 \\ 0 & -\frac{(\sigma_x^2 + \sigma_y^2 + \sigma_z^2)}{4} & 0 & 0 \\ 0 & 0 & -\frac{(\sigma_x^2 + \sigma_y^2 + \sigma_z^2)}{4} & 0 \\ 0 & 0 & 0 & \frac{-(\sigma_x^2 + \sigma_y^2 + \sigma_z^2)}{4} \end{bmatrix} \qquad \text{eq. A-24.}$$

$$[0{,}5\sigma_q \times]^3 = -\frac{(\sigma_x^2 + \sigma_y^2 + \sigma_z^2)}{4}[0{,}5\sigma_q \times]$$

Then the exponential expansion can be written:

$$B_k = \mathbf{I} + \left(1 - \frac{(0{,}5\sigma)^2}{3!} + \frac{(0{,}5\sigma)^4}{5!} - \ldots\right)[\sigma_q\times] + \left(\frac{1}{2!} - \frac{(0{,}5\sigma)^2}{4!} + \frac{(0{,}5\sigma)^4}{6!} - \ldots\right) + \ldots \quad \text{eq. A-25.}$$

$$= \cos\left(\frac{\sigma}{2}\right)\mathbf{I} + \frac{2}{\sigma}\sin\left(\frac{\sigma}{2}\right)[\sigma\times]$$

Provided that σ is the angle vector as defined above, Equation A-25 provide an exact representation of the attitude matrix which relates to body attitude at times t_{k+1} and t_k.

Thus the equation which propagates the quaternion over time is:

$$\mathbf{q}(t_{k+1}) = \begin{bmatrix} a_c & -a_s\sigma_x & -a_s\sigma_y & -a_s\sigma_z \\ a_s\sigma_x & a_c & a_s\sigma_z & -a_s\sigma_y \\ a_s\sigma_y & -a_s\sigma_z & a_c & a_s\sigma_x \\ a_s\sigma_z & a_s\sigma_y & -a_s\sigma_x & a_c \end{bmatrix} \cdot \mathbf{q}(t_k) \qquad \begin{cases} a_c = \cos\left(\frac{\sigma}{2}\right) \\ a_s = \frac{\sin\left(\frac{\sigma}{2}\right)}{\sigma} \end{cases} \quad \text{eq. A-26.}$$

A.4 System Concepts

A.4.1 Ordinary differential equations

Many systems that evolve dynamically as a function of a continuous-time variable can be modeled effectively by a system of *n*th-order ordinary differential equations. When the applicable equations are linear but time variant, each *n*th-order differential equation is represented as:

$$y^{(n)}(t) + d(t)_1 y^{(n-1)}(t) + \ldots + d(t)_n y(t) = n_1(t)u^{(n-1)}(t) + \ldots + n_n(t)u(t) \quad \text{eq. A-27.}$$

where()$^{(j)}$ denotes the *j*-th time derivative of the term.

When the applicable equations are nonlinear, the *n*th-order differential equation is represented in general as:

$$y^{(n)}(t) = f(y^{(n-1)}(t), \ldots, y(t), u^{(n-1)}(t), \ldots, u(t)) \quad \text{eq. A-28.}$$

Taylor series analysis of a nonlinear equation around a nominal trajectory can be used to provide a linear model described as: linear for local analysis. To solve *n*th-order differential equations for t>=t_0 requires *n* pieces of information (initial conditions) that describe the state of the system at time t_0.

Example: The differential equation for single-channel tangent-plane INS position error due to gyro measurement error is [93]:

$$p^{(3)}(t) + \frac{g}{R}p^{(1)}(t) = g\varepsilon_g(t) \quad \text{eq. A-29.}$$

where *g* is gravitational acceleration, *R* is the earth's radius, and ε_g represents the gyro measurement error.

A.4.1.1 Transfer functions

The transfer function of a time-invariant linear system (meaning all coefficients in Equation A-27 are constant in time) represented above is represented by:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{n_1 s^{n-1} + \ldots + n_n}{s^n + d_1 s^{n-1} + \ldots + d_n} \qquad \text{eq. A-30.}$$

is obtained by applying the Laplace transform to the differential equation and $s = j\omega$.

Example: The transfer function corresponding to the differential equation presented in the previous example is:

$$G(s) = \frac{P(s)}{E_g(s)} = \frac{g}{s\left(s^2 + \frac{g}{R}\right)} \qquad \text{eq. A-31.}$$

which is a low-pass filter. If the gyro error represents high-frequency noise, the position error will be small, but if the gyro error represents a bias error, then the integral nature of the transfer function will result in a large and growing position error.

