Hybrid Marker-less Camera Pose Tracking with Integrated Sensor Fusion

Mrs. Armaghan Moemeni
BEng. (Hons.), MSc.

A thesis is submitted to De Montfort University
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

September, 2014

Abstract

This thesis presents a framework for a hybrid model-free marker-less inertial-visual camera pose tracking with an integrated sensor fusion mechanism. The proposed solution addresses the fundamental problem of pose recovery in computer vision and robotics and provides an improved solution for wide-area pose tracking that can be used on mobile platforms and in real-time applications.

In order to arrive at a suitable pose tracking algorithm, an in-depth investigation was conducted into current methods and sensors used for pose tracking. Preliminary experiments were then carried out on hybrid GPS-Visual as well as wireless micro-location tracking in order to evaluate their suitability for camera tracking in wide-area or GPS-denied environments. As a result of this investigation a combination of an inertial measurement unit and a camera was chosen as the primary sensory inputs for a hybrid camera tracking system.

After following a thorough modelling and mathematical formulation process, a novel and improved hybrid tracking framework was designed, developed and evaluated. The resulting system incorporates an inertial system, a vision-based system and a recursive particle filtering-based stochastic data fusion and state estimation algorithm. The core of the algorithm is a state-space model for motion kinematics which, combined with the principles of multi-view camera geometry and the properties of optical flow and focus of expansion, form the main components of the proposed framework.

The proposed solution incorporates a monitoring system, which decides on the best method of tracking at any given time based on the reliability of the fresh vision data provided by the vision-based system, and automatically switches between visual and inertial tracking as and when necessary. The system also includes a novel and effective self-adjusting mechanism, which detects when

the newly captured sensory data can be reliably used to correct the past pose estimates. The corrected state is then propagated through to the current time in order to prevent sudden pose estimation errors manifesting as a permanent drift in the tracking output.

Following the design stage, the complete system was fully developed and then evaluated using both synthetic and real data. The outcome shows an improved performance compared to existing techniques, such as PTAM and SLAM. The low computational cost of the algorithm enables its application on mobile devices, while the integrated self-monitoring, self-adjusting mechanisms allow for its potential use in wide-area tracking applications.

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Eric Tatham. His wise advice and suggestions have greatly contributed to the success of this research.

Very special thanks to my colleague Dr Cristian Serdean for his continuing support and encouragement, and for proof reading this thesis and offering suggestions for improvement.

I would like to thank my parents Zahra and Shahab for always supporting and believing in me, and my sister Azadeh for her support and distractions when needed, and for always being there for me.

Most of all, I am very grateful to my dearest friend and soulmate, my husband Arash Ghadar, whose unconditional love and devotion, encouragement, and support enabled me to complete this thesis.

Last but not least, I would like to thank my three year old son, Artin, for giving me a reason to go through it all, and finish on time.

CONTENTS

CHAPTER ONE

1 Hybrid Marker-less Camera Pose Tracking with Integrated Sensor Fusion

Pose tracking is an enabling technology with potential applications in numerous industries, including entertainment and immersive games, augmented reality, industrial maintenance and engineering, architecture, medicine, assisted living for the elderly, security, education, prototyping and autonomous navigation systems. The aim of pose tracking is to find the three dimensional (3D) position and orientation of a moving object, such as a camera, using information collected from one or multiple sensors.

Over the past few decades there have been numerous proposals and solutions for pose recovery. In many cases, in computer vision systems, a camera has been the only sensor available for tracking and there have been several advances in pose tracking based on computer vision techniques, which recover the 2D/3D correspondences of 3D features in the environment in successive images. Notable examples of a computer vision approach to pose tracking include the work produced by (Irschara, 2012), (Klein, 2006) and (Chiuso, et al., 2002).

In a vision-based context, there are two main methods for tracking, namely marker-based and marker-less. The former is based on tracking fixed fiducial markers, which implicitly solves the tracking and localization problem since the

markers and their relative 3D positions are known. Zhang et al (Zhang, et al., 2002) have carried out a comprehensive study on approaches to marker-based tracking methods using fiducial markers. Examples include ARToolKit (ART, 1999) and ARTag (Fiala, 2010) both of which employ planar fiducial markers. These examples have been specifically designed for camera tracking as a solution to the problem of image registration for augmented reality.

The marker-less approach, however, uses naturally distinctive features such as points, lines, edges, or textures, whose 3D positions are not known. These systems use naturally occurring features in the world for both motion estimation and localization. Comprehensive surveys on monocular camera pose tracking using only vision-based approaches have been carried out by (Desouza & Kak, 2002), (Trucco & Plakas, 2006) and (Mautz & Tilch, 2011).

Human beings and animals consciously and subconsciously fuse various sources of information in order to navigate and interact with their environment. This is continuously occurring by processing the output signals from built-in biological sensors. One of the most utilised examples of senor fusion in our daily activities is the combination of human vision system (the eye-brain combination) and vestibular sensors (located inside the ear). This is also referred to as kinaesthetic-vestibular sensor fusion. The vestibular system encodes self-motion information by detecting the motion of our head in space; i.e. the three dimensional world. This provides the body with subjective senses of motion and orientation, which play an important role in the stabilization of gaze, and the control of balance and posture (Cullen, 2012) and (Angelaki & Cullen, 2008) .

Beyond some of the limitations of the human sensory experience, Hughes (1999) has also reviewed multi-sensory systems, which add extra sensing modality to some animals' navigation abilities and aids in interaction with their surroundings. Examples include echo-location in bats (acoustic ranging sensors), navigation using magnetic fields in some birds (magnetic dead-

reckoning) and the electric sensors used by some sharks and eels for prey detection.

In like manner, one of the proposed solutions to pose recovery and wide-area localisation and tracking has been the application of multi-sensory approaches. Combining the data from various sensors and devices such as cameras, acoustic sensors, inertial measurement units (IMUs), GPS, and wireless sensors has been extensively researched and commercialised to some degree in the past few decades. Notable examples of multi-sensory approaches for recovery of pose can be found in (Bleser, 2009), (Schleicher, et al., 2009), (Droeschel, et al., 2011), (Macii, et al., 2011), and (Scaramuzza, et al., 2014).

Inspired by the natural kinaesthetic-vestibular sensor fusion in humans, and animals in navigation and interaction with their surrounding environment, and considering the state-of-the-art of the computer vision algorithms as well as the recent enhancements in Micro-Electro-Mechanical Systems (MEMS)-based inertial sensors in measuring ego-motion, the work in this thesis has focused on an investigation into current computer-based tracking methods and on developing an improved inertial-visual sensor fusion technique for tracking the position and orientation (pose) of a moving camera.

Following thorough investigation into existing tracking systems, a new and improved hybrid algorithm for single camera pose tracking has been designed, implemented and evaluated. The core of the algorithm is a state-space model for motion kinematics which, combined with the principles of multi-view camera geometry and the properties of optical flow and focus of expansion, form the main components of the framework. The proposed solution also incorporates a monitoring system, which determines the best method of tracking at any given time based on the reliability of fresh vision data provided by the vision-based system, and automatically switches between visual and inertial tracking as and when necessary. The system also includes an effective self-adjusting mechanism, which detects when the newly captured sensory data can be

reliably used to correct the past pose estimates. The corrected state is then propagated through to the current time in order to prevent sudden pose estimation errors manifesting as a permanent drift in the tracking output. The system design and methodology have been the basis for a number of publications e.g. (Moemeni & Tatham, 2010)and (Moemeni, et al., 2014).

The proposed system provides a more comprehensive solution with reduced computational cost as well as an improved real-time ubiquitous performance compared with existing hybrid and vision-only systems. This has been addressed and analysed in more detail throughout this thesis.

1.1 Applications Requiring Pose Tracking

Camera tracking as an enabling technology can be exploited for various applications, in particular, navigation and localisation systems as well as tracking within augmented and mixed reality environments. These applications are described in some detail in the following sections.

1.1.1 Augmented Reality

An Augmented Reality (AR) system “supplements the real world with virtual (computer-generated) objects that appear to coexist in the same space as the real world” (Azuma, 2001).

The term augmented reality was initially introduced by Tom Caudell in 1990 in relation to Boeing’s Computer Services' Adaptive Neural Systems Research and Development project in Seattle (Caudell & Mizell, 1992). The purpose was to aid manufacture of complex wiring looms used in aircraft electronics by visually superimposing virtual graphics onto the real environment, as an alternative to the expensive diagrams and marking devices (Figure 1.1)

Although the term was initially born in the early 1990's, researchers had for years been creating technologies that could relate to their environment and give access to information.

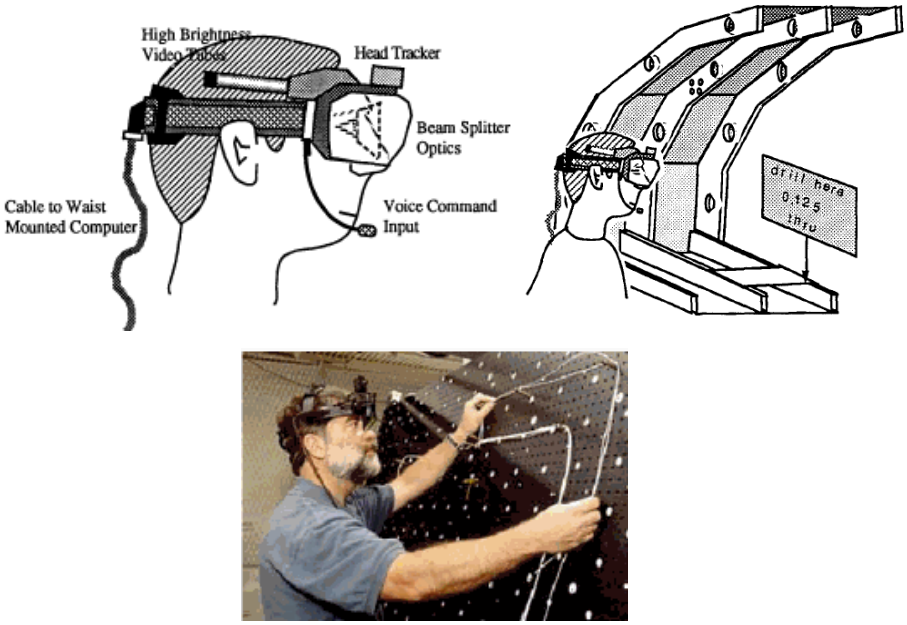


Figure 1.1 : Boeing's AR System – Source: (Caudell & Mizell, 1992).
An application of Heads-up display technology (HUDset) used to dynamically mark the position of drill/rivet hole inside an aircraft fuselage - ©Boeing Inc., USA

In 1968, Ivan Sutherland, invented the first head-mounted display (HMD) (Figure 1.2) in order to visually superimpose computer generated 3D models onto the user's view of the real environment (Sutherland, 1968).

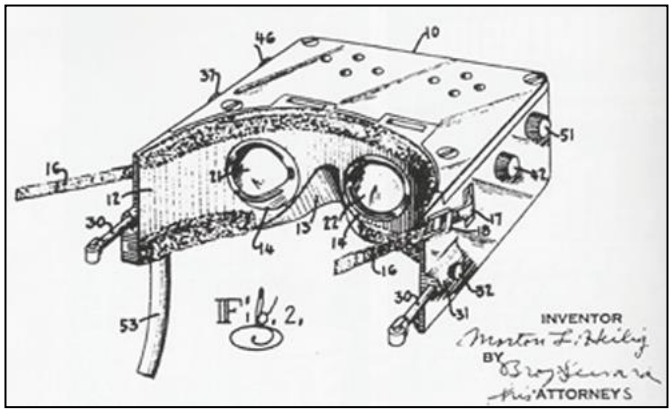


Figure 1.2 : The first Head-Mounted Display (HMD) by Sutherland

Furthermore, Morton Helig's Sensorama (U.S. Patent #3050870) (Figure 1.3) was built with the purpose of giving a more immersive 3D cinematic experience. Sensorama was one of the earliest multi-sensory (multimodal) technologies in the form of a 1980's arcade game. The game gave the players the experience of riding a motorcycle on the streets of Brooklyn, where they could feel the wind on the face, the vibration of the seat and the smell of the city. (Earnshaw, et al., 1993).

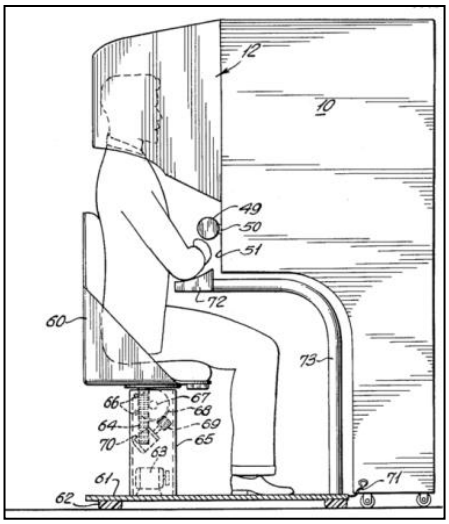


Figure 1.3 : The Sensorama, - U.S. Patent #3050870

In 1994, Azuma developed the first motion-stabilized AR display that worked outdoors and achieved tighter registration comparing to previous outdoor AR system (Azuma, 1997) (Azuma, et al., 1999). This hybrid tracking system for outdoor AR used an Omnistar 7000 differential GPS receiver, a Precision Navigation TCM2 compass and tilt sensor, and three Systron Donner GryroChop gyroscopes. It was operated in both head-worn ad handheld modes. Figure 1.4 illustrates the HMD and the sensors configurations.

The goal of this sensor fusion was to estimate the angular position and rotation rate of the HMD from the inputs from the TCM2 and the gyroscopes. However, the position was then extrapolated one frame to the future to estimate the head orientation at the time the image is displayed on the see-through display. Although the registration was not reported with very high accuracy considering

the inherited drifting errors introduced by inertial sensors, this could be regarded as one of the pioneered hybrid systems in the application of outdoor augmented reality (Azuma, et al., 1999).

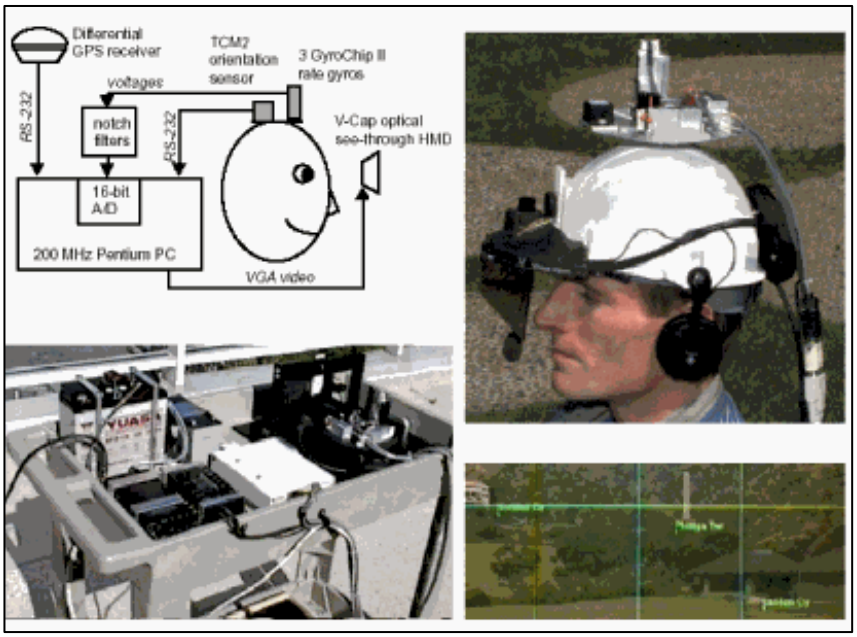


Figure 1.4 : Motion-stabilized AR Display by Azuma (Azuma, et al., 1999)
© Ronald Azuma, HRL Laboratories

As augmented reality continues to evolve and take different forms, the demand for its use and the scope of its applications are growing accordingly. The state-of-the-art of recent technologies, systems and applications of AR are surveyed in (Van Krevelen & Poelman, 2010) and (Carmigniani, et al., 2011).

In recent years, on-demand access to information and greater integration of real and virtual worlds have revolutionised the conventional perception of human computer interaction. The notion of ‘ubiquitous computing’, also known as ‘ambient intelligence’, aims to achieve an anytime anywhere model in modern computing systems.

Mobile augmented reality, as a paradigm of ubiquitous computing, is an alternative to head-mounted displays, and has become more widely employed in recent years. The ubiquity of mobile platforms in the form of phones and tablet computers has given rise to a wide-range of new applications, for which it is essential to know the position and orientation of the mobile device so that overlaying information or images can be superimposed in the correct size, position and at an appropriate viewing angle.

Mobile AR has shifted the focus towards smaller handheld devices with the potential for wider-area augmentation. Such systems present 3D/video information superimposed on the roaming user's view. However, the wide-area and ubiquitous nature of Mobile AR, requires reliable pose tracking for the purpose of image registration. In addition, device size and price constraints, as well as real-time interactive requirements, drive the need for efficient tracking algorithms. Historically, the Touring Machine (Feiner, et al., 1997), which used backpacks with laptop computers and HMDs, as depicted in Figure 1.5 , was perhaps the earliest work in the development of Mobile AR. The trend progressed further with the invention of Ultra-mobile personal computers (UMPCs) and mobile AR systems such as those described in (Wagner, et al., 2005), (Kruijff & Veas, 2007) and (Reitmayr & Drummond, 2006). Figure 1.6 illustrates the vision-based tracking approach for mobile AR developed by (Reitmayr & Drummond, 2006).

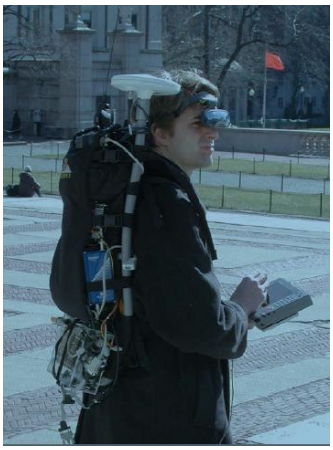


Figure 1.5 : Prototyping 3D Mobile Augmented Reality systems (Feiner, et al., 1997)

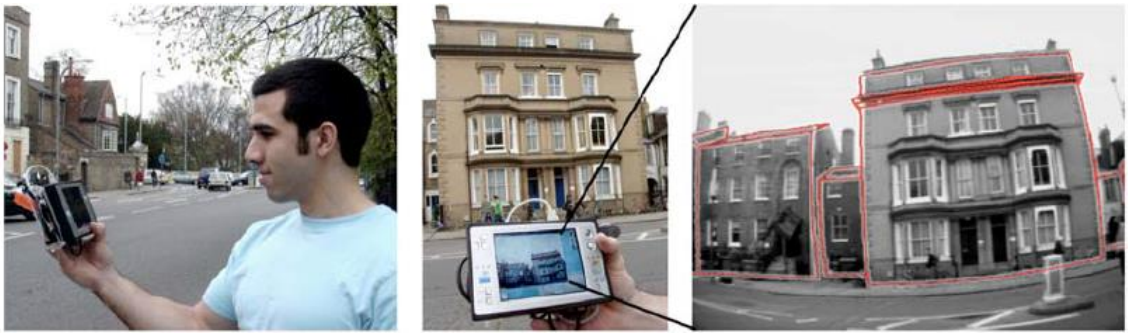


Figure 1.6 : Mobile Augmented Reality
(Left) A user operating a handheld AR unit tracked in an urban environment.
(Middle) Live shot showing the unit tracking a building
(Right) Screenshot from a pose close to the left images with overlaid building outline." Sources: (Reitmayr & Drummond, 2006) and (Schall, 2011)

Today, the advent of smart phones with integrated GPS, camera, inertial sensors and very high performance processing units, has made possible a new class of mobile AR applications. Notable examples are the Wikitude (Wikitude, 2014), Argon (Argon, 2014) and Layer (Layar, 2014) mobile AR applications.

For instance, Wikitude (Figure 1.7) was originally developed as a mobile AR travel guide based on user-generated Web2.0 Wikipedia or Panoramio content. The users could see the annotated landscape, mountain names or landmark descriptions in an AR camera view (Schall, 2011). In the recent developments at Wikitude lab, a solution called 'AR Window' was launched, which enabled any mobile webpage to include AR. This meant that the mobile web pages are now able to open up the camera view of its smartphone to view the live video streams with additional content on top of it. See Figure 1.7 as an example.

Figure 1.7 : Wikitude Mobile AR - Source (Schall, 2011).

1.2 Navigation and Localisation

The applications of pose tracking in navigation and localisation range from search and rescue helicopters to civilian drones and robotics as well as tracking and localisation in constrained environments. Although GPS is regarded as the most ubiquitous means of wide-area location tracking in these scenarios, it still suffers from ranging errors due limited accuracy and to obstruction of line of sight. Despite recent advances in GPS technologies such as the invention of differential GPS systems with less accuracy deficiency, wide-area tracking and localisation still remain ongoing problems in GPS-denied environments such as urban canyons, some combat zones like city warfare, and indoor environments like hospitals and shopping malls.

Technological advances in MEMS (Micro-Electro-Mechanical Systems)-based inertial sensors have enabled pose estimation in systems such as mobile robots or unmanned micro aerial vehicles (MAVs), often operating in urban canyon environments, where GPS signals are either unavailable or unreliable. Recently, there has been substantive research and progress in autonomous MAVs, such as the EU-funded (FP7:2007-2013) SFLY (Swarm of Micro Flying Robots) project (Zürich, 2014), which consists of a micro flying robot using only one single on-board camera and an inertial measurement unit (IMU). This vision-controlled MAV has proved capable of autonomous navigation in the GPS-denied environments.

The objective of the SFLY project was to develop several small and safe helicopters, as in Figure 1.8 and Figure 1.9, which could fly autonomously in city-like environments and hence be used to assist humans in tasks such as rescue and monitoring (Scaramuzza, et al., 2014). Improved positioning of these vehicles in GPS-denied areas has been achieved through computer-vision pose tracking approaches such as Visual SLAM, and variety of other sensor fusion systems.

Figure 1.8 : “The three SFLY hexacopters designed for inertial-visual navigation in GPS-denied environments.” Source : (Scaramuzza, et al., 2014) – *Sfly* Project : www.sfly.org

Figure 1.9 : Autonomous micro helicopter used in for search and rescue in a disastrous area Source : *Sfly* Project www.sfly.org

A further recent application of pose tracking for navigation and localisation has been the proposed Amazon civilian drones, called Amazon© Prime Air (BBC_News, 2013), meant to be used for package delivery. Amazon’s chief executive Jeff Bezos stated that the drones, called Octocopters, could deliver packages weighing up to 2.3kg to customers within 30 minutes of them placing the order. Although the drones have GPS tracking systems, when entering GPS denied areas they would require an alternative tracking system to find the correct address and then return to the point where the GPS signal can be recovered. This is another example which proves the need for developing other

multimodal tracking and localisation system which works in GPS denied environments.