A.4.1.2 The state space

The state-space representation converts an *n*th-order differential equation into *n*-coupled first-order differential equations. For analysis only, let the transfer function of Equation A-30 and Equation A-31 be decomposed into two separate filtering operations:

$$\begin{aligned} V(s) &= \frac{1}{s^n + d_1 s^{n-1} + \ldots + d_n} U(s) \\ Y(s) &= \left(n_1 s^{n-1} + \ldots + n_n\right) V(s) \end{aligned} \qquad \text{eq. A-32.}$$

The differential equation corresponding to the previous equation, if we define a state vector x such that $x^T = (v, v^{(1)}, \ldots, v^{(n-1)})$, is:

$$\dot{x}(t) = \left[v^{(1)}, \ldots, v^{(n-1)}, v^{(n)}\right] = \left[x_2(t), \ldots, x_n(t), u(t) - \sum_{i=1}^{n} d_i x_{n-i+1}(t)\right] \qquad \text{eq. A-33.}$$

The system output is represented as:

$$y(t) = \sum_{i=1}^{n} n_i v^{(n-1)}(t) = \sum_{i=1}^{n} n_i x_{n-i+1}(t) \qquad \text{eq. A-34.}$$

In matrix notation, the system can be described as:

$$\left.\begin{aligned} \dot{x}(t) &= \mathrm{F}x(t) + \mathrm{G}u(t) \\ y(t) &= \mathrm{H}x(t) \end{aligned}\right\} \qquad \text{eq. A-35.}$$

with:

$$\mathrm{F}=\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & \mathrm{M} & & & \mathrm{M} \\ -d_n & -d_{n-1} & -d_{n-2} & \dots & -d_1 \end{bmatrix}$$

$$\mathrm{G}=\begin{bmatrix} 0 \\ 0 \\ \mathrm{M} \\ 1 \end{bmatrix}$$

$$\mathrm{H}=\begin{bmatrix} n_n, n_{n-1}, \dots, n_1 \end{bmatrix}$$

eq. A-36.

In this representation, F is referred to as the system matrix, G is referred to as the input matrix, and H is referred to as the output matrix. By way of example: A state-space system describing the single-channel error dynamics of a tangent-plane INS is:

$$\begin{bmatrix} \dot{x} \\ \dot{v} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -g \\ 0 & \frac{1}{R} & 0 \end{bmatrix} \begin{bmatrix} x \\ v \\ \phi \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_a \\ \varepsilon_g \end{bmatrix}$$

eq. A-37.

where ε_a and ε_g represent the accelerometer and gyro measurement errors in the navigation frame, respectively.

For discrete-time systems we apply the Z transform instead of the Laplace transform in order to determine the transfer function.

A.4.1.3 Linear stochastic systems and state augmentation

Navigation system error analysis will often result in equations of the form:

$$\begin{aligned} \dot{x}_n(t) &= F(t)x_n(t) + G_n\varepsilon(t) \\ y(t) &= H(t)x_n(t) + \mu(t) \end{aligned}$$

eq. A-38.

where ε and μ represent instrumentation errors and x_n represents the error in the nominal navigation state. It is possible to model the error terms ε and μ as linear dynamic systems:

$$\begin{aligned} \dot{x}_\varepsilon(t) &= F_\varepsilon(t)x_\varepsilon(t) + G_\varepsilon(t)w_\varepsilon(t) \\ \varepsilon(t) &= H_\varepsilon(t)x_\varepsilon(t) + v_\varepsilon(t) \end{aligned}$$

eq. A-39.

and

$$\begin{aligned} \dot{x}_\mu(t) &= F_\mu(t)x_\mu(t) + G_\mu w_\mu(t) \\ \mu(t) &= H_\mu(t)x_\mu(t) + v_\mu(t) \end{aligned}$$

eq. A-40.

driven by the noise process $w_e(t)$, v_e, $w_m(t)$, and v_m, which can be accurately modeled as white noise. By the process of state augmentation, Equation A-39 and Equation A-40 can be combined into the state-space error model,

$$\dot{x}_a(t) = F_a(t)x_a(t) + G_a(t)w_a(t) \qquad \text{eq. A-41.}$$

$$= \begin{bmatrix} F_n & G_n(t)H_\varepsilon(t) & 0 \\ 0 & F_\varepsilon(t) & 0 \\ 0 & 0 & F_\mu(t) \end{bmatrix} x_a + \begin{bmatrix} G_n & 0 & 0 \\ 0 & G_\varepsilon(t) & 0 \\ 0 & 0 & G_\mu(t) \end{bmatrix} \begin{bmatrix} v_\varepsilon \\ w_\varepsilon \\ w_\mu \end{bmatrix}$$

which is driven only by white-noise processes. In these equations, the augmented state is defined as $x_a=[x_n,x_e,x_m]^T$. The measurements of the augmented system are modeled as:

$$\begin{aligned} y(t) &= H_a(t)x_a(t) + v_a(t) \\ &= \begin{bmatrix} H_n(t) & 0 & H_\mu(t) \end{bmatrix} x_a(t) + v_\mu(t) \end{aligned} \qquad \text{eq. A-42.}$$

which are corrupted only by additive white noise.

A.4.2 Linear Systems in Discrete Time

Equation A-35 on page 156 cannot be used in a computer-controlled environment. It is preferred to transform the system matrices into a discrete time system. When the control signal is discrete, $\{u(t_k) : k = ...,-1,0,1, ...\}$. Given the state at the sampling time t_k, the state at some future time t is obtained by solving Equation A-35. The state at time t, with $t_k \le t \le t_{k+1}$, is thus given by:

$$\begin{aligned} x(t) &= e^{F(t-t_k)}x(t_k) + \int_{t_k}^{t} e^{F(t-s)}dsGu(t_k) \\ x(t) &= \Phi(t, t_k)x(t_k) + \Gamma(t, t_k)u(t_k) \end{aligned} \qquad \text{eq. A-43.}$$

as u is constant between the sampling instants. The state vector at a time t is thus a linear function of $x(t_k)$ and $u(t_k)$. If the input u and output y can be regarded as being sampled at the same instants, the system equation of the sampled system at the sampling instants is then:

$$\begin{aligned} x(t_{k+1}) &= \Phi(t_{k+1}, t_k)x(t_k) + \Gamma(t_{k+1}, t_k)u(t_k) \\ y(t_k) &= Hx(t_k) + Du(t_k) \end{aligned} \qquad \text{eq. A-44.}$$

with:

$$\begin{aligned} \Phi(t_{k+1}, t_k) &= e^{F(t_{k+1}-t_k)} \\ \Gamma(t_{k+1}, t_k) &= \int_{t_k}^{t} e^{F(t-s)}dsG \end{aligned} \qquad \text{eq. A-45.}$$

In most cases D = 0. One reason for this is that in computer-controlled systems, the output y is first measured and the control signal $u(t_k)$ is then generated as a function of $y(t_k)$.

For periodic sampling with period h, we have t_k = k*T and the model simplifies to the time-invariant system:

$$\begin{aligned} x(kT+T) &= \Phi x(kT) + \Gamma u(kT) \\ y(kT) &= Hx(kT) + v(kT) \end{aligned} \qquad \text{eq. A-46.}$$

with:

$$\begin{aligned} \Phi &= e^{FT} \\ \Gamma &= \int_0^T e^{Fs} ds G \end{aligned} \qquad \text{eq. A-47.}$$

From Equation A-47 it follows that:

$$\begin{aligned} \frac{d}{dt}\Phi(t) &= F\Phi(t) = \Phi(t)F \\ \frac{d}{dt}\Gamma(t) &= \Phi(t)G \end{aligned} \qquad \text{eq. A-48.}$$

The matrices Φ and Γ therefore satisfy the equation:

$$\frac{d}{dt}\begin{bmatrix} \Phi(t) & \Gamma(t) \\ 0 & I \end{bmatrix} = \begin{bmatrix} \Phi(t) & \Gamma(t) \\ 0 & I \end{bmatrix}\begin{bmatrix} F & G \\ 0 & I \end{bmatrix} \qquad \text{eq. A-49.}$$

where I is a unit matrix of the same dimension as the number of inputs. The matrices $\Phi(T)$ and $\Gamma(T)$ for the sampling period T can therefore be obtained from the block matrix:

$$\begin{bmatrix} \Phi(t) & \Gamma(t) \\ 0 & I \end{bmatrix} = \exp\left(\begin{bmatrix} F & G \\ 0 & I \end{bmatrix} T\right) \qquad \text{eq. A-50.}$$

A.4.2.1 Computation of discrete time matrices

The calculation required to sample a continuous-time system is the evaluation of a matrix exponential and the integration of a matrix exponential. This can be done in different ways, for instance, by using:

- Numerical calculation using Matlab or a dedicated C/C++ library
- Series expansion of a matrix exponential
- Matrix transformation to diagonal or Jordan forms
- Symbolic computer algebra, using programs such as Maple and Mathematica.

If the sampling time h is small, then the matrices $\Phi(h)$ and $\Gamma(h)$ are calculated with the following formulas [86]:

$$\psi = \int_0^T e^{Fs} ds = IT + \frac{FT^2}{2!} + \frac{F^2T^3}{3!} + \ldots + \frac{F^iT^{i+1}}{(i+1)!} + \ldots \qquad \text{eq. A-51.}$$

and

$$\begin{aligned} \Phi &= I + F\psi \\ \Gamma &= \psi G \end{aligned} \qquad \text{eq. A-52.}$$

Computer evaluation can be done using several different numerical algorithms in Matlab.