Figure 1.10 : Amazon Prime Air : Source : (Amazon_PrimeAir, 2014) www.amazon.com

1.3 Problem Formulation, Objectives and Contributions

The aim of this study is to develop an improved solution for marker-less pose tracking in order to estimate the 6 Degrees of Freedom (6DOF) pose of a moving camera, consisting of 3D position and 3D orientation, with reference to a fixed coordinate system.

This led to the following specific objectives:

- To review the literature on existing systems used for tracking camera position and orientation
- To review the literature on hybrid pose tracking algorithms and techniques using computer vision
- To evaluate existing techniques in relation to their advantages and limitations
- To test the viability of using an inertial-visual hybrid approach for refining the accuracy of GPS location tracking and to determine the current reliability of using wireless beacons for location tracking.
- To establish an understanding of the mathematical models and tools underpinning an improved approach to inertial-visual hybrid tracking

- To design the system architecture and algorithm for an improved approach to inertial-visual hybrid tracking
- To validate the system efficiency using both synthetic and real data
- To review the outcomes of the work and to suggest directions for future research

In order to achieve these objectives, following a thorough investigation into existing single and multi-sensory pose tracking systems and algorithms (Chapter 2 and 3), a series of preliminary experiments were carried out and the shortcomings of existing tracking systems were identified. Following this study, a hybrid 'GPS-Visual SLAM (Simultaneous Localisation And Mapping)' tracking system was developed and its potential to refine GPS location accuracy was evaluated with a view to a potential comprehensive wide-area tracking solution in the future (Chapter 4, Section 4.1). In addition, the possibility of using wireless location tracking (such as Apple©'s iBeacons) were investigated as an alternative and/or as a supplement to GPS for achieving more refined location tracking in GPS-denied environments (Chapter 4, Section 4.2).

These investigations and experiments led to the design, development and evaluation of a novel hybrid tracking system (outlined in Chapters 6 and 7), featuring the following main contributions:

- **A proposal for a novel and improved hybrid Inertial-Visual pose tracking system:** The system benefits from the agility of inertial tracking and robustness of vision-based tracking, while addressing the shortcomings of each individual system. The proposed stochastic data fusion method deals with the uncertainty, noise and error of sensory inputs and provides a robust solution, which can potentially be used by applications requiring wide-area pose tracking.

- **Model-free, marker-less pose tracking:** The proposed system does not require a model of the environment, nor does it require markers to be

placed in the environment, thus making it suitable for use in unknown environments. In addition, the system does not need to retain a map of feature points, hence requiring reduced processing effort, enabling its use on mobile platforms with limited resources.

- **Incorporating a decision-making mechanism for selecting the suitable method of tracking:** The system incorporates a means of measuring the level of suitability of image data for tracking. Such information is used to automatically determine the most suitable method of tracking at any particular time. This feature minimises the probability of sudden errors, which could have permanent adverse effect on tracking performance, making the proposed tracking method suitable for the potential use in a wide-area outdoor environment.

- **Development of a self-adjustment mechanism to improve the pose estimate:** The system monitors the performance of tracking and detects when, based on new incoming sensory information, the past state estimate has been inaccurate. A mechanism has been designed to propagate the updated past state through to the current time, minimising the probability of pose error manifesting as a permanent drift in the pose tracking. This self-adjustment is beneficial in tracking applications requiring travelling over long distances.

- **Development of a test system for performance evaluation:** In order to evaluate the performance of the proposed solution, various indicative sample sensory data have been generated to simulate real world data, enabling a thorough evaluation of various aspects and parameters of the proposed camera tracking algorithm. In addition the system performance was evaluated using a dataset containing real data generated by the IMU and camera on board a micro aerial vehicle (MAV), together with a set of ground truth data produced by the ©Vicon motion capture system (Vicon,

1984). The system evaluation methodology, the datasets and results can be used for benchmarking and future work in this area.

1.4 Thesis Outline

The remainder of this thesis is organised as follows:

Chapter 2 provides a review of pose tracking systems from single to multi-sensory methods used in estimation of 6DOF position and orientation.

Chapter 3 reviews pose tracking techniques using computer vision-based methods, primarily exploiting image processing algorithms for detecting the natural features in an image in order to tackle the problems of motion tracking and matching. Recursive filtering techniques in inertial-visual sensor fusion are also reviewed in this chapter leading the proposal of a new system specification for the hybrid inertial-visual tracking system in Chapter 6.

Chapter 4 presents tracking techniques using radio frequency (RF)-based positioning systems. In order to evaluate the performance of such systems, a combined GPS and vision-based pose tracking system (GPS-Visual SLAM sensor fusion) and then a system based on wireless iBeacon technology were implemented. This chapter describes these two systems followed by performance analysis and conclusions based on carrying out a number of experiments.

Chapter 5 provides geometric models and mathematical tools for camera modelling and representation of 3D moving object. It introduces the notation and provides the background for understanding the system modelling and algorithms designed and presented in Chapter 6 and 7. It first introduces the operating principles, mathematical models, geometry of two views (i.e.

epipolar geometry), measurements of cameras and inertial sensors and then proceeds with the fundamental estimation techniques considering computer vision and recursive filtering methodologies.

Chapter 6 demonstrates the main inertial-visual hybrid camera tracking system architecture and algorithm proposed in this thesis. The hybrid tracking algorithm is fully explained and illustrated in this chapter.

Chapter 7 sets out the framework for system validation and provides the tracking outcome and analysis using both synthetic and real data. The simulated data has been synthesised (considering the precise kinematics motion equations and IMU datasheet characteristics) and evaluated. A proof-of-concept has been demonstrated in a simulated dataset setup through various procedural tests and evaluations. Moreover, the developed algorithm has been tested and evaluated with a set of real dataset (i.e. SFLY dataset).

Finally Chapter 8 concludes the thesis and offers suggestions for future work.

Appendix A includes the author's related published work.

CHAPTER TWO

2 A Review of Pose Tracking Systems

This chapter provides an overview of the pose tracking systems currently used in the estimation of 6DOF position and orientation, from single to multi-sensory approaches. Sensory approaches are reviewed and analysed, such as a single camera with passive or active visual markers; inertial measurement units (IMUs); Global Positioning System (GPS); systems utilising time-of-flight, including optical, radar and acoustic sensing; wireless systems; as well as hybrid techniques, which make use of the above in various combinations.

2.1 Camera as the Main Motion Sensor

Traditionally, in computer vision systems, a camera has been used as the only motion sensor for tracking. Over the past decades, there have been several advances in computer vision-based tracking techniques and algorithms that can recover 2D/3D correspondences between successive images. These can be divided into two main approaches; marker-based and marker-less. The former is characterised by tracking visually distinctive markers, which implicitly solves the tracking and localisation because the markers and their relative 3D positions are known. In the latter case, no such pre-prepared markers are employed and reliance is placed upon tracking naturally appearing features.

Zhang et al (2002) have carried out a comprehensive survey on the approaches to marker-based tracking methods with notable examples including; ARToolKit (ArToolKit, 1999) and ARTag (Fiala, 2010). In both of these cases, planar

fiducial markers are used for camera tracking to help solve the fundamental problem of image registration in Augmented Reality (AR). Marker-less approaches use naturally distinctive features such as points, lines, edges, or textures available in the scene, whose 3D positions are not known. These systems use these naturally occurring features for both motion estimation and localisation. Comprehensive surveys on monocular camera-pose tracking, using only vision-based approaches, have been carried out in (Desouza & Kak, 2002), (Trucco & Plakas, 2006) and (Mautz & Tilch, 2011). In this thesis, vision-based algorithms used for pose estimation are reviewed in Chapter 3.

2.1.1 Optical Sensing

Optical sensing relies on detecting reflected or emitted light. Therefore such systems have two main components; light sources and optical sensors. Optical tracking systems have the advantage of having high accuracy and are suitable for use in large visually uniform spaces, where feature detection using computer vision algorithms may fail or produce high rates of latency.

The optical systems, for which the light source is on the moving target and the sensors (markers) in the environment, are generally referred to as outside-looking-in or simply Outside-In. On the other hand, if the optical marker is attached to the moving target, the tracking is referred to as inside-looking-out or Inside-Out (Welch & Foxlin, 2002).

Marker-based systems are mainly used when it is convenient to attach markers to the tracking scene or where natural features are not easily available to be used by the vision-based systems alone. Despite recent advances in marker-less tracking approaches, using naturally distinctive features such as points, lines, edges, or textures whose 3D positions are not known, there are still applications where adding fiducial markers can be an advantage. Examples include; featureless indoor scenes, indoor augmented reality in known

environments, cases where message tags are used to trigger a behaviour and also generic pose estimation in industrial settings with fixed features.

2.1.1.1 Passive and Active Light Sensing using Markers

The light markers can be either passive objects that reflect the ambient light or active objects that emit internally generated light. In the case of passive markers, they are normally in the form of spherical shapes covered with retro-reflective material, which reflects infrared radiation from the incoming light. The systems work based on the triangulation of light for calculation of the 6DOF of the moving object. Examples of passive sensors include distinguishable man-made markers (e.g. retro-reflective markers used in ©Vicon motion capture systems) or else natural features in the environment. However, passive marker tracking systems often require carefully controlled lighting environments and are known to be prone to produce errors in the case of partial target occlusion. Also, when tracking multiple objects their general success rate is reduced (Steidle, et al., 2012).

In contrast, active sensing systems are based mainly on active electronic components such as illuminating light emitting diodes (LEDs) or lasers. They use the same principle of triangulation but with each active marker uniquely identifiable. Tracking efficiency tends to reduce with the increase in the number of markers and for large numbers of markers there is a need for a model-based tracking system (Steidle, et al., 2012).

A very early example of an optical tracking system was the Twinkle Box (Burton, 1973), (Burton & Sutherland, 1974). This system measured the position of user-worn flashing lights with optical sensors mounted in the environment behind rotating slotted disks. Also, the Selspot system (Woltring, 1974) used fixed photodiode sensors and target-mounted infrared LEDs that could be tracked within a one cubic meter volume. Later, systems such as; Flash-Point and Pixsys by Image Guided Technologies, Inc. (IGT) (acquired by ©Stryker in 2000) (StryKer, 1894), the laserBIRD system by ©Ascension Technology

(Ascension, 1986), the ©CODA Motion Capture Systems (CodaMotion, 1970), UNC's HiBall (Welch, et al., 2001), ©Vicon Motion Capture System (Vicon, 1984) , and ©ART (ART, 1999) used mainly optical sensor systems (passive and/or active) at a relatively high sample rate and for tracking a larger number of objects.

Welch, G. et al. (Welch, et al., 2001) surveyed the developments that majority had taken place at the University of North Carolina at Chapel Hill, where they pioneered active optical sensors/markers from the Self-Tracker (Bishop, 1984) to the HiBall Tracking System. The following figures illustrate some of these optical tracking systems.

Figure 2.1 shows the Electro-optical head-tracking system demonstrated in the Tomorrow's Realities gallery at ACM SIGGRAPH 1991. This system used four head-worn lateral-effect photodiodes looking upward at a regular array of infrared LEDs installed in precisely machined ceiling panels (Ward, et al., 1992) and (Welch, et al., 2001).

Figure 2.2 on the left shows the HiBall Tracking System. The HiBall is a cluster of 6 lenses and 6 photodiodes arranged so that each photodiodes can view LEDs through several lenses out of 6. This system generated more than 2,000 pose estimates per second, with less than 1ms of latency and up to 0.5 mm and 0.03° absolute error and noise, everywhere in a 4.5x3x8.5m room (with more than two meters of height variation). The weight of the user-worn HiBall was approximately 300 grams. The image on the right shows the physical sensors arrangement of the HiBall Tracking System.

Figure 2.1 : Active Markers - Electro-optical head-tracking system (Self-Tracker)
Source: (Welch, et al., 2001).

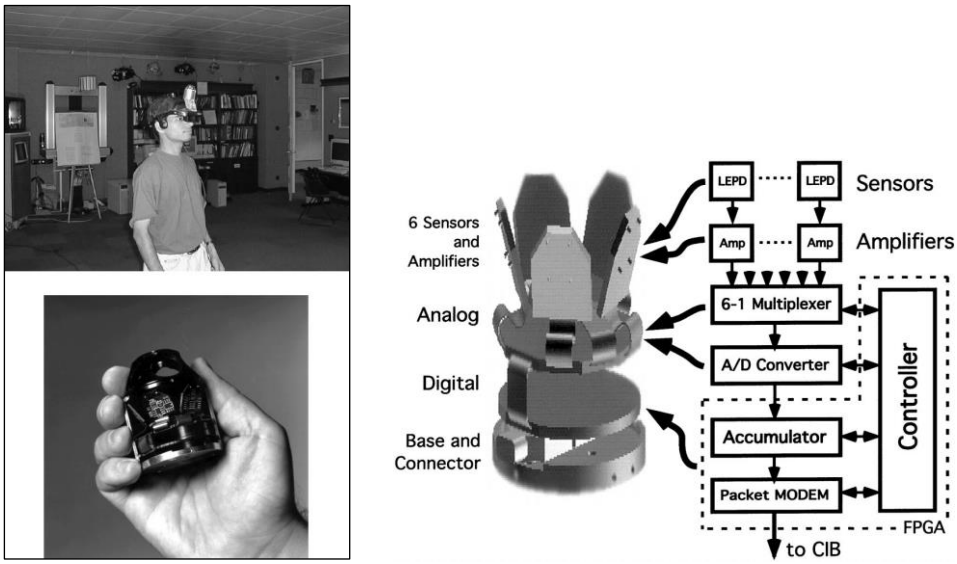


Figure 2.2 : Active Markers – The HiBall Tracking System (Welch, et al., 2001)

Figure 2.3 illustrates the ©Vicon motion capture system with sphere-shaped passive markers covered with retro-reflective material, which reflects the infrared radiation of the incoming light.

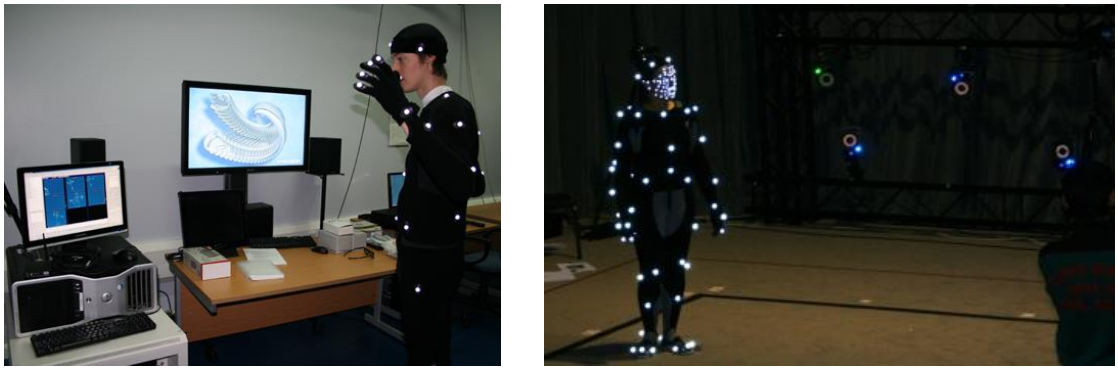


Figure 2.3 : Passive Markers (©Vicon motion capture system)

2.1.1.2 Fiducial Planar Markers

Visual fiducial markers have been extensively used in marker-based tracking systems for higher reliability and accuracy. Fiducial planar marker systems consist of unique patterns together with the detection algorithms to locate the projection of patterns in the camera image. The reliability of such marker systems depends on the proper design and manufacturing of the markers as well as the performance of pattern detection algorithms and camera calibration for accurate pose recovery using homography.

In practice, computer vision systems often use 2D patterns to carry information, very similar to barcodes seen on consumer products. In order for a vision system to recognise the details on a barcode, the distance between the marker and camera must be kept relatively short. Also, a distinction must be made between visual patterns such as barcodes, designed to convey information, and patterns designed for the purpose of tracking.. Examples of the former include the Maxicode markers used by US postal services to carry shipping information, DataMatrix and respectively the Quick Response (QR), which are both used in industrial settings for part labelling (Zhang, et al., 2001).

For camera pose recovery purposes, using a 2D visual pattern, at least three points of correspondence must be identifiable between the 2D pattern and their camera image. As a consequence this three point perspective pose estimation

problem and its solutions (Haralick, et al., 1994), the majority of the visual markers used in marker-based systems specifically for indoor augmented reality applications have been designed within a quadrilateral shape i.e. a square with at least 4 co-planar corresponding points. This allows feature correspondence and subsequent homography calculation even when there is only a single marker in the scene (minimum 3 points are available in a single marker). In some cases, circular visual markers have been used, providing only one point correspondence per marker, i.e. the centre of the circle, such that at least 3 markers with known 3D positions must be used for pose estimation.

Martix (Rekimoto, 1998), ARToolkit (Kato & Billinghurst, 1999) and (ArToolKit, 1999) , ARToolKit Plus (Wagner & Schmalstieg, 2007), BinAryID and ARTag (Fiala, 2005), IGD, HOM, SCR and the Cannon markers from the ARVIKA project (ARVIKA, 2009) use square planar markers producing quadrilateral perspective projection in the camera image. Some of these vision markers have been illustrated in Figure 2.2.

Fiducial planar marker systems are largely based on two main stages; hypothesis generation from detecting unique features and the verification/identification stage. The former stage is to detect the planar patterns based on perspective projection and the homography between the marker pattern and the image plane. A geometric shape such as dot, bar, ellipse, triangle, square, etc. provides an anchor to form the marker detection hypothesis. Figure 2.2 illustrates some of these geometry shapes and patterns used for feature detection. At this stage, normally more than one feature is used to provide a list of regions with homography presented to the next stage for verification to check against other similar objects in the scene. Several systems use blob detection techniques to find the connected components of interest in a binary image formed by thresholding in order to find the unique features.

The second stage of verification and identification is to determine whether the detected features correspond to the fiducial markers or similar objects in the

scene or not. Considering various such methods, ARTag claims to have an enhanced verification stage in comparison to other systems such as ARToolkit (Fiala, 2010). It generates a binary yes and no system for verifying whether the detected object is in fact a fiducial marker or not. Fiala (Fiala, 2010) established eleven evaluation criteria in order to address the performance, usability and robustness of fiducial markers. Subsequently he carried out a set of experiments to assess the effectiveness of the ARTag system considering the above criteria. The results of these tests have been depicted in Figure 2.3.

One image was captured with an array of ARTag markers and one with an array of ARToolit Plus markers. Both images had the same size and arrangement of markers. the so-called ‘inter-marker confusion rate’ and the ‘false negative rate’ were precisely measured. Figure 2.3 (a) shows the ‘Inter-Marker Confusion Rate’ criteria in ARTag and ARToolKit Plus Systems. The diagram provided in Figure 2.3 (b) shows the ‘False Negative Rate’ criteria in ARTag and ARToolKit Plus Systems. The ‘inter-marker confusion rate’ is one of Fiala’s eleven evaluation criteria for performance assessment of fiducial markers, which indicates whether a wrong ID was reported or a marker was mistaken for another. ‘False negative rate’ is also referred to as the probability of the presence of a marker in the image, although not being reported by the detection algorithm.

In this analysis, Fiala (Fiala, 2010) modelled the marker system as a communication system and the ‘Hamming distance approximation’ (Hamming, 1950) was used for measuring the approximate distance between the markers in order to minimise ‘inter-marker confusion rate’. The results of this experiment and analysis showed an improved performance from ARTag markers by several orders of magnitude in comparison to ARToolKit (as seen in Figure 2.3).

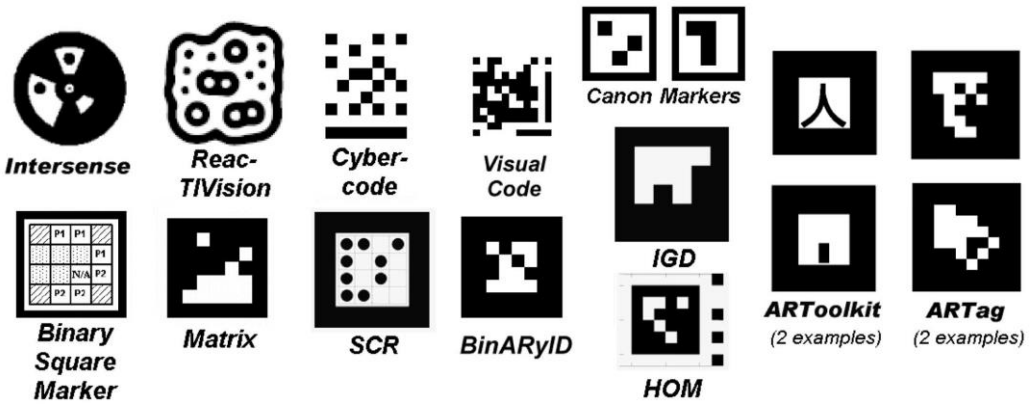
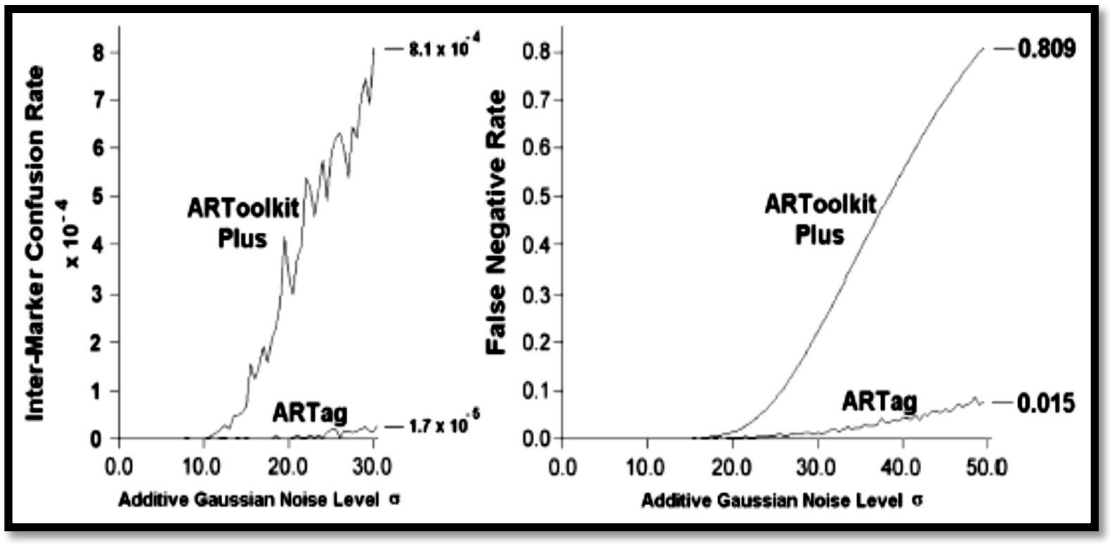


Figure 2.2: Several fiducial planar marker systems – source: (Fiala, 2010)



(a) (b)

Figure 2.3: Usability test results comparing ARTag with ARToolkit Plus - Source: (Fiala, 2010)

2.2 Global Positioning System (GPS) for Location Tracking

The Global Positioning System (GPS) relies on a number of satellites and ground stations spread around the world. It is one of the most ubiquitous means of wide-area location tracking and works anywhere on the Earth's surface provided there is an unobstructed line of sight to a sufficient number of GPS satellites. In practice at least four satellite signals are required to triangulate receiver location. Each satellite has atomic clocks with a current drift

rate of about 0.1 milliseconds per year, which creates a distance error on the order of 30m. However, ground stations control the satellites drift and recalibrate the atomic clocks every 30 seconds to reduce the measurement drift errors. The typical resolution of such GPS location tracking systems is of the order of few metres (Elliot, 1996).