A.4.2.2 Systems with random inputs

Because of the inevitable presence of modeling error and measurement noise, it is necessary to consider systems with random (nondeterministic) inputs. The model for such a linear discrete-time system with stochastic inputs can be represented as:

$$\begin{aligned} x(k+1) &= \Phi(k)x(k) + \Gamma(k)w(k) \\ y(k) &= H(k)x(k) + v(k) \end{aligned} \qquad \text{eq. A-53.}$$

where $w(t)$ and $v(t)$ are random variables.

The random variable w is called the process noise; the random variable v is called the measurement noise. The designation as random variables implies that although the value of the random variables at some time in the future is not completely predictable, the statistics of the random variables are known. The mean and the covariance of random variables $w(t)$ and $v(t)$ are denoted as:

$$\begin{aligned} \mu_w(t) &= E\langle w(t)\rangle \\ cov[w(t), w(\tau)] &= Q(t, \tau) \\ \mu_v(t) &= E\langle v(t)\rangle \\ cov[v(t), v(\tau)] &= R(t, \tau) \end{aligned} \qquad \text{eq. A-54.}$$

Unless otherwise stated, we will assume that, $u_w = 0$ and $u_v = 0$.

In the analysis that follows, it is often accurate (and convenient) to assume that the process and the measurement noise are independent of the current and the previous states:

$$\begin{aligned} cov[w(t), w(\tau)] &= 0 \qquad for\ \ t \geq \tau \\ cov[v(t), v(\tau)] &= 0 \qquad for\ \ t \geq \tau \end{aligned} \qquad \text{eq. A-55.}$$

and are independent of each other,

$$cov[w(t), v(\tau)] = 0 \qquad for\ \ all\ t \geq \tau \qquad \text{eq. A-56.}$$

A.4.2.3 The discrete input covariance matrix

We have the ideal control signal $\tilde{u}(t)$ with noise $\varepsilon_{u(t)}$.

$$u(t) = \tilde{u}(t) + \varepsilon_{u(t)} \qquad \text{eq. A-57.}$$

The covariance matrix of this noise is Q. When the continuous time system is discretized the Q matrix must be recalculated. There are different methods to calculate the Q_k matrix. In [86] and [89] we find two equations, Equation A-58 and Equation A-59.

$$Q_k = GQG^T T \qquad \text{eq. A-58.}$$

$$Q_k = \frac{1}{2}(FGQG^T F^T + GQG^T)T \qquad \text{eq. A-59.}$$

The exact formula for computing the Q_k matrix is given below. When the dynamics of the system of interest evolve in continuous time, but analysis and implementation are more convenient in discrete time, we require a way of determining a discrete-time model in the form of Equation A-53.

For this, Φ can be determined as in Equation A-52, but in order for $Q_{dw}(k)$ to have an equivalent effect as $Q_w(t)$, w_k must satisfy:

$$w(k) = \int_{kT}^{(k+1)T} e^{F[(k+1)T-s]} G(s) w(s) ds \qquad \text{eq. A-60.}$$

But as a consequence, we need to use a state-augmentation procedure since w_k does not have a white-noise character. Using the model structure of Equation A-41, with $\omega(t)$ being a white-noise process, we have:

$$Q(k) = cov[w(k)] \qquad \text{eq. A-61.}$$

$$\begin{aligned} &= E\left\langle \int_{(k+1)T}^{(k+1)T} \int_{kT}^{kT} \Phi[(k+1)T, s] G(s) w(s) w^T(\tau) G^T(\tau) \Phi[(k+1)T, \tau]^T d\tau ds \right\rangle \\ &= \int_{kT}^{(k+1)T} \Phi[(k+1)T, s] G(s) Q_\omega G^T(\tau) \Phi[(k+1)T, s]^T ds \end{aligned}$$

A common approximate solution to this equation is Equation A-58, which is accurate only when the eigenvalues of F are very small relative to the sampling period T (i.e., $\|FT\| \ll 1$).

A.4.3 Nonlinear Systems

In many applications, either the system dynamic equations or the measurement equations are not linear. It is therefore necessary to consider the extension of the linear optimal estimation equations to nonlinear systems. If the system state is nonlinear, but approximately linear for small perturbations in the state variable, then we can apply the linear estimation theory.

Consider a nonlinear system with dynamics and measurement equation described by:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), t) + \omega(t) \\ y(t) &= h(x(t), t) + v(t) \end{aligned} \qquad \text{eq. A-62.}$$

Although this model is restrictive in assuming additive uncorrelated white-noise processes ω and *v*, the model is typical for the systems found in navigation applications. The signal $u(t)$ is a deterministic signal and *f* and *h* are known smooth functions.

If the function *f* is continuously differentiable, then the influence of the perturbations on the trajectory can be represented by a Taylor series expansion around the nominal trajectory. A trajectory is a particular solution of a stochastic system, that is, with particular instantiations of the random variates involved. A nominal trajectory is a trajectory obtained when the random variates takes their expected values. If the perturbations are sufficiently small relative to the higher order coefficients of the expansion, then one can obtain a good approximation by ignoring terms beyond some order. Let δ denote the perturbations from the nominal:

$$\begin{aligned} \delta x_k &= x_k - x_k^{NOM} \\ \delta z_k &= z_k - h(x_k, k) \end{aligned} \qquad \text{eq. A-63.}$$

so that the Taylor series expansion is:

$$x_k = f(x_{k-1}, k-1) = f(x_{k-1}^{NOM}, k-1) + \left.\frac{\partial f(x, k-1)}{\partial x}\right|_{x = x_{k-1}^{NOM}} \delta x_{k-1} + hot \qquad \text{eq. A-64.}$$

$$\cong x_k^{NOM} + \left.\frac{\partial f(x, k-1)}{\partial x}\right|_{x = x_{k-1}^{NOM}} \delta x_{k-1}$$

If the higher order terms in δx can be neglected, then:

$$\delta x_k = F_k \delta x_{k-1} + w_{k-1} \qquad F_k = \left.\frac{\partial f}{\partial x}\right|_{x = x_{k-1}^{NOM}} = \left.\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}\right|_{x = x_{k-1}^{NOM}} \qquad \text{eq. A-65.}$$

We can do the same with the measurement. If the *h* function is differentiable, then the measurement can be represented by a Taylor series as the state. We then obtain:

$$\delta z_k = H_k \delta x_{k-1} + v_{k-1} \qquad H_k = \left.\frac{\partial h}{\partial x}\right|_{x = x_{k-1}^{NOM}} = \left.\begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial h_n}{\partial x_1} & \frac{\partial h_n}{\partial x_2} & \cdots & \frac{\partial h_n}{\partial x_n} \end{bmatrix}\right|_{x = x_{k-1}^{NOM}} \qquad \text{eq. A-66.}$$

The problem with linearization around a nominal trajectory, is that the deviation of the actual trajectory from the nominal tends to increase with time. As this deviation increases, the significance of the higher order terms in Taylor expansion of the trajectory also increases. A simple remedy for the deviation problem is to replace the nominal trajectory with the estimated trajectory. If the problem is sufficiently observable, then the deviation from the actual trajectory of the estimated one will remain sufficiently small and the linearization assumption will remain valid.

One of the drawbacks of this approach is that it increases the real-time computational burden. Whereas the linearization of F and H around a nominal trajectory may have been pre-computed off-line, linearization around the estimated trajectory must be computed in real time.

A.4.3.1 Linearized nonlinear systems in continuous time

The nonlinear system equations from Equation A-62 have been linearized, taking the first order derivatives with respect to the state *x* and control vector *u*. The linearization is done at the work point $x(t)$ and $u(t)$. The result is:

$$\delta\dot{x}(t) = \frac{\partial}{\partial x}\delta x(t) + \frac{\partial}{\partial u}\delta u(t) = F\delta x(t) + G\delta u(t)$$
$$\delta z(t) = \frac{\partial h}{\partial x}\delta x(t) = H\delta x(t) \qquad \text{eq. A-67.}$$

A.4.3.2 Linearized nonlinear systems in discrete time

We calculate the discrete transition matrix and the control matrix using Equation A-52 on page 159 and we have:

$$\Delta x(k+1) = \Phi\Delta x(k) + \Gamma\Delta u(k)$$
$$\Delta z(k) = H\Delta x(k) \qquad \text{eq. A-68.}$$

A.5 Camera Calibration

A.5.1 Perspective transformation

The relation between the camera coordinates and the ideal projection coordinates (in the image plane) is called perspective transformation. This perspective transformation is a kind of nonlinear mapping. We can choose the coordinate system (C,x,y,z) for the three-dimensional space and (c,u,v) for the retinal space. The relation between image coordinates and 3D space coordinates can be written as:

$$-\frac{f}{z} = \frac{u}{x} = \frac{v}{y} \qquad \text{eq. A-69.}$$

which can be rewritten in matrix form:

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \text{eq. A-70.}$$

where: u=U/S, v=V/S, if S≠0.