More precise GPS tracking is provided by differential GPS systems, which use emitting ground stations to refine the resolution to the order of one metre (Noe & Zabaneh, 1994). However, location accuracy achievable by consumer-grade GPS receivers is variable depending on environmental conditions but can typically be expected to be within 10 metres. Real-time positional accuracy can be improved by counteracting errors caused by satellite orbit irregularities and atmospheric conditions using differential GPS. This uses ground stations to calculate correctional data, which is uploaded to GPS satellites to be broadcast to the enabled receivers. This augmentation of the raw satellite data is available in regions of the Earth with an appropriate network of ground stations. For example, the Wide Area Augmentation System (WAAS) covers North America and Hawaii, while European-based stations provide the European Geostationary Navigation Overlay Service (EGNOS). These augmentation systems improve the reliability and accuracy of the GPS system to better than 7 metres laterally with some locations experiencing better than 2 metres.

Figure 2.4 : GNS 1000 WAAS Enabled GPS Receiver

2.3 Inertial Tracking Systems

Inertial Navigation Systems (INS) became widespread for ships, submarines, and airplanes in the 1950s. Initially, INS contained heavily weighted, high accuracy, spinning-wheel gyroscopes such as the one illustrated in Figure 2.5.

Despite their accuracy; they weigh far too much to be considered for head mounted displays or as a conventional input device. However, with the advent of MEMS (Micro-Electro-Mechanical Systems) inertial sensors in the 1990s, they became feasible to be considered as input devices and for attachment to moving bodies for tracking.

In principle an inertial measurement unit operates by trying to conserve either a given axis of rotation, as in the case of a mechanical gyroscope, or a position, as in the case of an accelerometer. Inertial Measurement Units (IMUs) contain three orthogonal gyroscopes and three orthogonal accelerometers measuring angular velocity and linear acceleration. However, despite their accuracy in agile motions, inertial sensors suffer from accumulation errors over time due to the fact that each measurement is relative to the previous.

Almost all INS systems fall into two categories; Stable Platform and Strapdown. Figure 2.6 illustrates the Stable Platform in which the inertial sensors are mounted on a platform isolated from any external rotational motor (Woodman, 2007). The platform is mounted using gimbals (frames) which allow rotational freedom in all three axes (Figure 2.6). This aligns the body frame (platform) with the reference frame. The platform mounted gyroscopes detect any rotations and feed this back to the torque motors which then rotate the gimbals to cancel out any such rotations. This keeps the platform aligned with the global frame.

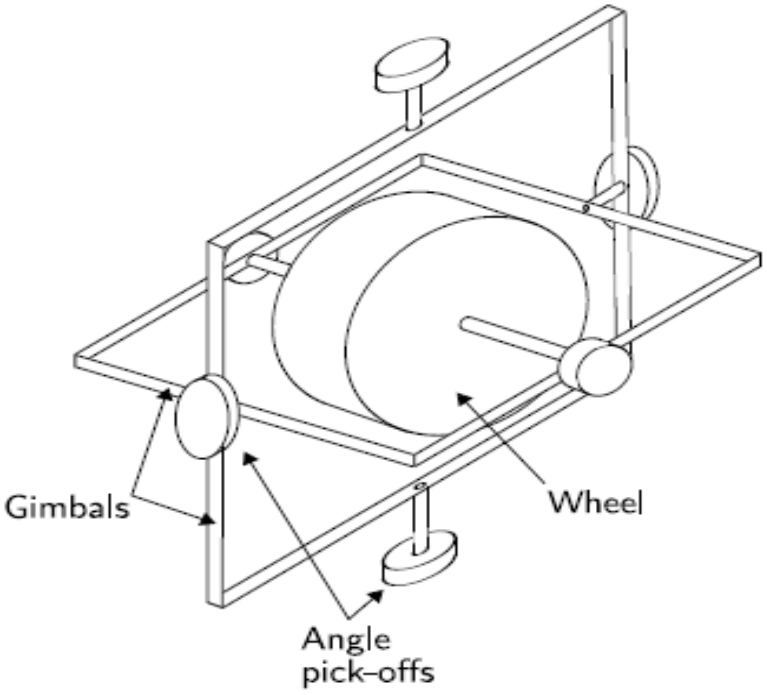


Figure 2.5 : A conventional mechanical gyroscope - source: (Titterton & Weston, 2004)

The orientation of the device is determined by reading the angles between adjacent gimbals using angle pick-offs. The position can be calculated by double integrating the signals from the platform mounted accelerometers.

The second type of INS, Strapdown INS, is lightweight and permits the system to eliminate the mechanical gimbals. The inertial sensors are simply strapped to the moving object and measure the orientation and translation by integrating the angular and linear velocity produced by the gyroscope and accelerometer. Therefore output quantities are measured in the body frame rather than the reference or global frame.

Figure 2.8 illustrates the algorithm for a Strapdown inertial system.

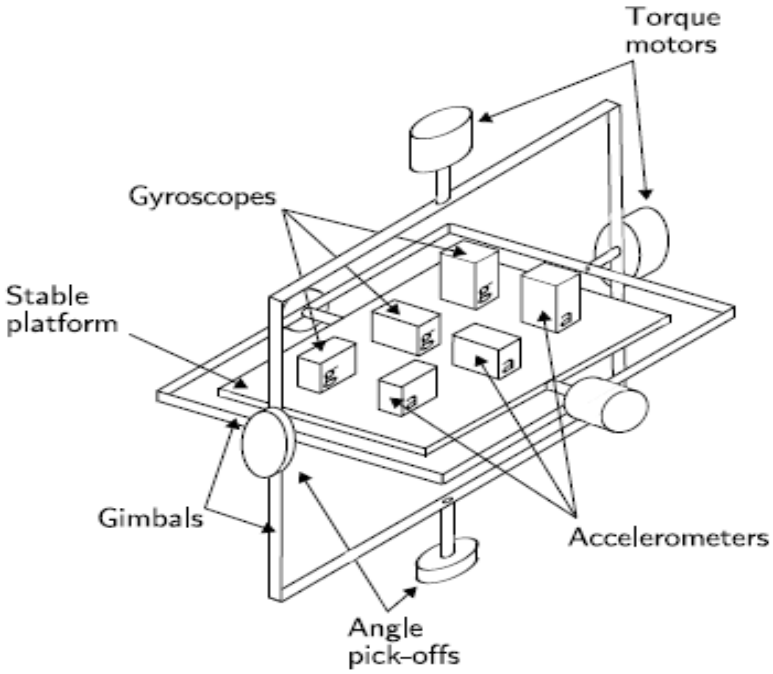


Figure 2.6 : Stable Platform IMU - source: (Woodman, 2007)

MEMS inertial measurement units are of the Strapdown type. There are various types of MEMS accelerometers, mainly based on the linear accelerometer model, where a proof mass is suspended between two springs. The displacement of the springs is proportional to the actual acceleration.

The acceleration output needs to be integrated once to provide the linear velocity and then integrated again to provide the linear displacement. Due to noise and error in acceleration measurement, and also the residual effect of gravity cancellation, the integration of accelerometer data often leads to significant drift over time, adversely affecting the registration stability. In particular, the integration of accelerometer data is known to introduce instabilities when used for position estimation.

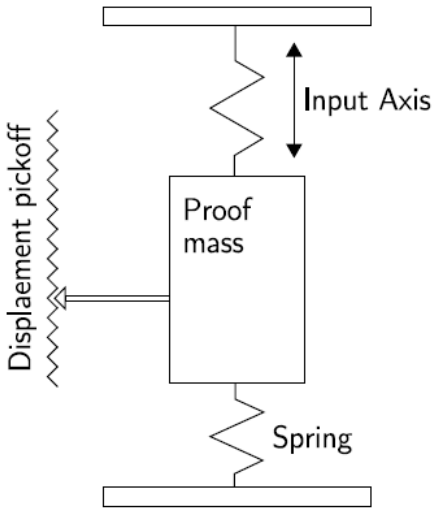


Figure 2.7 : Linear accelerometer. Source: (Woodman, 2007)

Nonetheless, despite the computational complexity of the Strapdown INS and the potential accumulation drift, they are still the foremost type of INS, due to their reduced mechanical complexity. Recent improvements in the performance of small and lightweight MEMS have made Strapdown MEMS devices the dominant inertial sensors used in navigation and tracking systems. It is beyond the scope of this thesis to present a comprehensive review of the architecture of MEMS devices. Full details about the MEMS-based IMU architecture can be found in (Woodman, 2007).

Examples of MEMS inertial sensors include standard Inertial Measurement Units (IMUs) such as MPU-9x50TM from InvenSense with up to nine degrees of freedom (3 from accelerometer, 3 from gyroscope and 3 from compass). It combines the 9DOF in a chip together with an on-board Digital Motion Processor™ (DMP™) capable of processing the complex 9-axis MotionFusion algorithms with more accuracy (InvenSense, 2003).

Given their agility and accuracy advantages in high speed motions, as well as their typical disadvantages such as inherent accumulation error, additive electronic, ambient noise, and gain noise, inertial sensors are particularly

valuable when combined with other sensing technologies in the form of so-called hybrid tracking systems.

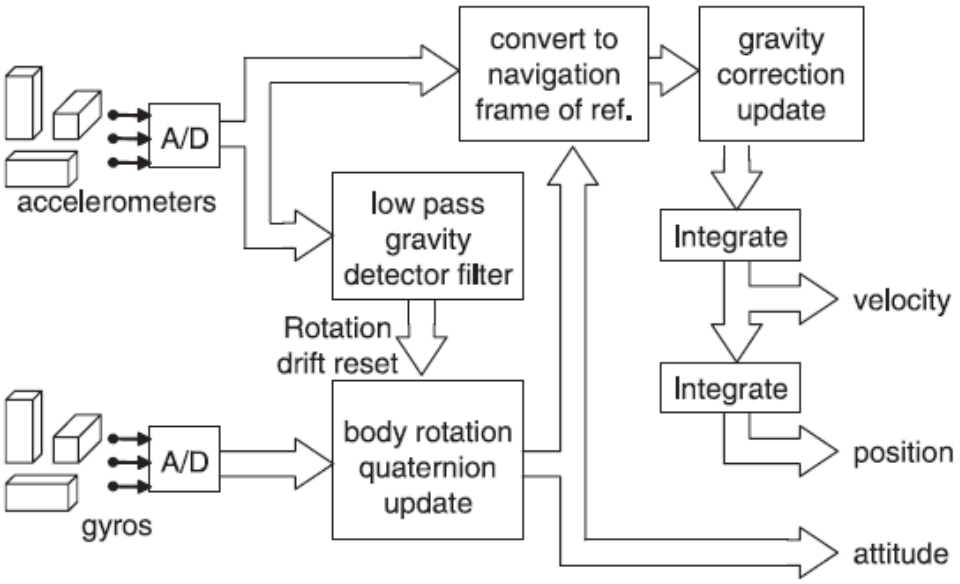


Figure 2.8 : Basic Strapdown INS - Source: (Corke, et al., 2007)

2.4 Tracking using Time of Flight (TOF)

Imaging technologies based on time-of-flight (TOF) measurements have also been considered as a solution to the problem of tracking and localisation. Such techniques can be based on acoustic (ultrasonic) signals, and optical signals in TOF cameras and the CCD sensors. The distance from sensors to the object (depth of the object) can be measured using TOF techniques, using light or ultrasound signals for this purpose. TOF resolves the distance based on the known speed of light or sound, by measuring the time of flight of a signal between the sensor and the object in the scene and back.

TOF techniques for positioning and localisation are more dominant in urban canyon environments where GPS signals are either unavailable or unreliable. Another of application is in biomedical and surgical environments where placing

invasive sensors or using other standard tracking techniques is not practical due to anatomy of human organs or because of the presence of fluid. In such applications most popular tracking methods employ radio frequency (RF) or ultrasound to determine the location of the target object in 2D or 3D by recording the round trip TOF of mentioned signals.

2.4.1 Acoustic Tracking using TOF

Acoustic systems use the transmission and sensing of sound waves. Historically, ultrasonic pulses are used in acoustic ranging systems, which operate by timing the signal flight duration. These ultrasonic-based tracking systems generally consist of three or more emitters on the target and three or more on the receiver. The emitter and receivers are transducers, which are attached to the object in a triangular arrangement. The emitted frequency is above 20KHZ, typically around 40KHZ, to prevent the users from hearing it. One of the main advantages of using ultrasonic TOF is its resistance to distortions. However, its accuracy depends on the consistency of the sound velocity as the speed of the sound varies with temperature, pressure, humidity, and turbulence. The other limitation in using ultrasound TOF is loss of energy of the signal with distance travelled, which limits the tracking range. Therefore this suggests it is not the best solution for wide-area tracking applications. In addition, ultrasonic waves have low update rate due to sequential triple emission of sound signals and the low speed of sound. Nevertheless, there are techniques that can be applied for increasing the update rate, such as coding the signals to be sent with variable frequencies (Rolland, et al., 2001). The systems described by (Holm, et al., 2005) and (Holm, 2005) are successful examples of ultrasound-only tracking systems for indoor positioning. The core of the system was a 40 KHz ultrasound communication system with an attainable tracking range of 10-20 metres. The main hardware was an Ethernet interface using digital signal processing to handle the acoustic environment and its noise, reverberations and Doppler shift.

Acoustic systems require maintaining the line of sight between the emitters and the receivers. Although this can be regarded as a disadvantage, their tolerance to occlusions is much higher than typical optical systems, as the sound wave can find its way through and around the obstacles more easily.

Hybrid tracking systems often tackle the limitations of ultrasonic-only systems; mainly the issues with low range accuracy, low robustness to external disturbances and occlusion of the line of sight. Among all such existing hybrid systems, combination of ultrasound with radio frequency (RF) has the largest number of applications. Examples of ultrasound-RF hybrid systems include; the Active Bat system (Ward, et al., 1997), the Cricket system (Priyantha, et al., 2000), and the Dolphin system (Fukuju, et al., 2003) with a reported accuracy of about 15 centimetres.

All of above mentioned systems combine ultrasound with RF (or RFID tags) and work based on the estimation of TOF considering the slow travel of ultrasound compared to RF. However, the main requirement of such systems is still unobstructed line-of-sight. The other hybrid system, which addressed the shortcomings of acoustic-only systems, is ultrasound-inertial tracking. An example is the hybrid inertial sensors and wireless ultrasound tracking system designed by InterSense Inc. which was developed by Azuma in his first Videometric-Inertial technology system (Azuma, 1995). In such a hybrid system, the inertial sensors provide the basic accuracy and ultrasound (at a lower update rate), which when available, is used to reset the drift generated by the accelerometer.

2.4.2 Optical Tracking Using TOF

In recent years, optical signals have been used for localisation and 6DOF tracking based on TOF calculation using state-of-the-art of TOF cameras. The TOF 3D cameras come as a compact solid-state sensor, which provide range and amplitude images at a video frame rate. They emit near infrared (NIR) signals to the objects in the scene and the reflected light is measured via a CCD

or CMOS sensor. The distance is then calculated using phase-shift principle. More details can be found in (Ratshidaho, et al., 2012).

Examples of commercial TOF cameras are the Swiss Ranger SR4000/SR4500 and SR4050 SLIM (MESA, 2006) with their associated visualisation software (SR-3D-View), which provides the depth information in colour coded form from violet (close) to red (far) (see Figure 2.9 for sample data captured by SR4000/4500).

TOF cameras have been used for solving the pose estimation problem due to their low power consumption and compactness compared to laser range finders, as well as being able to produce 3D range images at video frame rate (approximately 30 fps). However, depending on external noise factors such as sunlight, scene configurations, distances, surface orientations, and reflective distance measurements from various perspectives of the same scene, TOF cameras can cause significant error and large fluctuations in accuracy and precision. In addition, their narrow field of view makes the registration of the range images challenging. As a result, 3D laser scanning techniques are still mostly used for 3D mapping purposes (Ratshidaho, et al., 2012).

In terms of image registration in TOF-based tracking, the standard Iterative Closest Point (ICP) algorithm (Besl & McKay, 1992) and (Chen & Medioni, 1991) is normally used. The original ICP algorithm consists of two main steps. Having two sets of points, it is first necessary to identify pairs of candidate points for likely correspondence and, second, to compute a transformation that minimises, in a least-squares format, the distance between the two sets. This process is repeated until a convergence criterion is met.

TOF is widely used for ego-motion estimation in special environments. As an example, the work by Ohno, et al (2006) presented a rescue robot for collecting information in a rubble-strewn environment for Japan's MEX1 special project concerning an earthquake disaster management system. They proposed a solution involving 3D SLAM for rescue robot localisation and mapping in a special environment; i.e. earthquake rubble. In their work they used a TOF

Swiss Ranger 3D camera for robot trajectory estimation. They also applied a modified ICP algorithm to handle the TOF error rate in image registration. The edge detection was used for feature tracking in corresponding points extracted from the amplitude image (from TOF camera). Accuracy of 17% in rotation and 15% in translation was achieved by their hybrid system (Ohno, et al., 2006).

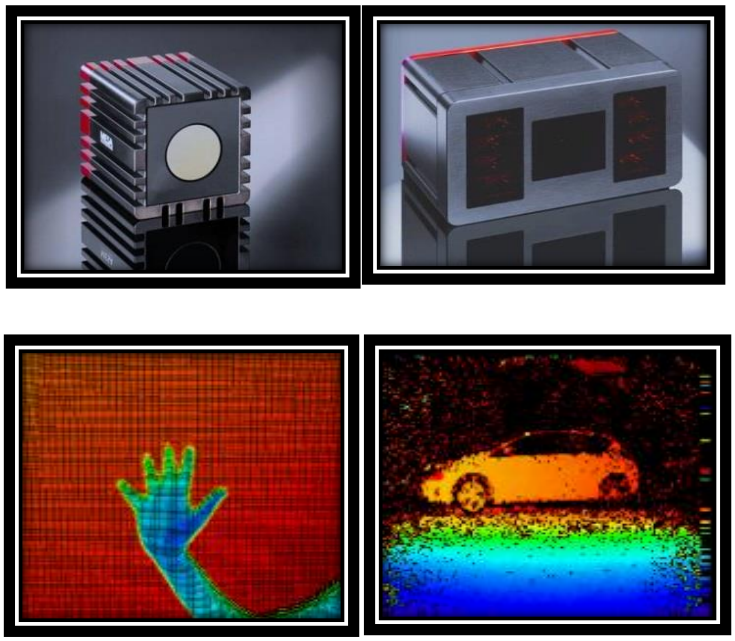


Figure 2.9 : Time of Fligh Cameras

Up Left to Right: SR4000, SR4500 & Bottom Left to Right: Sample data respectively.
Source: (MESA, 2006)

Another example is the work of May, et al (2009) which used a SR3K Swiss Ranger camera for motion estimation and map building. They also modified the original ICP algorithm to handle degree of overlap caused by the small field of view of TOF and mismatched correspondence points. Also, their approach was compared to SIFT (Scale-invariant feature transform) and KLT (Kandale-Lucas-Tomasi) for feature-based pose-tracking applied on the amplitude images captured from the TOF camera. Similarly, (Wang, et al., 2009) used a SR3K TOF camera and applied SURF (Speeded-Up Robust Features) feature

detection on amplitude images, which provided more accuracy compared to its application on a standard image.

The 3D TOF range-camera models, description of the error and error handling techniques are outside the scope of this thesis, however more details can be found in (Ratshidaho, et al., 2012), (Hochdorfer & Schlegel, 2010) (Ratshidaho, et al., 2012) and (May, et al., 2009).

2.5 WIFI Sensors for Tracking

Wireless Fidelity (WIFI) based indoor positioning and localisation using wireless proximity detection have been studied for more than a decade and a variety of systems, methods and algorithms have been proposed such as the RADA (Bah & Padmanabhan, 2000) by Microsoft Research, as well as other systems including those introduced by (Kjrergaard & Munk, 2008), (Subramanian, et al., 2008), (Ahamed, et al., 2008) and the Enhanced Localisation Solution (ELS) by (Papandrea & Giordano, 2012). Localisation in such systems is normally achieved through Received Signal Strength (RSS) measurements. The RSS decreases when the receiver moves away from the emitter, therefore the signal strength is used as an observable measure to estimate the distance from the emitter station. The RSS values can be obtained with minimal effort or the need for additional hardware, as most radio chips are natively equipped with an RSS indicator which returns a digital value with the average signal power of the received packet.

RSS-based proximity and localisation techniques and algorithms have been widely analysed and empirically tested in recent years. Some interesting results can be found in (Pivato, et al., 2010) and (Pavani, et al., 2006). However, the main drawback of RSS-based systems is their significant sensitivity to multipath and shadowing effects, which are particularly critical in indoor environments. This perturbs the ideal relationship between the RSS and the distance, thus

leading to biased and space-varying position estimates and unsatisfactory localisation accuracy (Pivato, et al., 2010).

Most recently, in 2013, ©Apple introduced iBeacons which have been mainly developed for indoor proximity detection. IBeacons are small, low-cost wireless transmitters which broadcast signals using Bluetooth Low Energy (BLE) standard. Mobile Apps (iOS 7.0 and Android) work as receivers and listen for signals from beacons in the physical world. Initially it has been widely promoted for use in the retail industry. An example is ©Exact Editions which launched their iBeacon service for publishers to make their magazine apps free of charge at certain locations in London such as in a café, a hotel or first class lounge (ExactEdition, 2013). Although iBeacon was mainly advocated for proximity detection in the retail industry, ©Apple has argued that the iBeacons have also been designed for micro-location tracking.

Presented in this thesis is an experiment performed by the author using iBeacons to assess the accuracy of RSS-based tracking system for micro-location tracking considering the signal strength. The experiment demonstrated the fairly low accuracy of these devices for micro-location estimation. In the experiment, an array of few iBeacons was placed in an indoor environment. The iBeacons were arranged at three corners of a 10x10 grid of tiled carpets where each tile was a square of 0.5 x 0.5 metres. The details of this experiment and results have been described in Chapter 4.

2.6 Multisensory Tracking Approaches

Multisensory tracking, or simply hybrid tracking, refers to the combination of different tracking technologies into a single system; such as Ultrasound-RF (Fukuju, et al., 2003), Inertial-TOF (Droeschel, et al., 2011), GPS-Visual (Schleicher, et al., 2009) , RSS-TOF (Macii, et al., 2011)] and Inertial-Visual (Bleser, 2009) and (Scaramuzza, et al., 2014) tracking systems.

It is beyond the scope of this thesis to present a full review all permutations of hybrid systems. However, of particular relevance to this thesis, a review of significant inertial-visual hybrid tracking systems is given in section 2.6.1. The important enabling principle of recursive filtering is discussed in Chapter 3. Subsequently, an improved inertial-visual hybrid camera pose tracking system has been developed, a full account of which is provided in Chapters 6 and 7.

Chapter 4 of this thesis also presents another hybrid system - GPS-Visual SLAM. An iOS mobile app was accordingly developed to trial this idea and collect the results for further analysis.

2.6.1 Inertial-Visual Sensor Fusion

The integration of vision and inertial sensors started with the early work of Viéville & Faugeras (1990) and has grown in interest and application. The advantages of this system integration were well described by Corke, et al. (2007).

Inertial sensors are unable to distinguish a change in inclination from acceleration of the body, due to Einstein's equivalence principle. These sensors have large measurement uncertainty at slow motion and lower relative uncertainty at high velocities. However inertial sensors can measure very high velocities and accelerations. Cameras on the other hand can accurately track features at low velocities. With increasing velocity, tracking is less accurate due to motion blur and the effect of camera sampling rate. For high velocities and accelerations cameras with higher frame rate can be used up to a point, but the increase in bandwidth complicates real-time implementations. Image-based tracking systems also suffer from a missing dimension due to 2D/3D transformation. Therefore a near object with low relative speed appears the same as a far object with high relative speed (Broida, et al., 1990).

In recent years hybrid tracking systems consisting of low cost inertial measurement units (IMUs) and robust and high-dimensional computer vision-aided algorithms have enhanced the performance and agility of tracking systems such as those described in (Lobo & Dias, 2003), (Bleser & Stricker, 2008) and (Corke, et al., 2010). Such solutions have also tackled the hurdles of real-time sensing and localisation especially in GPS-denied environments (Mingyang & Mourikis, 2012), (Scaramuzza, et al., 2014). The generic architecture of the IMU-Vision sensor fusion is well modelled and justified by (Corke, et al., 2007), in which the authors also analysed the main advantages of each system in detail.

Foremost examples of inertial-visual tracking systems include Marker-less Real-time Tracking for Augmented Reality Image Synthesis (MATRIS) as well as Visual-Inertial SLAM.

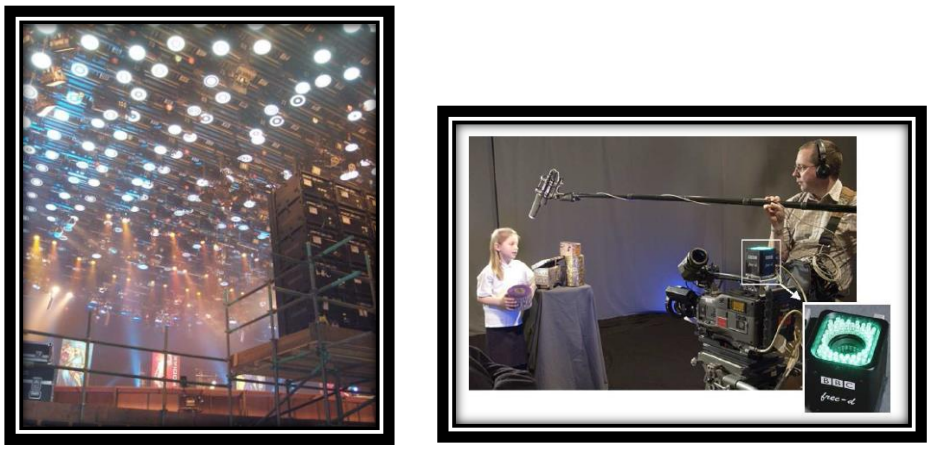


Figure 2.10: Virtual Studio – BBC-developed Free-d System

Left: “Circular barcoded markers mounted on the ceiling of a TV studio”
Right: “free-d tracking camera mounted on studio TV camera
Source: (Thomas, 2007)

The EU-funded project MATRIS (Chandaria, et al., 2007) was a tracking system mainly used in real-time augmented reality for film and TV production applications. MATRIS was initially developed as replacement for use in existing television production studios, which were mainly using a marker-based solution

such as the BBC-developed free-d system for real-time augmented reality (Thomas, et al., 1997), (Thomas, 2007) (see Figure 2.10).

The MATRIS was designed in the form of a model-based hybrid system which used structure-from-motion methodologies in order to create a 3D model of the scene offline before the real-time tracking and image registration process. This provided stability in tracking as well as an absolute reference frame for composition of virtual objects in a fixed location with known scaling and orientation for a repeatable performance. MATRIS benefited from the agility and accuracy of an IMU for compensation of latency and missing dimension of image-based system. The search and optimisation based technique Random Sample Consensus (RANSAC) was used for removing the outliners during the matching of dynamic tracked data with the stored data from offline model.

Another example is Visual Inertial SLAM tracking for augmented reality developed by Gabriele Bleser (Bleser, 2009). Bleser proposed and described the development of another model-based system which fused inertial and visual measurements using Extended Kalman Filtering (EKF). However, the system still relied on a ‘partially’ known environment, with the need for an offline CAD model of the scene similar to MATRIS systems. Bleser also experimented with the Marginalised Particle Filtering (MPS) method to compensate for the restrictions of EKF in linearization and limited area tracking (for more details on recursive filtering methods for sensor fusion refer to Chapter 3).

Advances in MEMS-based inertial sensors have enabled pose-estimation in systems such as mobile robots or unmanned micro aerial vehicles (MAVs), often operating in urban canyon environments where GPS signals are either unavailable or unreliable. Recently, there has been substantive research and progress in autonomous MAVs such as the EU-funded SFLY (Swarm of Micro Flying Robots) project (Scaramuzza, et al., 2014).

2.7 Summary

In the past decades, there have been several advances in single camera pose tracking techniques ranging from marker-based to marker-less approaches. The passive/active visual markers, inertial sensors, RF-based technologies such as application of GPS, radar, acoustic sensors and wireless sensors have been used either on their own or as part of a multisensory model. However, most systems reviewed and investigated in this chapter provide less accurate outcome in applications where wide-area tracking is required or there is a need for tracking in uncontrolled environment with no prior information i.e. model-free systems.

Although GPS is regarded as the most ubiquitous means of wide-area location tracking, it still suffers from ranging errors due to obstruction of line of sight. However, in recent years the accuracy of GPS's localisation has been improved by the design of more precise differential GPS systems, though wide-area tracking and localisation is still an ongoing problem in GPS-denied environments such urban canyons, battlefields, hospitals and shopping malls. Furthermore, multisensory approaches were reviewed in this chapter as a solution to enhance the performance of the single sensory approaches.

Finally, inertial-visual tracking systems were investigated considering the inherited characteristics of state-of-art of MEMS-based inertial sensors as well as the advances in computer vision algorithms for pose recovery. The latter is will be reviewed in more detail in the next chapter of this thesis.

CHAPTER THREE

3 A Review of Pose Tracking Techniques using Computer Vision Algorithms

Computer vision-based pose tracking methods are primarily based on detecting the natural features in an image using image processing algorithms. These methods estimate the camera pose relative to real world objects and are analogous to closed loop systems, which correct errors dynamically. After initially calculating the camera pose from known visual features, the system dynamically obtains additional natural features and uses them to continuously update the pose calculation. The rationale underlying all feature-based methods is to find a correspondence between 2D image features and their 3D world-frame coordinates. The camera pose can then be determined by projecting the 3D coordinates of the features into the observed 2D image coordinates and minimizing the distance to their corresponding 2D features.

In order to estimate the 6DOF of the camera, a set of at least three 2D/3D correspondences are required. Tracking a camera's 6DoF using 2D/3D correspondences can be regarded as an ill-ranked problem. This is due to the fact that the image formation - 3D to 2D transformation - results in missing a dimension, represented by a scaling factor as explained in Chapter 5. This dimension cannot be recovered unless additional information is provided through a 3D model, the presence of an object with known dimensions in the scene or using other sensors e.g. an IMU. 2D natural feature tracking is an essential step as its robustness reflects in the accuracy of the 2D/3D matching in the camera pose estimation process. 2D visual object tracking is concerned with tracking image features such as points, segments, object contours. A variety of techniques and algorithms have been developed for visual feature tracking; (Shi & Tomasi, 1994), (Blake & Isard, 1998), (Hanek & Beetz, 2004)

or regions of interest (Mayol & Murray, 2008) , (Hager & Belhumeur, 1998), which are represented in the 2D image plane.

On the whole, any purely computer vision-based camera pose tracking technique can be considered in two main categories. The first type of approach is to apply image processing techniques providing that some knowledge about the environment is readily available (online or offline) in the form of 3D scene geometry (*a priori* model) for determining 2D/3D correspondences. This approach is often referred to as 'model-based visual tracking' and has already been used widely in human computer interaction and augmented reality applications. Examples include the work developed by (Lowe, 1992) ; (Drummond & Cipolla, 2002) ; (Lepetit, et al., 2003) ; (Vacchetti, et al., 2004) ; (Comport, et al., 2006) (Irschara, et al., 2009) ; (Dong, et al., 2009) ; (Li, et al., 2010) and (Sattler, et al., 2011).

The alternative approach addresses the problem through Simultaneous Localisation And Mapping (SLAM) ((Dissanayake, et al., 2001) ; (Davison, 2003) ;(Montemerlo, et al., 2003) and (Durrant-Whyte & Bailey, 2006)) in which the 'robot' or 'camera' is tracked and localised in an 'unknown' scene, while a map of the environment is simultaneously constructed.

The early work of (Davison, 2003) was the foundation of Visual-SLAM, where the main motion sensor used for SLAM tracking was the camera. Some recent developments on variations of Visual-SLAM-based approaches include; (Simon, 2006), (Eade, et al., 2007), (Chekhlov, et al., 2007) and (Klein & Murray, 2007) and have demonstrated reasonable performance of Visual-SLAM in camera pose tracking for indoor augmented reality applications.

In both above-mentioned categories, the majority of tracking algorithms are iterative and rely on minimising particular error criteria through successive iterations. Computer vision and robotics communities provide similar approaches in solving this problem.

In this chapter, some of the above tracking techniques will be reviewed accompanied by algorithms with particular application in camera pose tracking.

Limitations of vision-only systems will also be considered, leading to an improved hybrid and sensor fusion system as the contribution of this study.

3.1 Model-Based Tracking Techniques

Model-based tracking is an established technique that has successfully been used in AR applications. In a model-based approach, an *a priori* model of the tracking environment is made available either offline or online. This provides for a closed-loop approach in estimation of 2D/3D correspondences as it compensates for the unknown scaling factor when a 3D model of the scene becomes available.

The main limitation of model-based methods is that they require a sufficiently accurate model of the tracking scene. Also such models may not always be available or may become out of date if the structure of the environment is modified. This also raises the question of whether the model of the environment can be constructed online during the operation of the tracking system or not. Nevertheless, the Visual SLAM methods have made a significant step in overcoming the limitations of model-based tracking systems in recent years.

Historically, the tracking system developed by (Gennery, 1992) was one of the earliest 3D model-based motion tracking systems, which tracked Sobel edges within a 5-pixel range of predicted edges. This included velocity extrapolation and filtering. Gennery, (1992) also examined the probabilistic evaluation of feature matches to a model.

In addition, the RAPID 3D tracking system (Harris, 1992) utilised the basic approach taken by several model-based tracking systems. A set of 3D points was sampled along the model edges. Then in each frame, these points were projected into the image and a one dimensional search for edges normal to the projected model edge was performed. The changes in model pose were then calculated by minimising the distance between the projected points and the

detected image edges. The pose of the object was then tracked over time using a Kalman Filter with a constant velocity model. This was performed assuming the model only moved with a small variation in translation and rotation at each frame. The advantage of this method was its applicability to various types of edge feature, as well as having a relatively easy real-time implementation. However, it was found unreliable in situations involving rapid movement, where rotation and translation are more noticeable. Also, the original implementation was not robust to occlusion or false edge matches as it treated the tracked feature points as completely independent features, despite the fact that they often lie on the same edge in the model.

Accordingly, Armstrong, M. et al (1995) improved the original Harris's RAPID tracking system by using Randomised Sample Consensus (RANSAC) on each model primitive to detect outliners amongst the detected edge matches in an image. That improved the reliability of false edge detection by removing the calculated outliers. Also, instead of computing the pose updates for the full set of tracked control points, the pose update was calculated with each primitive deleted in turn and the measured projected error used to score each calculated pose. Poses with large projection error indicated that a false primitive must have been present and therefore it was removed and marked as an outlier. During pose estimation the stable model primitives were favoured by weighting primitives with a confidence value that reflected their stability over time (Armstrong & Zisserman, 1995).

Following a similar approach in model-based tracking, more recent algorithms have used robust M-estimation and iteratively reweighted least squares (IRLS) to provide improved accuracy and robustness to outliers (Drummond & Cipolla, 2002), (Comport, et al., 2005), (Comport, et al., 2006). For instance, Drummond, et. al (2002) proposed a framework for 3D model-based tracking using Lie Algebra to simplify the representation of the pose update. They used Lie group formalism in order to transform the motion problem into simple geometric terms. Therefore, the tracking became a simple optimisation problem

solved by means of iterative reweighted least squares (Drummond & Cipolla, 2002).

3.1.1 Structure from Motion (SFM) for Visual Tracking

Structure from Motion (SFM) is a model-based approach, which deals with simultaneous estimation of the camera trajectory (3D pose) and the 3D scene structure from a continuous 2D image sequence. This subject has been extensively studied in the past decade (Hartley & Zisserman, 2004) and (Faugeras & Luong, 2001). In recent years, computer vision methods for 3D scene reconstruction have become robust enough to be used by non-vision experts. Fully automated reconstruction systems are now able to reconstruct a scene from unordered images such as online photo collections.

SFM consists of two interrelated tasks, namely triangulation and localisation. On one hand, by having the exact 3D pose of the camera, the 3D structure of the scene can be obtained by triangulation of image correspondences. On the other hand, an existing 3D model of the scene allows for determination of the image pose directly by camera localisation using 2D/3D correspondences matching.

Model-based tracking and 3D reconstruction of the tracked scene are not within the scope of this study. Irschara in his PhD thesis has detailed the fundamental concepts and geometrical modelling required for SFM and simultaneous 3D reconstruction (Irschara, 2012).

Nevertheless, the established problem for SFM is the accumulation error in the camera and structure registration. Therefore, due to the high processing requirements, the accumulation error generation and consequent low frame rate and latency, SFM is often fully or partially completed offline. In an offline system, the drift is typically corrected by performing batch optimisation involving the whole image sequence, i.e. global bundle adjustment as in (Chiuso, et al., 2002.) and (Cornelis, 2004). More details on bundle adjustment and batch optimisation techniques can be found in (Triggs, et al., 2000).

In recent years there have been some developments in applying bundle adjustment in real-time systems by either performing optimisation and processing on only a subset of images (Cornelis, 2004) or by applying the optimisation as a parallel background process (Klein & Murray, 2007). However, despite the stability and scalability of these approaches, real-time large scale localisation and tracking is still a challenging problem in vision-based tracking systems.

The recent advancement of SIFT (Scale Invariant Feature Transform) (Lowe, 2004), and SURF (Speeded Up Robust Features) (Bay, et al., June 2008) have enhanced the performance of vision-based tracking systems. In summary, these techniques do not only rely on the detection of feature points, but also propose the use of a local invariant descriptor. These descriptors are used to identify unique feature points and match them under disturbing situations such as variations in scale, rotation, viewpoint, illumination or any other additional environmental unwanted noise.

This invariance criterion became the strong advantage of these algorithms in tracking mobile systems where the environment conditions were neither stable nor repeatable. However, due to the high computational costs, real-time and wide-area tracking remained a challenge and a subject of ongoing research in this context. As an example, we can refer to the SFM-based method developed by (Dong, et al., 2009) which performed continuous pose recovery using SIFT, while the key-frame recognition technique applied on video frames to recover the 2D-3D matches. This approach achieved a real-time system with the speed of 6 fps (for single thread), and 20 fps (for parallel thread).

Another example is the real-time image-based 6-DOF localisation system introduced by (Lim, et al., 2012) which was a compromise between the scalability of SIFT and its latency for real-time applications. Lim, et. al (2012) developed an algorithm for continuously localising a camera in a large scale environment, which had already been reconstructed using SFM. They used a fast tracking method and binary feature descriptors (BRIEF) (Calonder, et al., 2012) to find the best frame-to-frame match. However, for 2D/3D

correspondence matching within the SFM, they applied DAISY descriptors (Tola, et al., 2010); an expensive computation technique. The key distinction of their approach, was avoiding the need for scale-invariant descriptors at runtime and not relying on SIFT for feature tracking. However, in order to make sure the 2D-3D matching was reliable, they had to perform an offline computation to eliminate redundant descriptors for each 3D point in the SFM reconstruction. They exploited the spatio-temporal coherence to reduce the per-frame latency. Their single-threaded algorithm ran at an average frame-rate of 30 Hz on a laptop and at 12 Hz on a low-power, mobile computer suitable for on-board computation on a micro aerial vehicle. The performance of (Lim, et al., 2012)'s algorithm was reported as being five times faster than the frame rate achieved by (Dong, et al., 2009).

3.2 Simultaneous Localisation and Mapping (SLAM)

Another alternative approach for 3D pose recovery is the Simultaneous Localisation and Mapping (SLAM) technique, which was originally used in robotics for localising a mobile robot while incrementally building a map of an unknown environment. Comprehensive surveys by (Thrun, et al., 2005), (Durrant-Whyte & Bailey, 2006) and (Bailey & Durrant-Whyte, 2006) explore the SLAM problem and its solutions in more detail. Among all computer-vision based techniques with a monocular camera sensor, visual SLAM is regarded as the most widespread method used for localisation and 6DOF tracking. An extensive review on recent developments of visual SLAM can be found in Gee's PhD thesis (Gee, 2010).

Standard SLAM, also referred to as Filter-based SLAM, mainly employs Bayesian recursive filters such as Kalman Filter (KF) and Extended Kalman Filter (EKF) to infer the current state (6 DOF pose) based on the past state observation of the system (refer to 3.4.3.1 and 3.4.3.2 for more details). Examples include (Azarbayejani & Pentland, 1995); (Davison, et al., 2003)

;(Davison, 2003) ; (Eade & Drummond, 2007), (Civera, et al., 2010) and (Strasdat, et al., 2010). However, in all identified methods, the state estimation process generates uncertainty for both the features and camera pose, which adds to the complexity of the system and makes the process computationally more expensive. Therefore, filter-based approaches in their original format, e.g. (Davison, et al., 2003), are not the suitable solutions for real-time and wider-area pose tracking due to the linearity assumptions for the Kalman Filter-based approach and computational costs which limit them to small-area indoor tracking.

Nevertheless, the Parallel Tracking and Mapping (PTAM) algorithm of Klein and Murray introduced an enhancement to the filter-based SLAM by splitting the simultaneous localisation and mapping tasks into two separate threads (Klein & Murray, 2007). The tracking thread detected the salient features in each camera image and compared the extracted feature points with the stored maps and thereby determined the camera pose. In addition, the mapping thread refined the orientation and position of the camera so that the error between the observed features and the projection of the map points into the current frame was minimised.

In the literature, PTAM is occasionally referred to as key-frame SLAM where the mapping thread uses a subset of all camera frames; i.e. key-frames to build a 3D-point map of the surroundings (Strasdat, et al., 2012). This process is called bundle adjustment in computer vision terminology (Triggs, et al., 2000). Generally, the key-frame SLAM approach has proved to out-perform the standard EKF-filter based SLAM as demonstrated in (Strasdat, et al., 2012).

However, even by taking into account the enhancement made by bundle adjustment and online batch optimization approaches (i.e. as in PTAM) this process is still considered to be computationally expensive and therefore more applicable in smaller workspaces as demonstrated by Klein and Murray in the

context of a small augmented reality workspace application (Klein & Murray, 2007).

Likewise, Visual Odometry (VO) is another similar method used in localisation and tracking. VO is defined as “the process of estimating the ego-motion of an agent (e.g. vehicle, human, and robot) using only the input of a single or multiple cameras attached to it” (Scaramuzza & Fraundorfer, 2011) and (Fraundorfer & Scaramuzza, 2012).

There are functional resemblances in Visual SLAM and Visual Odometry. The goal of Visual SLAM is to obtain a global and consistent estimate of the robot or single camera trajectory, which requires keeping a track of the map of the environment on a continuous basis in order to provide the loop closure; e.g. rerunning to the original location through an estimated path. In contrast, visual odometry aims to recover the path incrementally ‘pose after pose’ and uses optimization techniques to estimate the past poses; e.g. using windowed bundle adjustment. Therefore the goal of visual odometry is to estimate the local trajectory, and even if a map of the environment is used, it will only be used to assist with the accuracy of the local trajectory estimation.

Considering the shortcomings of purely Visual SLAM-based tracking systems, the hybrid algorithm proposed in this thesis has been primarily inspired from visual odometry and benefits from the advantages of both filter-based and key-frame based SLAM (see Chapters 6 and 7 for the system design and evaluation).

3.3 Augmented Reality using Vision-Based Tracking Algorithms

In recent years, model-based tracking - including image/video or 3D models - has become a widely used technique in mobile augmented reality and in solving

the problem of image registration. Endres, et al. (Endres, et al., 2005) have published an extensive survey and reviewed 29 software infrastructure and frameworks used in ubiquitous computing vision as described by Mark Weiser (Weiser, 1999) with a focus on augmented reality, intelligent environments and distributed mobile systems.

Most of the existing AR systems operate with a priori knowledge of the scene or in the presence of a map, or CAD model of the environment, or even a sparse map of fiducial markers known to be present in the tracking scene. However, a comprehensive map is not often available due to the fact that the objects of interest or fiducial markers may not constantly be visible to be measured. In the context of AR, a class of techniques known as extensible tracking (Klein & Murray, 2007) was introduced for tackling the limitations in range and quality of registration. In extensible tracking, the system attempts to add previously unknown scene elements to its initial map, and these then provide registration even when the original map is out of the sensing range (Park, et al., 1998) ,(Jiang & Neumann, 2001), and (Bleser, et al., 2006).

An example of model-based AR systems is the (Bleser, et al., 2006)'s algorithm which presented a semi-automated model-based tracking approach that required a CAD model of one object of the tracking scene. This model was only used for initialisation of the first camera pose and obtaining 3D features that lie on the model. The rest of the features (not part of the given CAD model) were then tracked frame by frame automatically. The camera pose and 3D structure recovery were achieved using RANSAC and LM techniques. Bleser enhanced the well-known Kanade Lucas Tomasi (KLT) feature tracker and achieved illumination and scale invariant tracking. However, this system is still not suitable for wide-area tracking due to its limitation for 3D reconstruction of distant objects. This is because the scene features can only be used for pose estimation after they are triangulated, and for distant objects this is not a reliable technique.