The geometric model of the pinhole camera consists of: a plane *R* called the *retinal plane* in which the image is formed through an operation called *perspective projection*; a point C, the *optical center*, located at a distance *f*, the focal length of the optical system, is used to form the image m in the retinal plane of the 3D point M as the intersection of the line ⟨C,M⟩ with the plane *R*.

A.6 Geometric Algorithms

A.6.1 Vertex Reduction

In vertex reduction, successive vertices that are clustered too closely are reduced to a single vertex. For example, if a polyline is being drawn in a computer display, successive vertices may be drawn at the same pixel if they are closer than some fixed application tolerance. In a large range geographic map display, two vertices of a boundary polyline may be separated by as much as a mile

(or more), and still be displayed at the same pixel; and the edge segments joining them are also drawn at this pixel. One would like to discard the redundant vertices so that successive vertices are separated several pixels, and edge segments are not just points.

Vertex reduction is the brute-force algorithm for polyline simplification. For this algorithm, a polyline vertex is discarded when its distance from a prior initial vertex is less than some minimum tolerance e > 0. Specifically, after fixing an initial vertex V0, successive vertices Vi are tested and rejected if they are less than e away from V0. But, when a vertex is found that is further away than e, then it is accepted as part of the new simplified polyline, and it also becomes the new initial vertex for further simplification of the polyline. Thus, the resulting edge segments between accepted vertices are larger than the e tolerance.

A.6.2 Douglas-Peucker Approximation

Instead of applying the rather sophisticated Teh-Chin algorithm to the chain code, we may try another way to get a smooth contour on a small number of vertices. The idea is to apply some very simple approximation techniques to the chain code with polylines, such as substituting ending points for horizontal, vertical, and diagonal segments, and then use the approximation algorithm on polylines. This preprocessing reduces the amount of data without any accuracy loss. The Teh-Chin algorithm also involves this step, but uses removed points for calculating curvatures of the remaining points.

The algorithm to consider is a pure geometrical algorithm by Douglas-Peucker [97] for approximating a polyline with another polyline with required accuracy:

1. Two points on the given polyline are selected, thus the polyline is approximated by the line connecting these two points. The algorithm iteratively adds new points to this initial approximation polyline until the required accuracy is achieved. If the polyline is not closed, two ending points are selected. Otherwise, some initial algorithm should be applied to find two initial points. The more extreme the points are, the better.
2. The algorithm iterates through all polyline vertices between the two initial vertices and finds the farthest point from the line connecting two initial vertices. If this maximum distance is less than the required error, then the approximation has been found and the next segment, if any, is taken for approximation. Otherwise, the new point is added to the approximation polyline and the approximated segment is split at this point. Then the two parts are approximated in the same way, since the algorithm is recursive. For a closed polygon there are two polygonal segments to process.

A.7 GPS NMEA Transmitted Sentences

The subsequent paragraphs define the sentences which can be transmitted on TXD1 by the GPS 25LP sensor boards.

A.7.1 Global Positioning System Fix Data (GGA)

$GPGGA,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,M,<10>,M,<11>,<12>*hh<CR><LF>

<1> UTC time of position fix, hhmmss format

<2> Latitude, ddmm.mmmm format (leading zeros will be transmitted)

<3> Latitude hemisphere, N or S

<4> Longitude, dddmm.mmmm format (leading zeros will be transmitted)

<5> Longitude hemisphere, E or W

<6> GPS quality indication, 0 = fix not available, 1 = Non-differential GPS fix available, 2 = Differential GPS (DGPS) fix available, 6 = Estimated

<7> Number of satellites in use, 00 to 12 (leading zeros will be transmitted)

<8> Horizontal dilution of precision, 0.5 to 99.9

<9> Antenna height above/below mean sea level, -9999.9 to 99999.9 meters

<10> Geoidal height, -999.9 to 9999.9 meters

<11> Differential GPS (RTCM SC-104) data age, number of seconds since last valid RTCM transmission (null if non-DGPS)

<12> Differential Reference Station ID, 0000 to 1023 (leading zeros will be transmitted, null if non-DGPS)

A.7.2 Recommended Minimum Specific GPS/TRANSIT Data (RMC)

$GPRMC,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,<10>,<11>,<12>*hh<CR><LF>

<1> UTC time of position fix, hhmmss format

<2> Status, A = Valid position, V = NAV receiver warning

<3> Latitude, ddmm.mmmm format (leading zeros will be transmitted)

<4> Latitude hemisphere, N or S

<5> Longitude, dddmm.mmmm format (leading zeros will be transmitted)

<6> Longitude hemisphere, E or W

<7> Speed over ground, 000.0 to 999.9 knots (leading zeros will be transmitted)

<8> Course over ground, 000.0 to 359.9 degrees, true (leading zeros will be transmitted)

<9> UTC date of position fix, ddmmyy format

<10> Magnetic variation, 000.0 to 180.0 degrees (leading zeros will be transmitted)

<11> Magnetic variation direction, E or W (westerly variation adds to true course)

<12> Mode indicator (only output if NMEA 2.30 active), A = Autonomous, D = Differential, E = Estimated, N = Data not valid

A.7.3 3D velocity Information (PGRMV)

The GARMIN Proprietary sentence $PGRMV reports three-dimensional velocity information.