The closest model-free mobile AR system, which is an alternative to standard SLAM, is the PTAM of (Klein & Murray, 2007). This was developed on a 3G iPhone with a denser map with lower-quality features being tracked in a small workspace. As discussed in previous sections, Klein and Murray initiated the concept of tracking without any *a priori* map or deep understanding of the user's environment. However, due to its use of bundle adjustment techniques, this system still provides less scalability when the size of the map is increased. Therefore despite the accuracy and high performance of PTAM, it is still not a suitable marker-less mobile AR for wide-area tracking.

In recent years, considering the advances in the development of invariant descriptors in feature tracking and 2D/3D matching, a combination of these algorithms was used in order to enhance the performance of pose tracking specifically for mobile augmented reality. For instance, (Wagner, et al., 2010) combined two algorithms namely SIFT (only as the feature detector) and FERN (as a classifier for invariant descriptor) in a model-based tracking system to compensate for the drawbacks of each method for mobile tracking applications. Accordingly PhonySIFT, PonyFerns and PatchTracker were introduced and combined using Extended Kalman Filtering, which proved to be more effective as an extension to original SIFT and Ferns algorithms for pose estimation from planar targets in real-time on a mobile phone. Wagner, et al (2010) evaluated their tracking system considering the CPU performance of the mobile phones, on planer objects.

In addition, (Maidi, et al., 2011) introduced another method for pose tracking for mobile augmented reality. They focused on enhancing the feature points tracking using SURF as a reliable tracking technique under general variant environmental noises such as illumination/contrast changes, object rotation/translation, image blurring and occlusions. The solution they proposed for pose tracking was a hybrid technique combining both analytical and iterative algorithms. They used Extended Kalman Filter as an iterative method for pose estimation, by considering an analytical pose estimator, based on planar

homography matrix decomposition (Maidi, et al., 2010), to initialise the pose and improve the EKF's convergence.

3.4 Recursive Filtering for Hybrid Camera Pose Tracking

As discussed in Chapter 2, inertial sensors are used in sensing rapid motions due to their ability to capture high-frequency motion. Also the previous sections have summarised the advantages of vision-based tracking systems in terms of their accuracy and less propensity to drift. However, IMU and image sensors are both influenced by measurement noise and error, which adversely affect the accuracy of pose estimation. The noise and error cause an IMU-only tracking system to drift significantly over time, making it an unsuitable sensor to be used on its own for pose tracking. On the other hand camera-only tracking systems not only suffer from noise and measurement error, but also exhibit an inherent deficiency, which is the inability to estimate all 6 DOFs.

Inertial-visual hybrid tracking operates by the application of recursive filtering in the context of a state-space model. There are various techniques for recursive filtering, such as Kalman Filter (KF), Extended Kalman Filter (EKF) and Particle Filter (PF). The selection of the appropriate recursive filtering method depends on the state-space model.

These methods will be outlined in the following sub-sections. Also covered are some tracking systems based on recursive filtering. Once the current systems are reviewed, an introduction to the hybrid inertial-visual tracking system proposed as a contribution of this thesis is presented.

3.4.1 State Space Model

The state-space approach (Ristic, et al., 2004) is the most-commonly used method to model a dynamic system in digital control and monitoring systems. A

state-space model consists of two main components. The first one is commonly referred to as the system or process model, which describes the evolution of the system state with time, subject to the control input. The second one is the measurement or observation model, which relates the noisy measurements to the system state.

In summary, the state space model describes a system with a set of inputs, outputs and state vectors represented by $\mathrm{U(t)}, \mathrm{Y(t)}$ and $\mathrm{S(t)}$ as shown in Figure 3.1 where $\mathrm{S'(t)}$ is the first derivative of $\mathrm{S(t)}$.

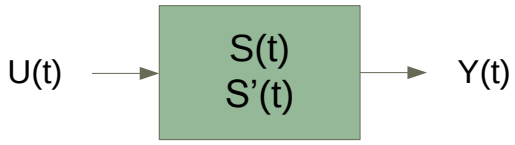


Figure 3.1 : State Space Model

The model describes the relationship between $U(t), Y(t), S(t)$ and $S'(t)$ vectors, using process and measurement functions $f(.)$ and $h(.)$ as it follows:

(3.1) $\boldsymbol{S'(t) = f(S(t), U(t))}$

(3.2) $\boldsymbol{Y(t) = h(S(t))}$

These equations provide the process and measurement models, respectively. The discrete representation of these equations is:

(3.3) $\boldsymbol{S_k = f(S_{k-1}, U_{k-1})}$

(3.4) $\boldsymbol{Y_k = h(S_k)}$

The above models formulate the case for a perfect system, where there is no error in modelling or measurement of the output or the control inputs. In practical applications, however, such effects need to be considered. Therefore equations (3.3) and (3.4) are re-written in order to include noise as well as measurement and modelling error, which are referred to collectively as noise. v and w are called process and measurement noise, respectively.

(3.5) $\mathbf{S_k = f(S_{k-1}, U_{k-1}, v_{k-1})}$

(3.6) $\mathbf{Y_k} = \mathbf{h}(\mathbf{S_k}, \mathbf{w_k})$

Here the recursive filtering techniques come into play. These methods provide an estimate of the correct state of the system using the noisy and erroneous measurement and process model. They first predict the current system state using equation (3.5), considering past system states, S_{k-1}, control input, U_{k-1}, and noise v_{k-1}. Once new observation data become available, equation (3.6) is used to correct the state prediction and update filter parameters.

3.4.2 Recursive Filtering

Filtering addresses the problem of estimating an unknown state of a system from a sequence of noisy observations or measurements made on the system as well as a sequence of known control inputs or input signals that carry information about the changes applied to the system. Recursive filtering is often considered in the context of probability theory. In such systems, three *probability density functions* (pdf) are often used; the transitional prior, posterior and likelihood probability density functions (Ristic, et al., 2004).

The transitional prior, $\mathrm{p}(\mathrm{S_k}|\mathrm{S_{k-1}})$, is the probability of having the current system state, given the previous state. The likelihood function, $\mathrm{p}(Y_k|\mathrm{S_k})$, is the probability of having the current observation, given the current system state. The transitional prior and likelihood functions are in fact another way of expressing state evolution described by equation (3.5) and the measurement model described by equation (3.6).

(3.7) $\mathbf{p}(\mathbf{S_k}|\mathbf{S_{k-1}}) \triangleq \mathbf{S_k} = \mathbf{f}(\mathbf{S_{k-1}}, \mathbf{U_{k-1}}, \mathbf{v_{k-1}})$

(3.8) $\boldsymbol{p}(\boldsymbol{Y_k}|\boldsymbol{S_k}) \triangleq \boldsymbol{Y_k} = \boldsymbol{h}(\boldsymbol{S_k}, \boldsymbol{w_k})$

The posterior density function, $p(S_k|Y_k)$, is the probability of having the current state of the system, given the current observation data. The aim of the recursive

filtering is to estimate the posterior function, by having the transitional prior and the likelihood functions.

In recursive filtering approaches, the observations and controls are processed sequentially rather than as a batch. It is also assumed that a Markov model is applied, therefore the current state is a complete summary of the past, implying that it is neither necessary to store the entire data set nor to reprocess the existing data when a new measurement or input data become available. This significantly reduces the computational cost of the filter.

Recursive filtering is often applied in two stages; prediction and update. The prediction stage uses the past system state, $\mathrm{p}(S_{\mathrm{k}-1}|Y_{\mathrm{k}-1})$ and the transitional prior, $\mathrm{p}(S_{\mathrm{k}}|S_{\mathrm{k}-1})$, via the Chapman-Kolmogorov equation (Ristic, et al., 2004).

(3.9) $$\boldsymbol{p(S_k|Y_{k-1}) = \int p(S_k|S_{k-1})\, p(S_{k-1}|Y_{k-1}) dS_{k-1}}$$

Once the new observation data are gathered, using Bayes rule the state predication is updated as it follows:

(3.10) $$\boldsymbol{p(S_k|Y_k) = \frac{p(Y_k|S_k)p(S_k|Y_{k-1})}{\int p(Y_k|S_k)p(S_k|Y_{k-1})dS_k}}$$

3.4.3 Recursive Filtering and Sensor Fusion

Recursive filtering is widely used in sensor fusion, where one or more sensors contribute to the formation of the transitional prior, likelihood functions or both. The manner in which this takes effect has been the subject of several studies in hybrid tracking methods, some of which were described in Chapter 2.

In order to understand how recursive filtering can be applied to sensor fusion, the main recursive filtering techniques; namely, Kalman Filter (KF), Extended Kalman Filter (EKF) and Particle Filter (PF) are briefly described in the following sections. A detailed description of these methods is outside the scope of this work, and the reader is invited to refer to (Ristic, et al., 2004) and (Arulampalam, et al., 2002) for a more comprehensive reviews.

3.4.3.1 Kalman Filtering

Kalman filtering only applies when $f(.)$ and $h(.)$ in the state-space equations (3.5) and (3.6) are linear, as presented in equations (3.11) and (3.12). Moreover the process and measurement noise (w and v) must have normal probability distributions, with zero mean values.

(3.11) $\boldsymbol{S_{k+1} = AS_k + BU_k + w_k}$

(3.12) $\boldsymbol{Y_k = HS_k + v_k}$

Kalman filtering is carried out in two stages; prediction and update. During prediction, past estimated state of the system ($\hat{\mathrm{S}}_{\mathrm{k-1}}$) is used to predict the current state vector, $\bar{S}_k$.

(3.13) $\boldsymbol{\overline{S}_k = A\widehat{S}_{k-1} + BU_{k-1}}$

In the update stage, the new observation data along with the prediction are used to provide an estimate for the current state of the system.

(3.14) $\boldsymbol{\widehat{S}_k = \overline{S}_k + K_k(Y_k - H\overline{S}_k)}$

K_k is the KF gain matrix, which is estimated at every step before the update stage is executed. In order to define K_k, the covariance matrices for the error in state prediction ($\bar{e}_k = \bar{S}_k - S_k$) and state estimation ($\hat{e}_k = \hat{S}_k - S_k$) are required.

$\overline{\wp}_k$ is the covariance matrix for the predication error, and is estimated using the covariance matrix for the past state estimation, $\wp_{\mathrm{k-1}}$, and process noise covariance matrix Q as follows.

(3.15) $\boldsymbol{\overline{\wp}_k = A\wp_{k-1}A^T + Q}$

Matrix $\mathrm{K_k}$ is estimated using prediction error covariance matrix, $\overline{\wp}_{\mathrm{k}}$, and measurement noise covariance matrix R.

(3.16) $\mathbf{K_k = \overline{\wp}_k H^T (H\overline{\wp}_k H^T + R)^{-1}}$

The state estimation error covariance matrix is then provided as follows:

(3.17) $\wp_{\mathbf{k}} = (\boldsymbol{I} - \boldsymbol{K_k H})\bar{\wp}_{\boldsymbol{k}}$

The Kalman filter offers a simple, yet robust and effective approach to state estimation, although it can only apply to systems with linear process and measurement models. However if a non-linear system can be linearised around the current state, a modified form of KF, called Extended Kalman Filter (EKF), can apply, which is explained in the following section.

3.4.3.2 Extended Kalman Filtering

Extended Kalman Filter (EKF) is applicable when $f(.)$ or $h(.)$ in the state-space equations (3.5) and (3.6) are non-linear but can be linearised around the current system state as follows:

(3.18) $\boldsymbol{S_k} \approx \boldsymbol{f}(\widehat{\boldsymbol{S}}_{\boldsymbol{k-1}}, \boldsymbol{U_{k-1}}, \mathbf{0}) + \boldsymbol{A}(\boldsymbol{S_{k-1}} - \widehat{\boldsymbol{S}}_{\boldsymbol{k-1}}) + \mathbf{W}\boldsymbol{w_{k-1}}$

(3.19) $\boldsymbol{Y_k} \approx \boldsymbol{h}(\widehat{\boldsymbol{S}}_{\boldsymbol{k}}, \mathbf{0}) + \boldsymbol{H}(\boldsymbol{S_k} - \boldsymbol{f}(\widehat{\boldsymbol{S}}_{\boldsymbol{k}}, \boldsymbol{U_k}, \mathbf{0})) + \boldsymbol{V}\boldsymbol{v_k}$

A, C, H and D are the Jacobian matrices and are defined as follows:

(3.20) $\mathbf{A}_{[\mathbf{i},\mathbf{j}]} = \frac{\partial \mathbf{f}_{[\mathbf{i}]}(\hat{\mathbf{S}}_{\mathbf{k-1}}, \mathbf{U_{k-1}}, \mathbf{0})}{\partial \mathbf{s}_{[\mathbf{j}]}}$, $\mathbf{W}_{[\mathbf{i},\mathbf{j}]} = \frac{\partial \mathbf{f}_{[\mathbf{i}]}(\hat{\mathbf{S}}_{\mathbf{k-1}}, \mathbf{U_{k-1}}, \mathbf{0})}{\partial \boldsymbol{\eta}_{[\mathbf{j}]}}$

(3.21) $\boldsymbol{H}_{[\boldsymbol{i},\boldsymbol{j}]} = \frac{\partial \boldsymbol{h}_{[\boldsymbol{i}]}(\widehat{\boldsymbol{S}}_{\boldsymbol{k}}, \mathbf{0})}{\partial \boldsymbol{s}_{[\boldsymbol{j}]}}$, $\boldsymbol{V}_{[\boldsymbol{i},\boldsymbol{j}]} = \frac{\partial \boldsymbol{h}_{[\boldsymbol{i}]}(\widehat{\boldsymbol{S}}_{\boldsymbol{k}}, \mathbf{0})}{\partial \boldsymbol{\xi}_{[\boldsymbol{j}]}}$

And the indices i and j vary from 1 to the total number of states in the state vector. Similar to KF, the EKF has a predict stage defined as it follows:

(3.22) $\bar{\boldsymbol{S}}_{\boldsymbol{k}} = \boldsymbol{f}(\widehat{\boldsymbol{S}}_{\boldsymbol{k-1}}, \boldsymbol{U_{k-1}}, \mathbf{0})$

(3.23) $\bar{\wp}_{\boldsymbol{k}} = \boldsymbol{A}\wp_{\mathbf{k-1}}\boldsymbol{A^T} + \boldsymbol{W_k Q_{k-1} W_k^T}$

The update stage follows the following formulas:

(3.24) $\mathbf{K_k} = \bar{\wp}_{\mathbf{k}}\mathbf{H^T}(\mathbf{H}\bar{\wp}_{\mathbf{k}}\mathbf{H^T} + \boldsymbol{V_k R V_k^T})^{-1}$

(3.25) $\widehat{\boldsymbol{S}}_{\boldsymbol{k}} = \overline{\boldsymbol{S}}_{\boldsymbol{k}} + \boldsymbol{K}_{\boldsymbol{k}}\left(\boldsymbol{Y}_{\boldsymbol{k}} - \boldsymbol{h}\left(\widehat{\boldsymbol{S}}_{\boldsymbol{k}}, \mathbf{0}\right)\right)$

(3.26) $\wp_{\mathbf{k}} = (\boldsymbol{I} - \boldsymbol{K}_{\boldsymbol{k}}\boldsymbol{H})\overline{\wp}_{\boldsymbol{k}}$

As described earlier, EKF has applications in Visual SLAM and Hybrid Visual-Inertial Camera Tracking. (Jeroen D. Ho, et al., 2007) presented a real-time tracking approach by fusing measurements from inertial and vision sensors applying EKF filtering. In his work he compared the advantages of EKF with KF approach.

3.4.3.3 Particle Filtering

When $f(.)$ and $h(.)$ in the state-space equations (3.5) or (3.6) are non-linear, particle filtering is a suitable method for filtering. This method provides an estimate for the posterior pdf, $p(S_k|Y_k)$, using the weighted summation of a number of particles, each representing a probable state of the system.

(3.27) $\mathbf{p}(\mathbf{S_k}|\mathbf{Y_{k-1}}) \approx \sum \mathbf{w}_{\mathbf{i,k}}^{\mathbf{p}} \boldsymbol{\delta}(\mathbf{S_k} - \mathbf{S}_{\mathbf{i,k}}^{\boldsymbol{p}})$

Particles, $S_{i,k}^{p}$, are selected from a proposal distribution, referred to as the importance pdf, $q(\mathrm{S_k}|S_{\mathrm{k-1}}, \mathrm{Y_k})$. The particles are then evaluated to find the associated weight. The weighted particles are used to provide an estimate for the posterior pdf. This process can be summarised as it follows:

A. Draw a new particle from the proposal distribution, $q(S_k|S_{i,k-1}^{p}, Y_k)$.

B. Assign a weight to each particle using the past weight, the likelihood pdf, the transitional prior and the importance pdf.

$$\widehat{\boldsymbol{w}}_{\mathbf{i,k}} \propto \widehat{\boldsymbol{w}}_{\mathbf{i,k-1}} \frac{\mathbf{p}(\mathbf{Y_k}|\mathbf{S}_{\mathbf{i,k}}^{\mathbf{p}})\mathbf{p}(\mathbf{S}_{\mathbf{i,k}}^{\mathbf{p}}|\mathbf{S}_{\mathbf{i,k-1}}^{\mathbf{p}})}{\mathbf{q}(\mathbf{S}_{\mathbf{i,k}}^{\mathbf{p}}|\mathbf{S}_{\mathbf{i,k-1}}^{\mathbf{p}}, \mathbf{Y_k})}$$

C. Normalise the weights to have the total value of 1.

$$\boldsymbol{w}_{\mathbf{i,k}} \propto \frac{\widehat{\boldsymbol{w}}_{\mathbf{i,k}}}{\sum \widehat{\boldsymbol{w}}_{\mathbf{i,k}}}$$

One of the problems associated with particle filtering is the degeneracy phenomenon, which means that after a certain number of recursive steps, all but one particle will have negligible normalised weight. A suitable measure for degeneracy is the effective sample size, which is defined as $N_{eff} = 1/\sum \mathcal{W}_{i,k}^2$. If the number of effective particles is less than an application-specific threshold, a resampling process needs to be applied. Resampling eliminates samples with low weight and multiplies samples with high weight. For more details please refer to (Ristic, et al., 2004) and (Arulampalam, et al., 2002). The choice of importance function and the mechanisms for particle selection and resampling are application-specific. These methods have been covered in more detail in Chapter 6, as one of the contributions of this work.

3.4.4 Applications of Recursive Filtering in Inertial-Visual Tracking

As described in Chapter 2, there are several hybrid tracking systems based on recursive filtering methods such as extended Kalman filter and particle filtering. Such methods use both accelerometer and gyroscope (often incorporated into a single IMU) data, and by taking into account the kinematic motion model of the moving camera, together with the characteristics of the IMU, form the state space equations. Here the hybrid tracking system, proposed by Weiss (Weiss, 2012) for his PhD work on “Vision Based Navigation for Micro Helicopters” is presented as a bench mark. This work, to the author’s knowledge is the latest, most advanced system proposed for hybrid-tracking, which outperforms its predecessors and has been successfully applied to the EU-funded SFly (Swarm of Micro Flying Robots) project (Scaramuzza, et al., 2014). This work is referred to as the Weiss Vision Based Navigation system, or simply Weiss VBN in this section.

The Weiss VBN system (Weiss, 2012) consists of an IMU, a vision sensor and an EKF filter for hybrid pose estimation. The state vector in the Weiss system consists of position, velocity, orientation, accelerometer bias, gyroscope bias,

the visual scaling factor, distance between the IMU and camera frames and finally the rotation between these two frames. The state transition model is a non-linear one, which is linearised so that the current state can be predicted using the EKF update equations. The measurement model, on the other hand, is formed using the vision-based data and associated feature points. The algorithm requires the measurement covariance matrix to be found using the vision-based methods presented in (Beder & Steffen, 2006) and (Eudes & Lhuillier, 2009). The system uses key-frames for vision-based pose estimation, similar to the PTAM algorithm (Klein & Murray, 2007). However the number of frames is capped in order to reduce the computational cost of the algorithm. The core principle of the system is based on fusing IMU data and map-based vision data using an EKF algorithm as mentioned above. Figure 3.2 shows the block diagram of the system.

In addition to the core algorithm, Weiss proposes additional sensors, such as magnetometer and GPS, to correct the drift in the system. It also provides a method for estimating the scaled velocity using optical flow, as an additional mechanism for pose correction.

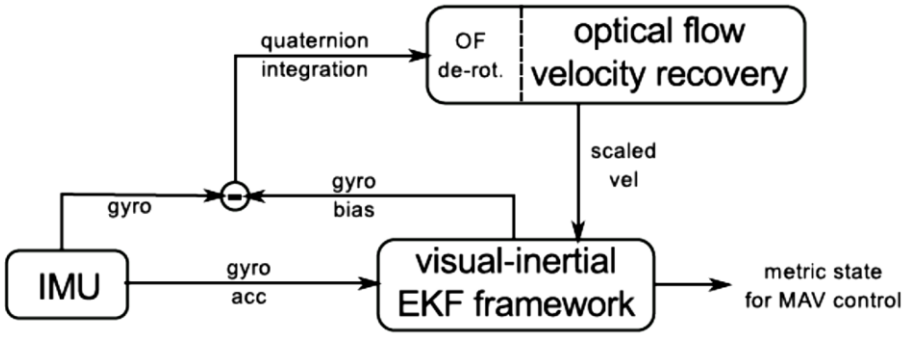


Figure 3.2 Weiss's VBN System Block Diagram - Source: (Beder & Steffen, 2006)

3.4.5 Hybrid Tracking Based on Particle Filtering and Focus of Expansion

The Weiss VBN system (Weiss, 2012) is an effective hybrid tracking system; however it has a relatively high computational cost due to the use of a state vector with a high number of states (28 elements) and also a collection of key frames. The use of high-dimension matrices for the application of EKF filtering is also prone to numerical error and inefficiencies. This method employs visual pose estimation based on PTAM which, although has an improved performance compared with the standard Visual SLAM methods, still requires a considerable computational effort for handling several key-frames. The estimation of the measurement error covariance matrix also needs a vision-based method, which adds further to the computational complexity. The proposed solution presented in this thesis addresses the issues raised above and provides a robust and accurate yet effective solution for hybrid camera tracking.

3.5 Summary and Conclusion

This chapter provided a review of tracking methods, which use computer vision (image)-based tracking algorithms as the sole means of tracking or as part of a multisensory approach. There have been numerous proposals and solutions for pose recovery in the past few decades. Among these, Visual SLAM, Visual Odometry and PTAM based solutions provide reasonable accuracy especially for the mobile Augmented Reality applications. However these methods have limitations with regard to use in wide-area tracking measurements and uncontrolled real-time localisation due to their expensive computational costs and high complexity involved.

In recent years, hybrid systems consisting of low cost inertial measurement units and robust and high-dimensional computer vision-aided algorithms have

enhanced the performance and agility of the tracking systems. Such solutions have also helped towards tackling the hurdles of real-time sensing and localisation especially in GPS-denied environments.

As a result of this study, in Chapter 4, the author proposed and developed a multisensory system based on Visual SLAM-GPS sensor fusion to examine the potential for a hybrid system (Visual SLAM) to enhance the accuracy of GPS.

During the course of this study it has become evident that vision-based tracking systems alone are not an adequate solution to wide-area pose tracking and localisation and there is a need for design and implementation of sensor-aided systems to compensate for the shortcomings of the existing algorithms. Consequently, recursive filtering approaches for sensor fusion were also reviewed as a tool for the development of an improved approach for pose recovery of a moving camera leading to the inertial-visual pose tracking system using optical flow-aided particle filtering presented in Chapters 6 and 7.

CHAPTER FOUR

4 Tracking Using RF-Based Positioning Systems

Radio Frequency (RF) signals can also be used as a means of pose estimation - primarily location rather than orientation. Such methods operate either on the basis of signal travel time or signal strength. The Global Positioning System (GPS) is one example of a system based on signal travel time. GPS satellites transmit signals at a known time and position. The time of the received signal in conjunction with the time of signal transmission, and also the position of the satellite at the time of transmission, are used to determine location information for the GPS receiver.

On the other hand, ©Apple's recently introduced iBeacons are intended for micro-location estimation where the Received Signal Strength (RSS) is used as measurement criteria for position determination. Although iBeacon is currently envisaged mainly for proximity detection in the retail industry, Apple© argues that iBeacons also have utility for micro-location tracking. In this scenario a mobile device with its Bluetooth receiver appropriately configured uses the strength of signal received from different iBeacons to triangulate receiver position by a process of trilateration.

In order to evaluate the performance of such systems, a combined GPS and vision-based pose tracking system (Visual SLAM-GPS sensor fusion), and then a system based on iBeacon technology, were implemented. This chapter covers the outcome of these two systems followed by performance analysis and conclusions based on these experiments.

4.1 GPS-Visual SLAM Hybrid Tracking System

One of the most ubiquitous means of wide-area location tracking is the Global Positioning System (GPS). However, high-accuracy, wide-area tracking cannot be achieved using GPS alone. Moreover in GPS-denied areas, where the direct lines of sight to the satellites are not maintained, GPS on its own is not a technique that can be always relied upon. On the other hand, Visual-SLAM solutions (as described in Chapter 3), which rely on the derivation of landmarks from detected feature points and their continued robust association, becomes increasingly difficult as the mapped area widens. Wide-area tracking also requires appropriate strategies for dealing with long-term management of features as the map size grows.

Although some work has been done on integrating GPS with odometry and inertia data ((Schall, et al., 2009) ; (Berrabah, et al., 2011) and (Schleicher, et al., 2009)), very little attention has been paid to utilising image data from the monocular camera typically found in mobile consumer devices combined with the GPS location information usually available on these platforms.

Presented here is a strategy for utilising visual-SLAM to substantially improve the output accuracy of GPS on mobile devices.

Figure 4.1 is a system diagram to illustrate the GPS_Visual SLAM algorithm.

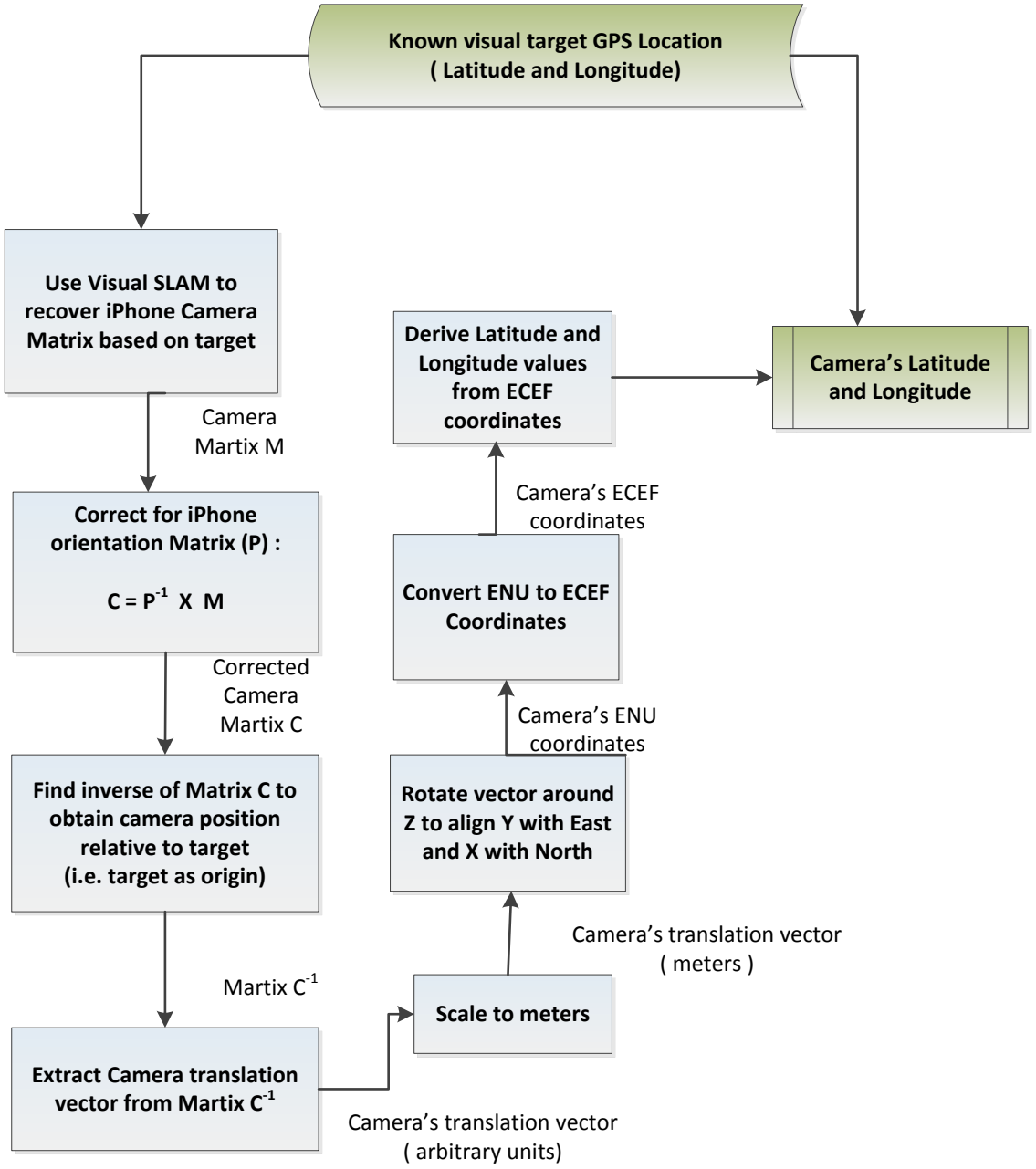


Figure 4.1 GPS_Visual SLAM System Overview

4.1.1 Converting the Geodetic to ENU Coordinates

GPS provides geodetic data in the form of latitude, longitude and altitude. However, for tracking purposes, it is more practical to use an orthogonal reference frame such as provided by East, North, Up (ENU) Cartesian

coordinates. This coordinate system forms a plane tangent to the Earth's surface at any location desired to provide a local coordinate system in that region of the Earth's surface.

The Earth-Centred Earth-Fixed (ECEF) coordinate system is by convention labelled x, y, z and has its origin at the centre of the Earth with its x-axis passing through the equator at the prime meridian and its z-axis passing through the North Pole. The y-axis, determined by the right-hand rule, passes through the equator at 90 degrees longitude as illustrated in Figure 4.2.

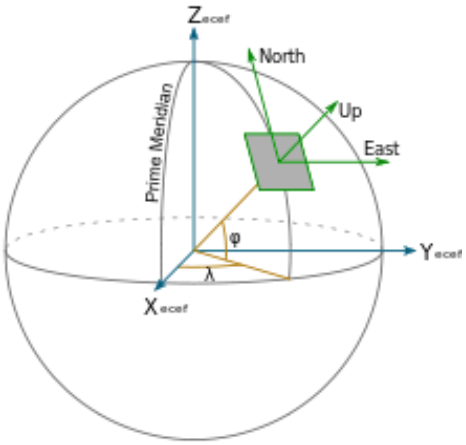


Figure 4.2 : East, North, Up (ENU) Cartesian coordinates

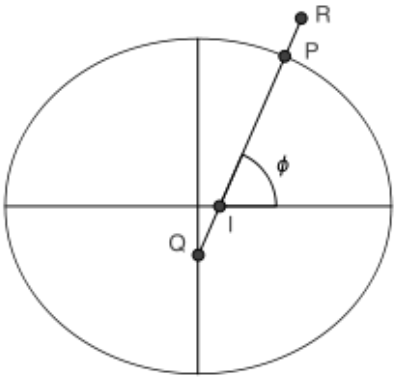


Figure 4.3 : Geodetic-ECEF Coordinate

In order to convert the Geodetic coordinate systems (latitude (ϕ), longitude (λ), height (h)) to the local ENU (East, North, and Up), two main stages need to be

followed : firstly to convert Geodetic to ECEF coordinates, and second to convert ECEF coordinates to local ENU coordinates.

Geodetic coordinates $\rightarrow$ ECEF coordinates $\rightarrow$ ENU coordinates

Similarly, conversion from ENU to Geodetic coordinates requires:

ENU coordinates $\longrightarrow$ ECEF coordinates $\longrightarrow$ Geodetic coordinates

The above coordinate conversions were implemented and the code can be found in Appendix B.

4.1.2 Converting Geodetic to ECEF Coordinates

The following needs to be performed for the conversion of Geodetic to ECEF coordinates (see Figures 4.2 and 4.3). The World geodetic System 1984 (WGS84) standard is applied with respect to these conversions. Equations (4.1) to (4.8) are derived for (Farrell, 1999). The length PQ is called the *Normal* and is the distance from the surface to the Z-axis along the ellipsoid normal and can be determined as in equation (4.2). The Geodetic to ECEF coordinate conversions are implemented based on the following formulations and can be found in Appendix B.

(4.1) $\mathbf{PQ} = \mathbf{N}\,(\boldsymbol{\phi})$

(4.2) $\mathbf{IQ} = \mathbf{e^2} \times \mathbf{N}\,(\boldsymbol{\phi}).\mathbf{R}(\mathbf{X},\mathbf{Y},\mathbf{Z})$

Geodetic coordinates (latitude ϕ, longitude λ, height h) can be converted into ECEF coordinates using the following equations:

(4.3) $\mathbf{X} = (\mathbf{N}\,(\boldsymbol{\phi}) + \mathbf{h})\,\mathbf{cos}\,\boldsymbol{\phi}\;\mathbf{cos}\,\boldsymbol{\lambda}$

(4.4) $\mathbf{Y} = (\mathbf{N}\,(\boldsymbol{\phi}) + \mathbf{h})\,\mathbf{cos}\,\boldsymbol{\phi}\;\mathbf{sin}\,\boldsymbol{\lambda}$

(4.5) $\mathbf{Z} = (\mathbf{N}\,(\boldsymbol{\phi})(\mathbf{1} - \mathbf{e^2}) + \mathbf{h})\,\mathbf{sin}\,\boldsymbol{\phi}$

where:

(4.6) $$\mathbf{N}(\boldsymbol{\phi}) = \frac{\mathbf{a}}{\sqrt{1-\mathbf{e}^2\ \sin^2\boldsymbol{\phi}}} \quad : \quad \boldsymbol{e} = \frac{\boldsymbol{f}}{\boldsymbol{a}} \ : \ \mathbf{0} < \boldsymbol{e} < \mathbf{1}$$

(a, b and e are the semi-major, semi-minor axis and the first numerical eccentricity of the ellipsoid respectively)

Furthermore, the following equations hold:

(4.7) $$\frac{\mathbf{p}}{\cos\boldsymbol{\phi}} - \frac{\mathbf{z}}{\sin\boldsymbol{\phi}} - \mathbf{e}^2\,\mathbf{N}(\boldsymbol{\phi}) = \mathbf{0}$$

(4.8) $$\mathbf{p} = \sqrt{\mathbf{X}^2 + \mathbf{Y}^2}$$

4.1.3 Converting the ECEF to Geodetic Coordinates

The conversion of ECEF to Geodetic coordinates can be solved efficiently using the Newton–Raphson iteration method (see (Bowring, 1985) and (Misra & Enge, 2001) for the detail).

(4.9) $$\boldsymbol{k} - \mathbf{1} - \frac{\mathbf{e}^2\,\mathbf{ak}}{\sqrt{\mathbf{p}^2+(\mathbf{1}+\mathbf{e}^2)\mathbf{z}^2\mathbf{k}^2}} = \mathbf{0}$$

(4.10) $$\boldsymbol{k} = \frac{\boldsymbol{p}}{\boldsymbol{z}}\,\boldsymbol{tan}\,\boldsymbol{\phi}$$

The height is calculated as follows:

(4.11) $$\mathbf{h} = \mathbf{e}^{-2}\ \left(\mathbf{k}^{-1} - \mathbf{k_0}^{-1}\right)\sqrt{\mathbf{p}^2 + \mathbf{z}^2\,\mathbf{k}^2}$$

(4.12) $$\mathbf{k_0} = (\mathbf{1} - \mathbf{e}^2)^{-1}$$

The iteration can be transformed into the following calculation:

(4.13) $$\mathbf{k_{i+1}} = \frac{\mathbf{c_i} + (\mathbf{1}-\mathbf{e}^2)\,\mathbf{z}^2\,\mathbf{k_i}^3}{\mathbf{c_i} - \mathbf{p}^2} = \mathbf{1} + \frac{\mathbf{p}^2+(\mathbf{1}-\mathbf{e}^2)\mathbf{z}^2\ \mathbf{k_i}^3}{\mathbf{c_i} - \mathbf{p}^2}$$

(4.14) $$\mathbf{c_i} = \frac{(\mathbf{p}^2+(\mathbf{1}-\mathbf{e}^2)\mathbf{z}^2\ \mathbf{k_i}^3)^{3/2}}{\mathbf{ae}^2}$$

k_0 is a good starter for the iteration when $h \approxeq 0$. Bowring (Bowring, 1985) showed that the single iteration produces a sufficiently accurate solution. He used extra trigonometric functions in his original formulation.

4.1.3.1 Converting from ECEF to ENU Coordinates

To transform from ECEF coordinates to the local coordinates, a local reference point is needed. If the reference point is located at $\{X_r, Y_r, Y_r\}$ and an object at $\{X_p, Y_p, Y_p\}$ then the vector pointing from the reference to the object in the ENU frame is:

$$(4.15)\quad \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{pmatrix} = \begin{pmatrix} -\sin\boldsymbol{\lambda}_\mathbf{r} & \cos\boldsymbol{\lambda}_\mathbf{r} & \mathbf{0} \\ -\sin\boldsymbol{\phi}_\mathbf{r}\cos\boldsymbol{\lambda}_\mathbf{r} & -\sin\boldsymbol{\phi}_\mathbf{r}\sin\boldsymbol{\lambda}_\mathbf{r} & \cos\boldsymbol{\phi}_\mathbf{r} \\ \cos\boldsymbol{\phi}_\mathbf{r}\cos\boldsymbol{\lambda}_\mathbf{r} & \cos\boldsymbol{\phi}_\mathbf{r}\sin\boldsymbol{\lambda}_\mathbf{r} & \sin\boldsymbol{\phi}_\mathbf{r} \end{pmatrix} \begin{pmatrix} \mathbf{X_p} - \mathbf{X_r} \\ \mathbf{Y_p} - \mathbf{Y_r} \\ \mathbf{Z_p} - \mathbf{Z_r} \end{pmatrix}$$

where ϕ is the geodetic latitude. More details on this conversion can be found in (Farrell, 1999) (see Appendix B for the details on implementation).

4.1.3.2 Converting from ENU to ECEF

This is just the inversion of the ECEF to ENU transformation so:

$$(4.16)\quad \begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \end{pmatrix} = \begin{pmatrix} -\sin\boldsymbol{\lambda} & -\sin\boldsymbol{\phi}\cos\boldsymbol{\lambda} & \mathbf{con}\,\boldsymbol{\phi}\cos\boldsymbol{\lambda} \\ \cos\boldsymbol{\lambda} & -\sin\boldsymbol{\phi}\sin\boldsymbol{\lambda} & \mathbf{con}\,\boldsymbol{\phi}\sin\boldsymbol{\lambda} \\ \mathbf{0} & \cos\boldsymbol{\phi} & \sin\boldsymbol{\phi} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{pmatrix} + \begin{pmatrix} \mathbf{X_r} \\ \mathbf{Y_r} \\ \mathbf{Z_r} \end{pmatrix}$$

4.1.4 Refinement of GPS with Visual-SLAM

This section describes the implementation of a GPS-Visual SLAM tracking system and the experimental results achieved by this system. The system was implemented as an iOS mobile app using Objective C. The Visual SLAM for iPhone implementation was obtained from the PointCloud SDK (PointCloud, 2014) and an Objective C wrapper was developed for this SDK library.

4.1.4.1 Principles of Operation:

The following assumptions were made for the operation of the Visual SLAM-GPS tracking system:

- A reference image was captured at a known GPS location and with known orientation. The centre of this location forms the initial coordinate system origin and its axis directions; x tangent to the reference point parallel to the surface in the image; y normal to the reference image and z orthogonal to x and y as seen in Figure 4.4.

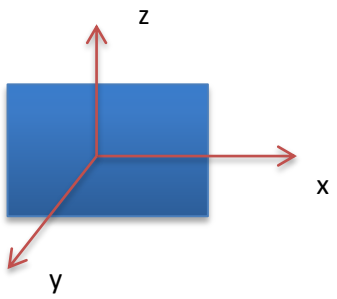


Figure 4.4 : Coordinates Assumption

- Visual-SLAM was used to recover the camera location and orientation in relation to this image in the form of a viewing transformation matrix.
- The actual deduced camera location was calculated from the inverse of this matrix.
- The camera location relative to the reference location was scaled to metres based on a calibration measurement.
- The camera location was rotated so that x, y, z aligned with E, N, U coordinates.
- The geodetic (GPS) location of the camera was then calculated by comparison to the reference GPS location.

4.1.4.2 Method of Operation

An initial image was tracked as the reference point. This image was taken from a section of the exterior of Gateway House, De Montfort University (see Figure 4.5) and the “Dry riser” as the reference point). Uploading and conversion of

this image via the pointcloud.io web server was used to obtain a corresponding point cloud feature model. An Objective C program was developed to use the iPhone camera in order to optically track from this feature model and use it to recover the camera matrix relative to the centre of the “Dry riser” image.

Figure 4.5 : 'Dry riser' on Gateway House. De Montfort University

The actual GPS location and the orientation of the wall were found using Google maps and a compass. Three camera locations were selected and metre distances measured relative to the ‘Dry riser’ as origin. One camera location was selected in order to calibrate the scale between the real world and that of the imaging system. The camera was placed at each of the three selected locations, orientated so as to be looking back at the Dry riser and the camera matrix was recorded in each case. Finally the camera was moved gradually between one location and another so that it ended at a known location but not orientated so that the ‘Dry riser’ was anywhere within view. Again, the camera matrix was recorded.

In each of the above camera locations, a GPS reading was taken directly from the GPS receiver in the iPhone and the GPS location was also determined from

the iPhone camera matrices. The measured and calculated GPS were then compared. The next section illustrates the results.

4.1.4.3 Experimental Results

The actual GPS location of the 'Dry riser' wall location was obtained from Google maps, where the Latitude and Longitude were found to be at 52.629527 and -1.138096, respectively. The Altitude was not measured in the testing but was kept constant throughout. The orientation of the wall was also measured using a compass and found to be facing 285 degrees to the NW. Figure 4.6 shows a planar view of the wall with the 'Dry riser' location shown along with the three camera locations including, in each case, the direction of camera view. Note that locations C and D are the same although in the latter the camera is facing the wall orthogonally and can no longer 'see' the 'Dry riser'.

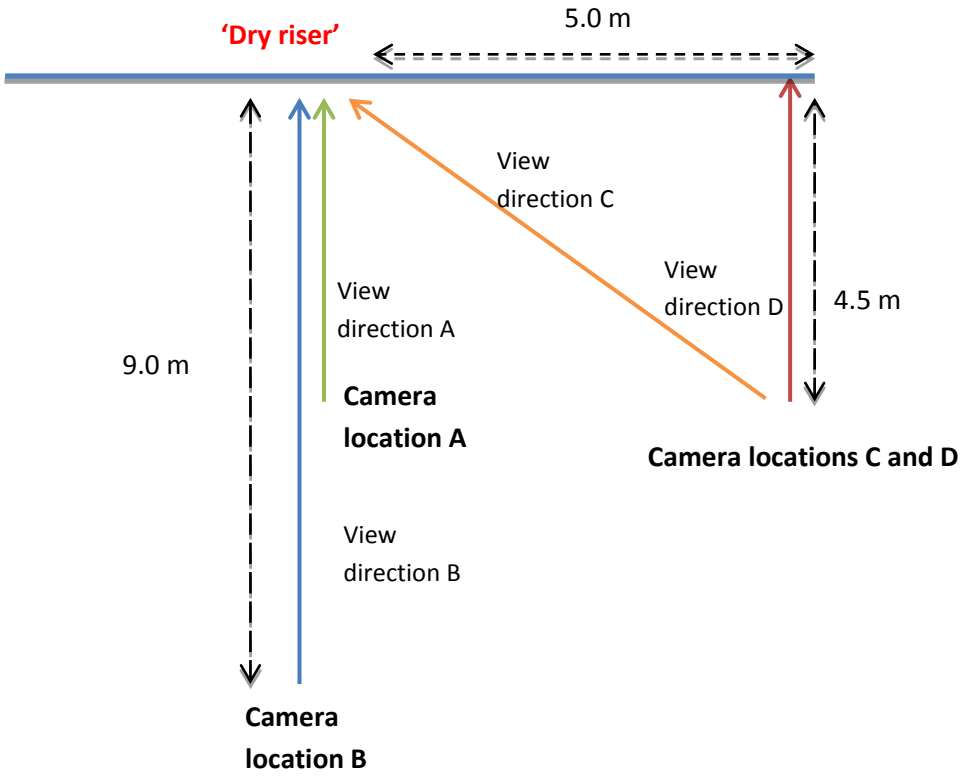


Figure 4.6 : Camera locations relative to wall and 'Dry riser'

Table 4.1 to 4.3 summarise the iPhone camera distances from the DryRiser target and the errors in distance estimation using GPS_Visual SLAM. The details of each set of experiment and the measurements are fully described and illustrated in following sections with the snapshots of measured values of latitude and longitude.