$PGRMV,<1>,<2>,<3>*hh<CR><LF>

<1> True east velocity, -514.4 to 514.4 meters/second

<2> True north velocity, -514.4 to 514.4 meters/second

<3> Up velocity, -999.9 to 9999.9 meters/second

A.7.4 GPS DOP and Active Satellites (GSA)

$GPGSA,<1>,<2>,<3>,<3>,<3>,<3>,<3>,<3>,<3>,<3>,<3>,<3>,<3>,<3>,<3>,<4>,<5>,<6>*hh<CR><LF>

<1> Mode, M = manual, A = automatic

<2> Fix type, 1 = not available, 2 = 2D, 3 = 3D

<3> PRN number, 01 to 32, of satellite used in solution, up to 12 transmitted (leading zeros will be transmitted)

<4> Position dilution of precision, 0.5 to 99.9

<5> Horizontal dilution of precision, 0.5 to 99.9

<6> Vertical dilution of precision, 0.5 to 99.9

A.7.5 Differential GPS

Differential positioning with GPS, abbreviated DGPS, is a technique for improving GPS performance where two or more receivers are used. One receiver, usually at rest, is located at the reference site A with known coordinates and the remote receiver B is usually roving. The reference or base station calculates pseudorange corrections (PRC) and range rate corrections (RRC), which are transmitted to the remote receiver in near real time. The remote receiver applies the corrections to the measured pseudoranges and performs point positioning with the corrected pseudoranges. The use of the corrected pseudoranges improves positional accuracy[98].

A.7.6 Coordinate transformations

Transformation from RD coordinate system (*x*,*y*) to geographic coordinate system(φ,λ):

x_0 = 155000.000;
y_0 = 463000.000;
φ_0 = 52.156160556;
λ_0 = 5.387638889;
$\delta x=(x-x_0)*pow(10,-5)$;
$\delta y=(y-y_0)*pow(10,-5)$;

$\delta\varphi = a_{01}*\delta y + a_{20}*pow(\delta x,2) + a_{02}*pow(\delta y,2) + a_{21}*pow(\delta x,2)*dy + a_{03}*pow(\delta y,3)$;
$\delta\varphi += a_{40}*pow(\delta x,4) + a_{22}*pow(\delta x,2)*pow(\delta y,2) + a_{04}*pow(\delta y,4) + a_{41}*pow(\delta x,4)*\delta y$;
$\delta\varphi += a_{23}*pow(\delta x,2)*pow(dy,3) + a_{42}*pow(\delta x,4)*pow(\delta y,2) + a_{24}*pow(\delta x,2)*pow(\delta y,4)$;
$\varphi = \varphi_0 + \delta\varphi/3600$;

$\delta\lambda = b_{10}*\delta x + b_{11}*\delta x*\delta y + b_{30}*pow(\delta x,3) + b_{12}*\delta x*pow(dy,2) + b_{31}*pow(\delta x,3)*\delta y$;
$\delta\lambda += b_{13}*\delta x*pow(\delta y,3) + b_{50}*pow(\delta x,5) + b_{32}*pow(\delta x,3)*pow(\delta y,2) + b_{14}*\delta x*pow(\delta y,4)$;
$\delta\lambda += b_{51}*pow(\delta x,5)*\delta y + b_{33}*pow(\delta x,3)*pow(\delta y,3) + b_{15}*\delta x*pow(\delta y,5)$;
$\lambda = \lambda_0 + \delta\lambda/3600$;

In this formula δx, δy, $\delta\varphi$ and $\delta\lambda$ are the difference in coordinate with respect to the central point in Amersfoort. For that we need to apply the following translations::

$\delta x = x - 155000.000m$
$\delta y = y - 463000.000m$
$\delta\varphi = \varphi - 52^o09'22.178''$
$\delta\lambda = \lambda - 5^o23'15.500''$