Locations	Actual Distance from the Wall		
	x	y	Distance (m)
C	5	4.5	6.727
A	0	4.5	4.5
B	0	9	9
D	5	4.5	6.727

Table 4.1 : Actual Distances from the Wall

Locations	GPS_Visual SLAM Distance			
	x	y	Distance (m)	Error (m)
C	4.99	4.43	6.673	0.054
A	- 0.478	3.629	3.66	0.84
B	0.037	8.141	8.141	0.859
D	4.466	4.919	6.644	0.083

Table 4.2 : GPS Visual_SLAM Distances

Locations	GPS_Visual SLAM Distance with amended scale factor (scaling factor = 2.72)			
	x	y	Distance (m)	Error (m)
C	5.026	4.463	6.722	0.005
A	- 0.481	3.656	3.687	0.813
B	0.0372	8.201	8.201	0.799
D	4.499	4.955	6.693	0.034

Table 4.3 : GPS Visual_SLAM Distances with scale factor

4.1.4.4 Tracking Location C

Camera location C was used to calibrate the scale factor, which relates the real-world metre measurements to the arbitrary units used in the camera matrix. At location C the SLAM software gave the column-order matrix as in equation (4.17) - column-order camera matrix (R,T).

(4.17)

$$\mathbf{M} = \begin{pmatrix} \mathbf{R_{11}} & \mathbf{R_{12}} & \mathbf{R_{13}} & \mathbf{T_1} \\ \mathbf{R_{21}} & \mathbf{R_{22}} & \mathbf{R_{23}} & \mathbf{T_2} \\ \mathbf{R_{31}} & \mathbf{R_{32}} & \mathbf{R_{33}} & \mathbf{T_3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix} \triangleq \begin{pmatrix} \mathbf{0.094} & \mathbf{0.109} & \mathbf{0.990} & \mathbf{-0.275} \\ \mathbf{-0.531} & \mathbf{0.846} & \mathbf{-0.043} & \mathbf{-0.408} \\ \mathbf{-0.842} & \mathbf{-0.5222} & \mathbf{0.138} & \mathbf{2.423} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

The iPhone coordinate system, itself, introduces a transformation. The iPhone R&T matrix (P) is :

(4.18) $$\mathrm{P} = \begin{pmatrix} 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

This needs to be undone in order to get the real Camera Matrix so the raw matrix result needs to be multiplied by the Inverse of the iPhone matrix P (P^{-1}):

(4.19) $$\mathrm{P}^{-1} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Therefore the real Camera Matrix is given by:

(4.20) $$\mathrm{C} = P^{-1} \times M$$

Given the matrix C, as described above the actual camera matrix considering the iPhone matrix, in this case, will be:

$$(4.21) \quad \mathbf{C} = \begin{pmatrix} \mathbf{0.531} & \mathbf{-0.846} & \mathbf{0.043} & \mathbf{0.408} \\ \mathbf{0.842} & \mathbf{0.522} & \mathbf{-0.138} & \mathbf{-2.423} \\ \mathbf{0.109} & \mathbf{0.343} & \mathbf{0.990} & \mathbf{-0.275} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

The camera matrix represents the transformation from the 'Dry riser' target at the World Origin to the camera as the origin. To obtain the Camera position in the world relative to the world origin it is necessary to find the camera's Inverse of matrix C^{-1} as in equation (4.22).

$$(4.22) \quad \mathbf{C}^{-1} = \begin{pmatrix} \mathbf{0.532} & \mathbf{0.842} & \mathbf{0.094} & \mathbf{1.849} \\ \mathbf{0.043} & \mathbf{0.521} & \mathbf{0.109} & \mathbf{1.639} \\ \mathbf{0.004} & \mathbf{-0.137} & \mathbf{0.989} & \mathbf{-0.078} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

The last column of this matrix gives the translation vector e.g. x, y, z coordinates of the camera. Therefore the final location will be at $x = 1.849$, $y = 1.639$ and $z = -0.078$. However, the physically measured y-distance to the camera was at $x = 5.0m$ and $y = 4.5m$. x = 5.0. Therefore a scaling factor of approximately 2.7 was needed to convert x, y, z to metre, thus $X' = 4.99m$ and $Y' = 4.43m$. Figure 4.7 shows the x, y orientation of the 'Dry riser' wall relative to North and East.

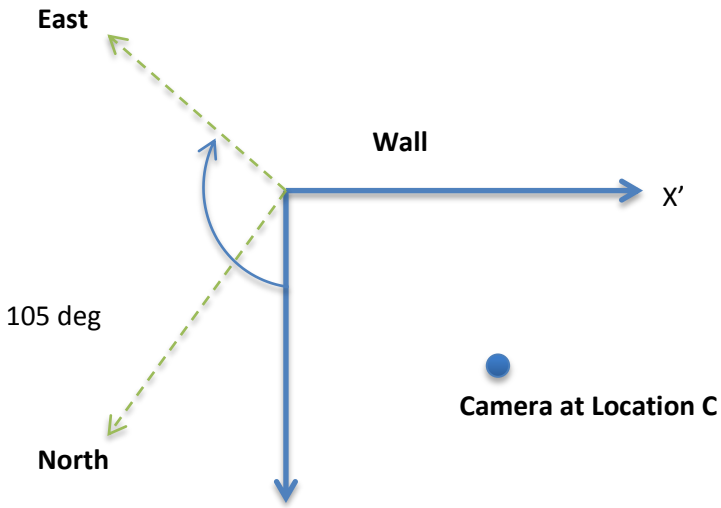


Figure 4.7 : Camera Location C

The X', Y' axes need to be rotated clockwise by 105 degrees to make y align with East and x align with North. Thus it is necessary to rotate the camera around the origin anti-clockwise by 105 degrees. In order to achieve this, the following rotation is required:

$$\text{(4.23)} \quad \mathbf{R} = \begin{pmatrix} \mathbf{\cos 105} & \mathbf{\sin 105} & \mathbf{0} & \mathbf{0} \\ \mathbf{-\sin 105} & \mathbf{\cos 105} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix} = \begin{pmatrix} \mathbf{-0.26} & \mathbf{0.97} & \mathbf{0} & \mathbf{0} \\ \mathbf{-0.97} & \mathbf{-0.26} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

Transforming the camera from X', Y', Z' to N, E, U coordinates gives:

$$\text{(4.24)} \quad N = 2.982 \quad E = -5.968 \quad U = -0.211$$

Converting this to ECEF coordinates and then to geodetic coordinates, relative to the 'Dry riser' location gives:

$$\text{(4.25)} \quad Latitude = 52.629554\,; \; Longitude = -1.138184$$

Comparing these results with the actual 'Dry riser' GPS location with GPS-only tracking system is depicted in the figures below. Figures 4.8 to 4.11 show how the original GPS-only tracking results was improved combining that with the visual SLAM tracking data.

Figure 4.8 : Dry Riser Location: Lat: 52.629527, Long: -1.138096

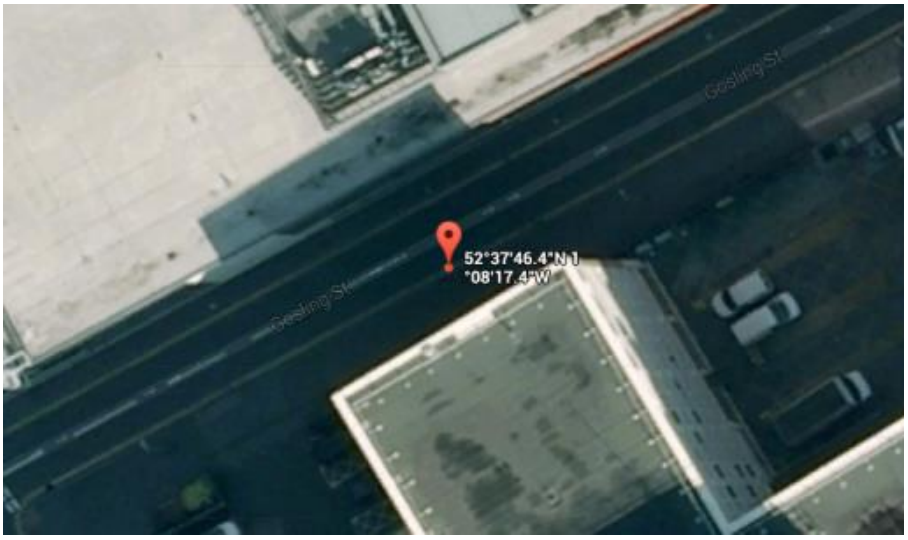


Figure 4.9 : Actual Location at C: Lat: 52.629556, Long: -1.138167

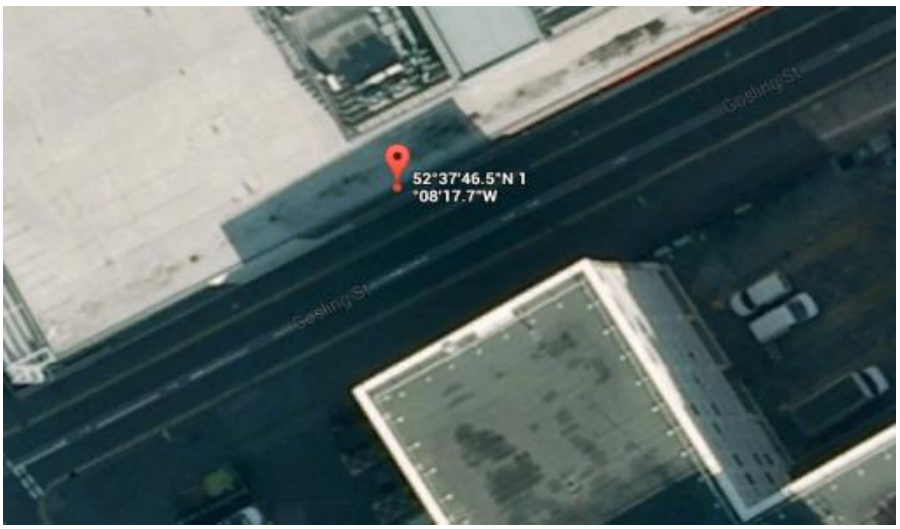


Figure 4.10 : iPhone GPS reading at C: Lat: 52.629583, Long: -1.138250

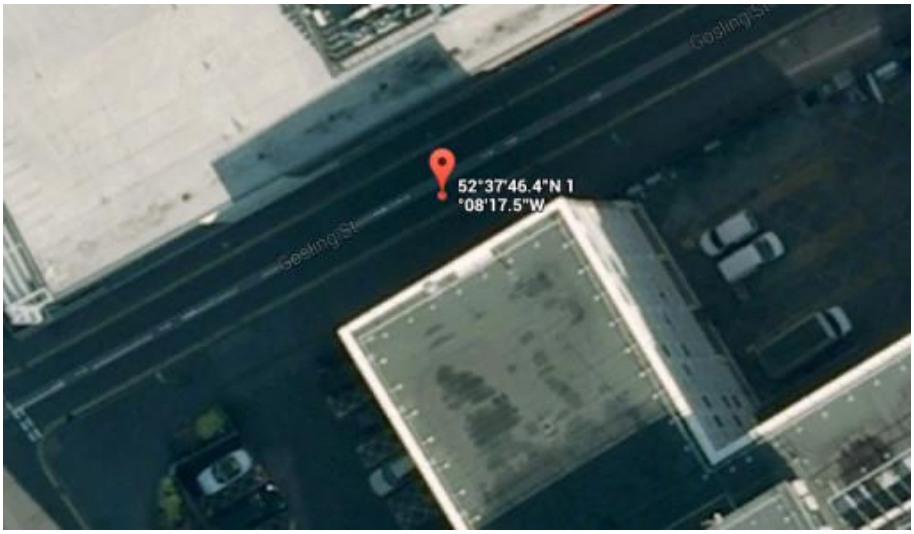


Figure 4.11 : GPS_SLAM measure at C: Lat: 52.629554, Long: -1.138184

4.1.4.5 Tracking Location A

The next test was performed for camera location A; 4.5 metres directly in front of the 'Dry riser'. The raw camera matrix was recorded as:

(4.26)
$$\begin{pmatrix} -0.051 & 0.234 & 0.971 & -0.159 \\ -0.992 & -0.125 & -0.022 & -0.011 \\ 0.116 & -0.964 & 0.239 & 1.357 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

After allowing for the iPhone transformation, the camera matrix, C, was found to be:

(4.27) $$\mathbf{C} = \begin{pmatrix} \mathbf{0.992} & \mathbf{0.125} & \mathbf{0.022} & \mathbf{0.011} \\ \mathbf{-0.116} & \mathbf{0.964} & \mathbf{-0.239} & \mathbf{-1.357} \\ \mathbf{-0.051} & \mathbf{0.234} & \mathbf{0.971} & \mathbf{-0.159} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

The inverse of C is:

(4.28) $$\mathbf{C}^{-1} = \begin{pmatrix} \mathbf{0.992} & \mathbf{-0.116} & \mathbf{-0.051} & \mathbf{-0.117} \\ \mathbf{0.125} & \mathbf{0.964} & \mathbf{0.235} & \mathbf{1.344} \\ \mathbf{0.022} & \mathbf{-0.238} & \mathbf{0.971} & \mathbf{-0.170} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

This gave x, y, z camera location as;

(4.29) $x = -0.11\ ;\ y = 1.344\ ;\ z = -0.170$

Applying 2.7 scaling factor (as calculated through the calibration process in location C) resulted in:

(4.30) $X' = -0.478m.\ ;\ Y' = 3.629\ m.\ ;\ Z' = -0.459m$

After rotation to E, N, U:

(4.31) $E = -0.478m\ ;\ N = 3.629m\ ;\ U = -0.459m$

Which gave geodetic coordinates as below:

(4.32) $Latitude: 52.629560\ ;\ Longitude: -1.138103$

Figures 4.12 to 4.14 show how the original GPS-only tracking results were improved combining that with the visual SLAM tracking data at Location A.

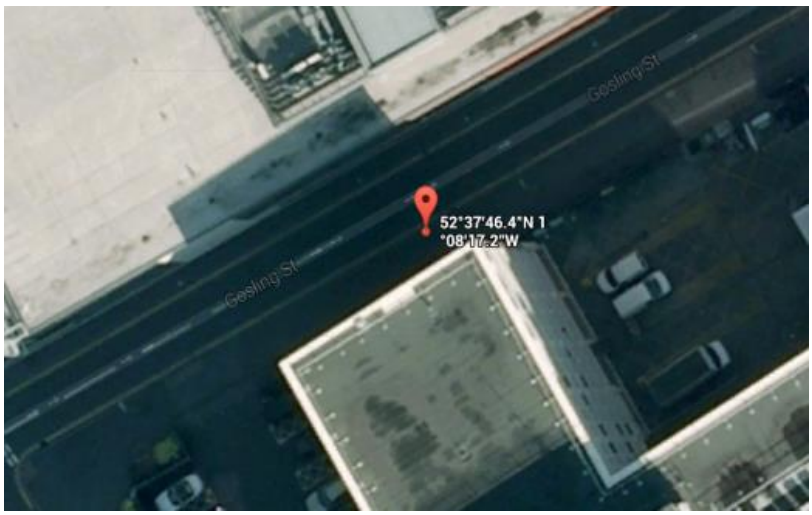


Figure 4.12 : Actual Location at A: Lat: 52.629560, Long: -1.138111

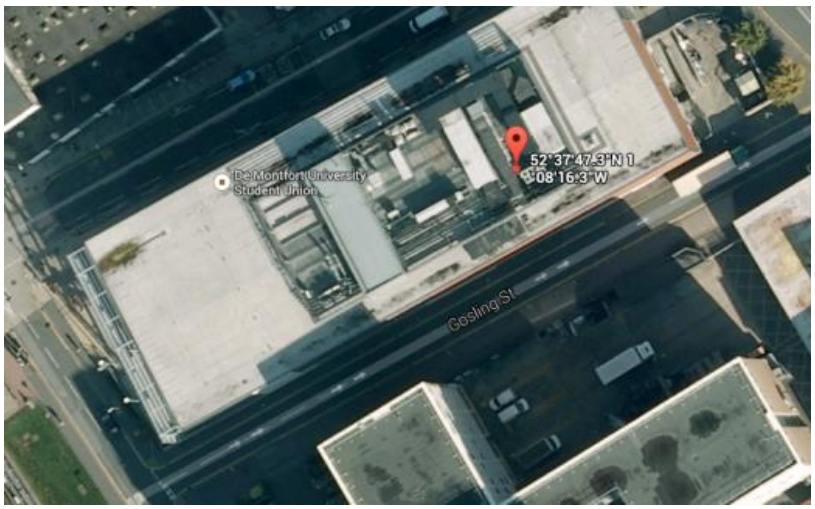


Figure 4.13 : iPhone GPS reading at A: Lat: 52.629806, Long: -1.137861

Figure 4.14 : GPS_SLAM measure at A: Lat: 52.629560, Long: -1.138103

4.1.4.6 Tracking Location B

For camera location B; located 9 metres directly in front of the Dry riser, the raw camera matrix was recorded as:

$$\text{(4.33)} \quad \mathbf{M} = \begin{pmatrix} \mathbf{-0.028} & \mathbf{0.939} & \mathbf{0.971} & \mathbf{-0.443} \\ \mathbf{-0.998} & \mathbf{-0.070} & \mathbf{-0.004} & \mathbf{-0.175} \\ \mathbf{0.065} & \mathbf{-0.937} & \mathbf{0.343} & \mathbf{3.070} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

After allowing for the iPhone transformation, the camera matrix, C, is:

$$\text{(4.34)} \quad \mathrm{C} = \begin{pmatrix} 0.998 & 0.070 & 0.004 & 0.175 \\ -0.065 & 0.937 & -0.343 & -3.070 \\ -0.028 & 0.343 & 0.939 & -0.443 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The inverse of C is:

$$\text{(4.35)} \quad \mathbf{C^{-1}} = \begin{pmatrix} \mathbf{0.997} & \mathbf{-0.064} & \mathbf{-0.028} & \mathbf{-0.384} \\ \mathbf{0.071} & \mathbf{0.937} & \mathbf{-0.342} & \mathbf{3.015} \\ \mathbf{0.004} & \mathbf{-0.344} & \mathbf{0.939} & \mathbf{-0.641} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

This gave x, y, z camera location;

$$(4.36) \quad x = -0.384\,;\ y = 3.015\,;\ z = -0.641$$

Applying 2.7 scaling factor (from calibration in location C) as before;

$$\text{(4.37)} \quad X' = -1.037m.\ ;\ Y' = 8.141\,m.\ ;\ Z' = -1.731m.$$

And after rotation to E, N, U;

$$\text{(4.38)} \quad E = -1.105m\,;\ N = 8.131m\,;\ U = -1.731m$$

Resulting in geodetic coordinates;

(4.39) $Latitude$: 52.629600 ; $Longitude$: -1.138112

Figures 4.15 to 4.17 show how the original GPS-only tracking results were improved combining that with the visual SLAM tracking data at Location B.

Figure 4.15 : Actual Location at B: Lat: 52.629583, Long: -1.138167

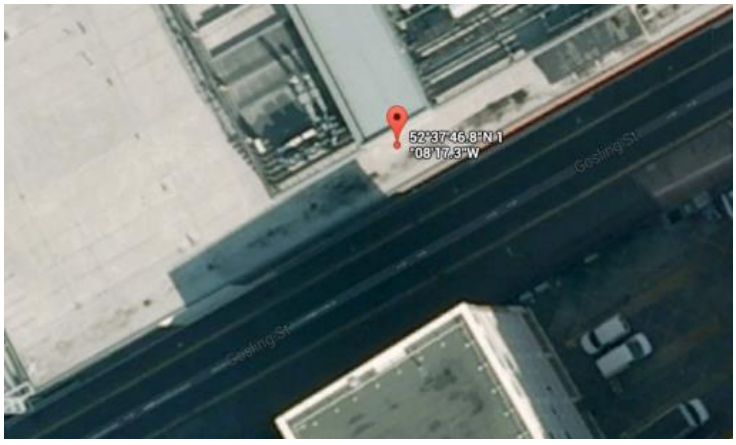


Figure 4.16 : iPhone GPS reading at B: Lat: 52.629668, Long: -1.138139

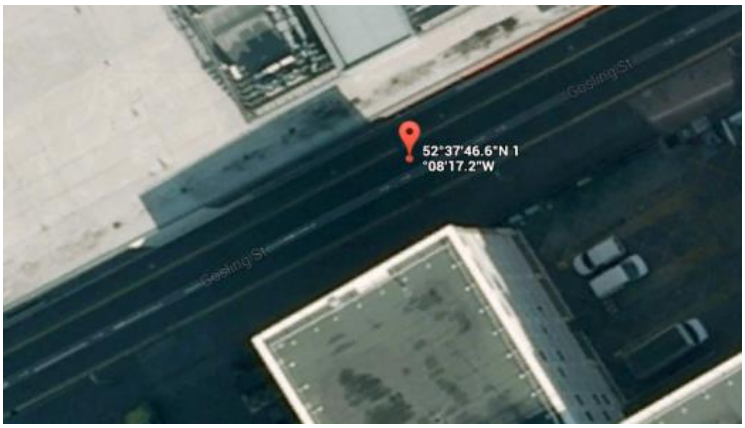


Figure 4.17 : GPS_SLAM measure at B: Lat: 52.629600, Long: -1.138112

4.1.4.7 Tracking Location D

For camera location D; located 5 metres to the right of the 'Dry riser' and 4.5 metres in front of the wall, with the tracking relying on SLAM as the original visual target was not in view at the end of the camera movement. The raw matrix was recorded as:

(4.40) $$\boldsymbol{M} = \begin{pmatrix} \mathbf{-0.004} & \mathbf{0.288} & \mathbf{0.958} & \mathbf{-0.383} \\ \mathbf{-0.930} & \mathbf{-0.354} & \mathbf{0.103} & \mathbf{2.198} \\ \mathbf{0.369} & \mathbf{-0.890} & \mathbf{0.269} & \mathbf{1.049} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

After allowing for the iPhone transformation, the camera matrix, C and C^{-1}, became;

(4.41) $$\mathbf{C} = \begin{pmatrix} \mathbf{0.930} & \mathbf{0.354} & \mathbf{-0.103} & \mathbf{-2.198} \\ \mathbf{-0.369} & \mathbf{0.890} & \mathbf{-0.269} & \mathbf{-1.049} \\ \mathbf{-0.004} & \mathbf{0.288} & \mathbf{0.958} & \mathbf{-0.383} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

(4.42) $$\mathbf{C^{-1}} = \begin{pmatrix} \mathbf{0.929} & \mathbf{-0.368} & \mathbf{-0.004} & \mathbf{1.654} \\ \mathbf{0.354} & \mathbf{0.890} & \mathbf{0.288} & \mathbf{1.822} \\ \mathbf{-0.103} & \mathbf{-0.269} & \mathbf{0.957} & \mathbf{-0.141} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

Giving x, y, z camera location;

(4.43) $x = 1.654$; $y = 1.822$; $z = -0.141$

Applying the same 2.7 scaling factor as before;

(4.44) $X' = 4.466m.$; $Y' = 4.919m.$; $Z' = -0.381m.$

And after rotation to E, N, U;

(4.45) $E = -5.587m$; $N = 3.596m$; $U = -0.381m$

Giving geodetic coordinates;

(4.46) $Latitude: 52.629556\ ;\ Longitude:\ -1.138177$

Figures 4.18 to 4.20 show how the original GPS-only tracking results were improved combining that with the visual SLAM tracking data at Location D.