The coefficients for the transformation are:

a_{01}	=	3236.0331637	b_{10}	=	5261.3028966
a_{20}	=	-32.5915821	b_{11}	=	105.9780241
a_{02}	=	-0.2472814	b_{12}	=	2.4576469
a_{21}	=	-0.8501341	b_{30}	=	-0.8192156
a_{03}	=	-0.0655238	b_{31}	=	-0.0560092
a_{22}	=	-0.0171137	b_{13}	=	0.0560089
a_{40}	=	0.0052771	b_{32}	=	-0.0025614
a_{23}	=	-0.0003859	b_{14}	=	0.0012770
a_{41}	=	0.0003314	b_{50}	=	0.0002574
a_{04}	=	0.0000371	b_{33}	=	-0.0000973
a_{42}	=	0.0000143	b_{51}	=	0.0000293
a_{24}	=	-0.0000090	b_{15}	=	0.0000291

Transformation from geographic coordinate system(φ,λ) to RD coordinate system (x,y) is given by:

$\delta\varphi=(\varphi - \varphi_0) * 0.36;$
$\delta\lambda=(\lambda - \lambda_0) * 0.36;$

$\delta x = c_{01}*\delta\lambda + c_{11}*\delta\varphi*\delta\lambda + c_{21}*pow(\delta\varphi,2)*\delta\lambda + c_{03}*pow(\delta\lambda,3);$
$\delta x+= c_{31}*pow(\delta\varphi,3)*\delta\lambda + c_{13}*\delta\varphi*pow(\delta\lambda,3) + c_{23}*pow(\delta\varphi,2)*pow(\delta\lambda,3);$
$\delta x+= c_{41}*pow(\delta\varphi,4)*\delta\lambda + c_{05}*pow(\delta\lambda,5);$
$x=x_0 + \delta x;$
$x=round(100*x)/100;$

$\delta y = d_{10}*\delta\varphi + d_{20}*pow(\delta\varphi,2) + d_{02}*pow(\delta\lambda,2) + d_{12}*\delta\varphi*pow(\delta\lambda,2);$
$\delta y+= d_{30}*pow(\delta\varphi,3) + d_{22}*pow(\delta\varphi,2)*pow(\delta\lambda,2) + d_{40}*pow(\delta\varphi,4);$
$\delta y+= d_{04}*pow(\delta\lambda,4) + d_{32}*pow(\delta\varphi,3)*pow(\delta\lambda,2) + d_{14}*\delta\varphi*pow(\delta\lambda,4);$
$y=y_0 + \delta y;$
$y=round(100*y)/100;$

The coefficients for the transformation are:

c_{01}	=	190066.98903	d_{10}	=	309020.31810
c_{11}	=	-11830.85831	d_{02}	=	3638.36193
c_{21}	=	-114.19754	d_{12}	=	-157.95222
c_{03}	=	-32.38360	d_{20}	=	72.97141
c_{31}	=	-2.34078	d_{30}	=	59.79734
c_{13}	=	-0.60639	d_{22}	=	-6.43481
c_{23}	=	0.15774	d_{04}	=	0.09351
c_{41}	=	-0.04158	d_{32}	=	-0.07379
c_{05}	=	-0.00661	d_{14}	=	-0.05419
			d_{40}	=	-0.03444

To the best of our knowledge this conversion procedure is accurate to about 50 cm, partly because of the truncation of series used in the conversion. A rough indication of the numerical accuracy of software conversion can be obtained by converting back and forth between the same coordinates.

Acknowledgements

It is my pleasure to thank my advisor, Pieter Jonker, for his entire support and advice that he gave me while working towards my PhD, and for the comprehensive reviews of this thesis. I appreciate the several discussions and the always open door whenever I had a question. I enjoyed working with him, and I hope he enjoyed working with me too.

It is a pleasure to thank prof. Ted Young, one of my promotors, for his enthusiasm and promptitude in reading and structuring the ideas in my thesis. I also thank prof. Inald Lagendijk for following my steps during my PhD studies, and being always ready to provide me with help and advice. I would further like to thank all my promotors and committee members for their valuable comments that helped me improving my work and presentation style.

I would like to thank to all the people with whom I worked during my PhD studies. In UBICOM team I found a friendly environment in which I could easily develop myself. I would also like to thank to all PH people, staff and aio's for their support and for the nice time spent among them.

There is a person with whom I worked in the past, who had decisively influenced my development, and to whom I would like to express all my gratitude. He is prof. Aurel Vlaicu, with whom I worked at Technical University Cluj-Napoca, during and after my graduation. He introduced me in the world of image processing, and made me like it.

Many thanks to my friends Sorin, Maria, Stefan, Andreea, Ildi, Gerold, Bogdan, Dani, Rares, Cristina and Andrei, because they were always besides me, and made living in Delft to be as enjoyable as possible.

Well, nothing would have been possible without my family who provide me with all the support. First of all I thank you to my lovely wife, Monica, for her understanding and to my daughter Irina who brought happiness in our life. I thank my parents and my brothers for their continuous love, support and understanding.