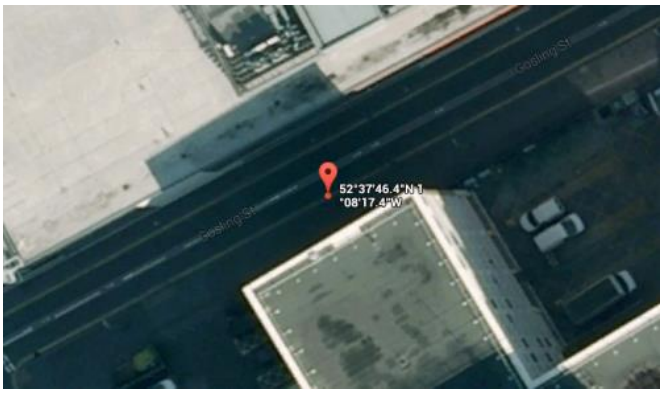


Figure 4.18 : Actual Location at D: Lat: 52.629556, Long: -1.138167

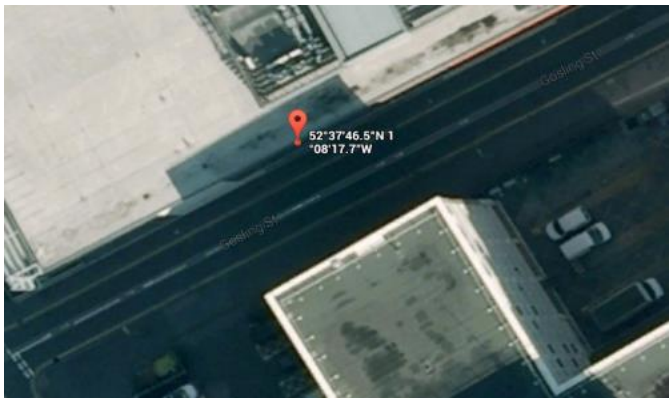


Figure 4.19 : iPhone GPS reading at D: Lat: 52.629583, Long: -1.138250

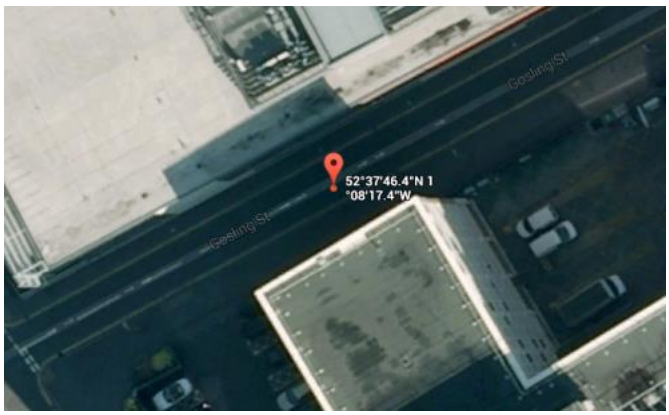


Figure 4.20 : PS_SLAM measure at D: Lat: 52.629556, Long: -1.138177

4.2 iBeacons for Micro-Location

This section summarises an experiment in micro-location estimation carried out using iBeacons in order to estimate receiver location using RSSI. As explained earlier, iBeacons were recently introduced by Apple© for micro-location tracking and estimation using RSSI.

Presented here are a set of experiments performed using iBeacons to assess the accuracy of an RSS-based system for micro-location tracking considering based on signal strength. In this experiment, an array of 3 iBeacons was arranged in an indoor environment as depicted in Figure 4.21. The iBeacons were placed in an environment of 10 x 10 tiled carpets where each tile is a square of 0.5 x 0.5 meters. Note that the object circled in the centre was the iPhone receiver. The third iBeacon was just out of frame to the right of the photograph.

Figure 4.21 : Experiment with iBeacons for Micro-Location

The tile dimensions in metres were used to determine a scale factor to convert tile-length based units, used for simplicity during experimentation, to the metric system. With a mobile phone placed in the centre each tile, as the receiver, the

RSS value was measured for each tile position and distance calculated using the following signal propagation relationship:

(4.47) $\mathbf{RSSI} = -\mathbf{10n}\log_{\mathbf{10}}\mathbf{d} + \mathbf{A}$

n is the signal propagation constant (set n=2 for free space reference), d is distance from the iBeacon and A (TxPower) represents the characteristic transmitted signal power from an individual iBeacon, defined as the dBm measured at a distance of $1\mathrm{m}$ from the transmitter. Therefore d at each location could be determined using:

(4.48) $\boldsymbol{d} = \mathbf{10}^{\frac{(-RSSI-A)}{10n}}$

Table 4.4 shows the measured RSSI values received by the mobile phone when it was located at the centre of each tile. Table 4.4 is the result of applying equation (4.48) to determine the estimated distance of the receiver from each iBeacon. Each cell contains three numbers referring to the distance between the phone and each iBeacon; the top value in each triplet being distance from Beacon 1; middle value, the distance from Beacon 2; and the third value, the distance from Beacon 3. The metric distance from the centre of each tile to the centre of the next tile was 0.5 metres.

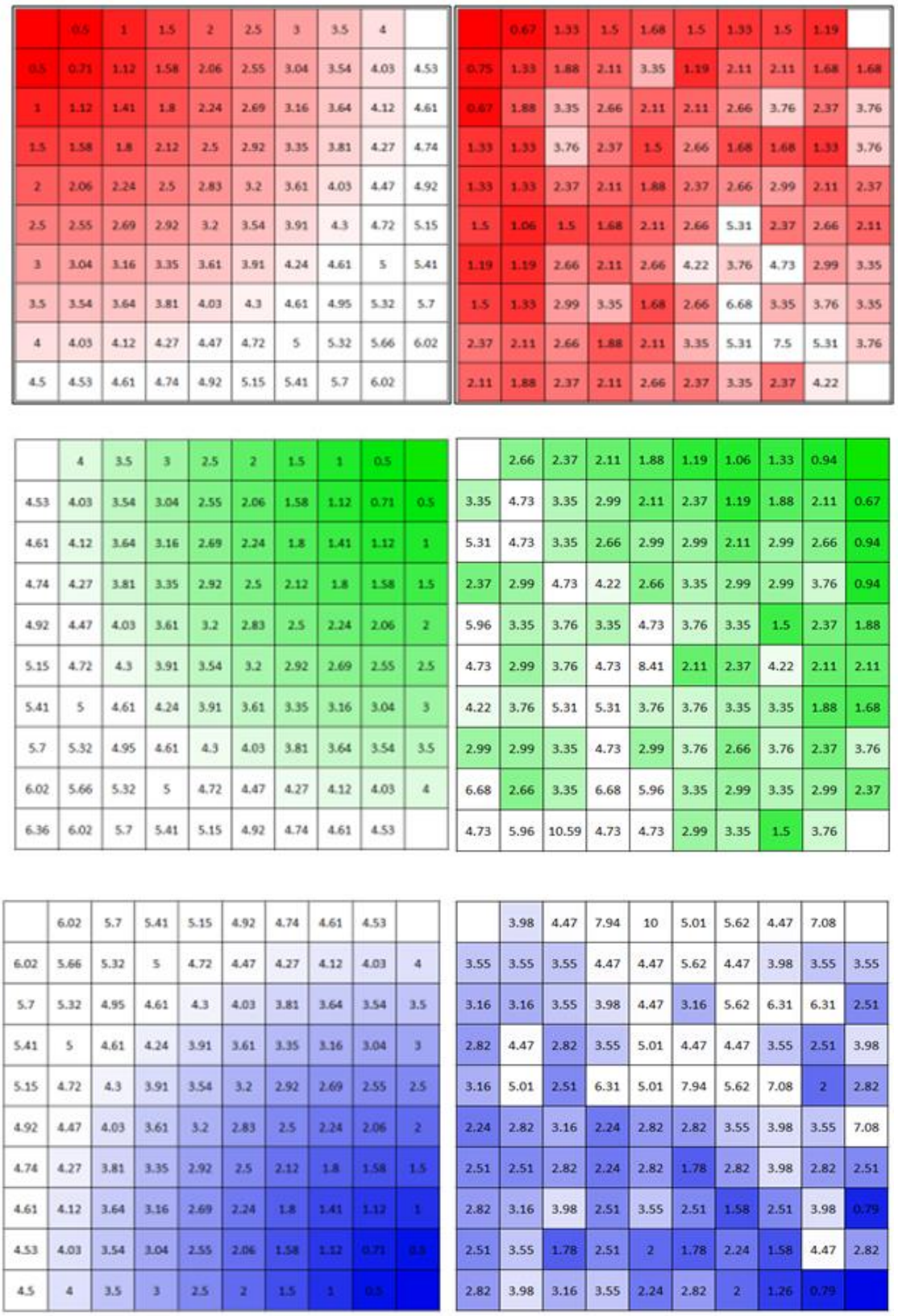
	1	2	3	4	5	6	7	8	9	10
1	**Beacon 1**	-67 -75 -77	-73 -74 -78	-74 -73 -83	-75 -72 -85	-74 -68 -79	-73 -67 -80	-74 -69 -78	-72 -66 -82	**Beacon 2**
2	-68 -77 -76	-73 -80 -76	-76 -77 -76	-77 -76 -78	-81 -73 -78	-72 -74 -80	-77 -68 -78	-77 -72 -77	-75 -73 -76	-75 -63 -76
3	-67 -81 -75	-76 -80 -75	-81 -77 -76	-79 -75 -77	-77 -76 -78	-77 -76 -75	-79 -73 -80	-82 -76 -81	-78 -75 -81	-82 -66 -73
4	-73 -74 -74	-73 -76 -78	-82 -80 -74	-78 -79 -76	-74 -75 -79	-79 -77 -78	-75 -76 -78	-75 -76 -76	-73 -78 -73	-82 -66 -77
5	-73 -82 -75	-73 -77 -79	-78 -78 -73	-77 -77 -81	-76 -80 -79	-78 -78 -83	-79 -77 -80	-80 -70 -82	-77 -74 -71	-78 -72 -74
6	-74 -80 -72	-71 -76 -74	-74 -78 -75	-75 -80 -72	-77 -85 -74	-79 -73 -74	-85 -74 -76	-78 -79 -77	-79 -73 -76	-77 -73 -82
7	-72 -79 -73	-72 -78 -73	-79 -81 -74	-77 -81 -72	-79 -78 -74	-83 -78 -70	-82 -77 -74	-84 -77 -77	-80 -72 -74	-81 -71 -73
8	-74 -76 -74	-73 -76 -75	-80 -77 -77	-81 -80 -73	-75 -76 -76	-79 -78 -73	-87 -75 -69	-81 -78 -73	-82 -74 -77	-81 -78 -63
9	-78 -83 -73	-77 -75 -76	-79 -77 -70	-76 -83 -73	-77 -82 -71	-81 -77 -70	-85 -76 -72	-88 -77 -69	-85 -76 -78	-82 -74 -74
10	-77 -80 -74	-76 -82 -77	-78 -87 -75	-77 -80 -76	-79 -80 -72	-78 -76 -74	-81 -77 -71	-78 -70 -67	-83 -78 -63	**Beacon 3**

Table 4.4 : Raw RSSI data for each tile (Each cell represents three readings considering the RSS from iBeacon 1, 2, and respectively 3)

	1	2	3	4	5	6	7	8	9	10
1	**Beacon 1**	0.67 2.66 3.98	1.33 2.37 4.47	1.5 2.11 7.94	1.68 1.88 10	1.5 1.19 5.01	1.33 1.06 5.62	1.5 1.33 4.47	1.19 0.94 7.08	**Beacon 2**
2	0.75 3.35 3.55	1.33 4.73 3.55	1.88 3.35 3.55	2.11 2.99 4.47	3.35 2.11 4.47	1.19 2.37 5.62	2.11 1.19 4.47	2.11 1.88 3.98	1.68 2.11 3.55	1.68 0.67 3.55
3	0.67 5.31 3.16	1.88 4.73 3.16	3.35 3.35 3.55	2.66 2.66 3.98	2.11 2.99 4.47	2.11 2.99 3.16	2.66 2.11 5.62	3.76 2.99 6.31	2.37 2.66 6.31	3.76 0.94 2.51
4	1.33 2.37 2.82	1.33 2.99 4.47	3.76 4.73 2.82	2.37 4.22 3.55	1.5 2.66 5.01	2.66 3.35 4.47	1.68 2.99 4.47	1.68 2.99 3.55	1.33 3.76 2.51	3.76 0.94 3.98
5	1.33 5.96 3.16	1.33 3.35 5.01	2.37 3.76 2.51	2.11 3.35 6.31	1.88 4.73 5.01	2.37 3.76 7.94	2.66 3.35 5.62	2.99 1.5 7.08	2.11 2.37 2	2.37 1.88 2.82
6	1.5 4.73 2.24	1.06 2.99 2.82	1.5 3.76 3.16	1.68 4.73 2.24	2.11 8.41 2.82	2.66 2.11 2.82	5.31 2.37 3.55	2.37 4.22 3.98	2.66 2.11 3.55	2.11 2.11 7.08
7	1.19 4.22 2.51	1.19 3.76 2.51	2.66 5.31 2.82	2.11 5.31 2.24	2.66 3.76 2.82	4.22 3.76 1.78	3.76 3.35 2.82	4.73 3.35 3.98	2.99 1.88 2.82	3.35 1.68 2.51
8	1.5 2.99 2.82	1.33 2.99 3.16	2.99 3.35 3.98	3.35 4.73 2.51	1.68 2.99 3.55	2.66 3.76 2.51	6.68 2.66 1.58	3.35 3.76 2.51	3.76 2.37 3.98	3.35 3.76 0.79
9	2.37 6.68 2.51	2.11 2.66 3.55	2.66 3.35 1.78	1.88 6.68 2.51	2.11 5.96 2	3.35 3.35 1.78	5.31 2.99 2.24	7.5 3.35 1.58	5.31 2.99 4.47	3.76 2.37 2.82
10	2.11 4.73 2.82	1.88 5.96 3.98	2.37 10.59 3.16	2.11 4.73 3.55	2.66 4.73 2.24	2.37 2.99 2.82	3.35 3.35 2	2.37 1.5 1.26	4.22 3.76 0.79	**Beacon 3**

Table 4.5 : RSSI converted to distance (d) as in Equation (4.48)

Table 4.7 also shows the calculated distances separated out for each iBeacon with cells shaded relative to distance such that fully saturated colour represents small distance and low saturation indicating greater distances with white at distances >= 4.5 metres. For each iBeacon, the first table shows the actual distances and the second, the calculated distances.



	0.5	1	1.5	2	2.5	3	3.5	4	
0.5	0.71	1.12	1.58	2.06	2.55	3.04	3.54	4.03	4.53
1	1.12	1.41	1.8	2.24	2.69	3.16	3.64	4.12	4.61
1.5	1.58	1.8	2.12	2.5	2.92	3.35	3.81	4.27	4.74
2	2.06	2.24	2.5	2.83	3.2	3.61	4.03	4.47	4.92
2.5	2.55	2.69	2.92	3.2	3.54	3.91	4.3	4.72	5.15
3	3.04	3.16	3.35	3.61	3.91	4.24	4.61	5	5.41
3.5	3.54	3.64	3.81	4.03	4.3	4.61	4.95	5.32	5.7
4	4.03	4.12	4.27	4.47	4.72	5	5.32	5.66	6.02
4.5	4.53	4.61	4.74	4.92	5.15	5.41	5.7	6.02	

	0.67	1.33	1.5	1.68	1.5	1.33	1.5	1.19	
0.75	1.33	1.88	2.11	3.35	1.19	2.11	2.11	1.68	1.68
0.67	1.88	3.35	2.66	2.11	2.11	2.66	3.76	2.37	3.76
1.33	1.33	3.76	2.37	1.5	2.66	1.68	1.68	1.33	3.76
1.33	1.33	2.37	2.11	1.88	2.37	2.66	2.99	2.11	2.37
1.5	1.06	1.5	1.68	2.11	2.66	5.31	2.37	2.66	2.11
1.19	1.19	2.66	2.11	2.66	4.22	3.76	4.73	2.99	3.35
1.5	1.33	2.99	3.35	1.68	2.66	6.68	3.35	3.76	3.35
2.37	2.11	2.66	1.88	2.11	3.35	5.31	7.5	5.31	3.76
2.11	1.88	2.37	2.11	2.66	2.37	3.35	2.37	4.22	

	4	3.5	3	2.5	2	1.5	1	0.5	
4.53	4.03	3.54	3.04	2.55	2.06	1.58	1.12	0.71	0.5
4.61	4.12	3.64	3.16	2.69	2.24	1.8	1.41	1.12	1
4.74	4.27	3.81	3.35	2.92	2.5	2.12	1.8	1.58	1.5
4.92	4.47	4.03	3.61	3.2	2.83	2.5	2.24	2.06	2
5.15	4.72	4.3	3.91	3.54	3.2	2.92	2.69	2.55	2.5
5.41	5	4.61	4.24	3.91	3.61	3.35	3.16	3.04	3
5.7	5.32	4.95	4.61	4.3	4.03	3.81	3.64	3.54	3.5
6.02	5.66	5.32	5	4.72	4.47	4.27	4.12	4.03	4
6.36	6.02	5.7	5.41	5.15	4.92	4.74	4.61	4.53	

	2.66	2.37	2.11	1.88	1.19	1.06	1.33	0.94	
3.35	4.73	3.35	2.99	2.11	2.37	1.19	1.88	2.11	0.67
5.31	4.73	3.35	2.66	2.99	2.99	2.11	2.99	2.66	0.94
2.37	2.99	4.73	4.22	2.66	3.35	2.99	2.99	3.76	0.94
5.96	3.35	3.76	3.35	4.73	3.76	3.35	1.5	2.37	1.88
4.73	2.99	3.76	4.73	8.41	2.11	2.37	4.22	2.11	2.11
4.22	3.76	5.31	5.31	3.76	3.76	3.35	3.35	1.88	1.68
2.99	2.99	3.35	4.73	2.99	3.76	2.66	3.76	2.37	3.76
6.68	2.66	3.35	6.68	5.96	3.35	2.99	3.35	2.99	2.37
4.73	5.96	10.59	4.73	4.73	2.99	3.35	1.5	3.76	

	6.02	5.7	5.41	5.15	4.92	4.74	4.61	4.53	
6.02	5.66	5.32	5	4.72	4.47	4.27	4.12	4.03	4
5.7	5.32	4.95	4.61	4.3	4.03	3.81	3.64	3.54	3.5
5.41	5	4.61	4.24	3.91	3.61	3.35	3.16	3.04	3
5.15	4.72	4.3	3.91	3.54	3.2	2.92	2.69	2.55	2.5
4.92	4.47	4.03	3.61	3.2	2.83	2.5	2.24	2.06	2
4.74	4.27	3.81	3.35	2.92	2.5	2.12	1.8	1.58	1.5
4.61	4.12	3.64	3.16	2.69	2.24	1.8	1.41	1.12	1
4.53	4.03	3.54	3.04	2.55	2.06	1.58	1.12	0.71	0.5
4.5	4	3.5	3	2.5	2	1.5	1	0.5	

	3.98	4.47	7.94	10	5.01	5.62	4.47	7.08	
3.55	3.55	3.55	4.47	4.47	5.62	4.47	3.98	3.55	3.55
3.16	3.16	3.55	3.98	4.47	3.16	5.62	6.31	6.31	2.51
2.82	4.47	2.82	3.55	5.01	4.47	4.47	3.55	2.51	3.98
3.16	5.01	2.51	6.31	5.01	7.94	5.62	7.08	2	2.82
2.24	2.82	3.16	2.24	2.82	2.82	3.55	3.98	3.55	7.08
2.51	2.51	2.82	2.24	2.82	1.78	2.82	3.98	2.82	2.51
2.82	3.16	3.98	2.51	3.55	2.51	1.58	2.51	3.98	0.79
2.51	3.55	1.78	2.51	2	1.78	2.24	1.58	4.47	2.82
2.82	3.98	3.16	3.55	2.24	2.82	2	1.26	0.79	

Table 4.6 : Distance from each iBeacon (metres) (The colour shades illustrate how far the receiver was from the beacon – Red : iBeacon1, Green: iBeacon 2, and Blue : iBeacon 3)

The scale of the colour scheme used in this table represents the relative distance from the iBeacon. The darker the colour of each cell, the smaller the distance of that cell to the the iBeacon should be. However, due to RSS errors, this is not always as expected (as seen in the second set of tables (right) which reported the actual measurements).

4.2.1 Error Measurements using Trilateration Calculations

In order to combine each distance triplet to determine the receiver location, trilateration is required (Cook, et al., 2006). In a two-dimensional arrangement, as effectively used in this experiment, trilateration calculates the location of the receiver based on the geometry of signal-strength circles centred on each iBeacon transmitter. In an ideal system, the distance of the receiver from each iBeacon represented by the circles in Figure 4.22, will intersect at the location of the receiver.

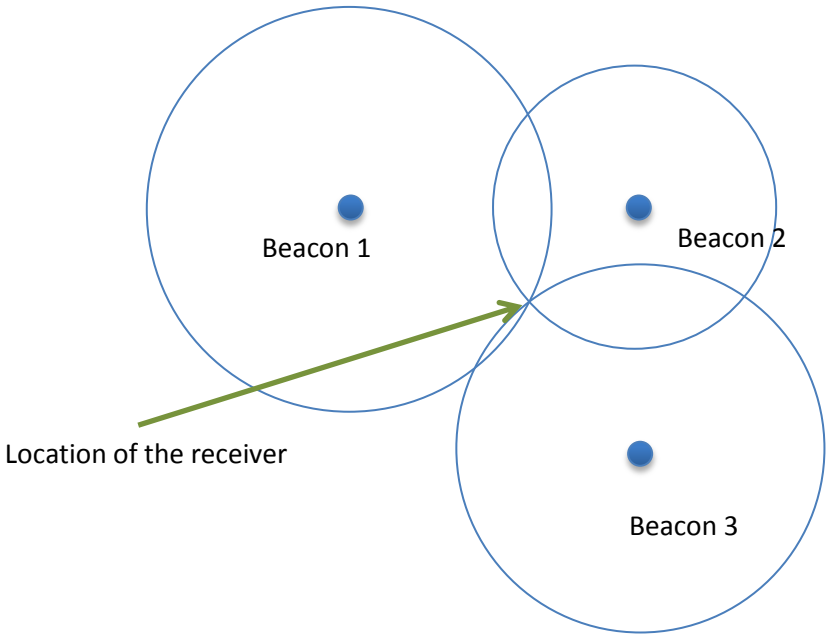


Figure 4.22 : Trilateration with iBeacons – ideal location of receiver

However, in practice, the circles are unlikely to intersect at an exact point, therefore trilateration is used to calculate the centre of intersection based on the

relative circle radii as shown in Figure 4.23. It is also possible that the circles may not intersect at all.

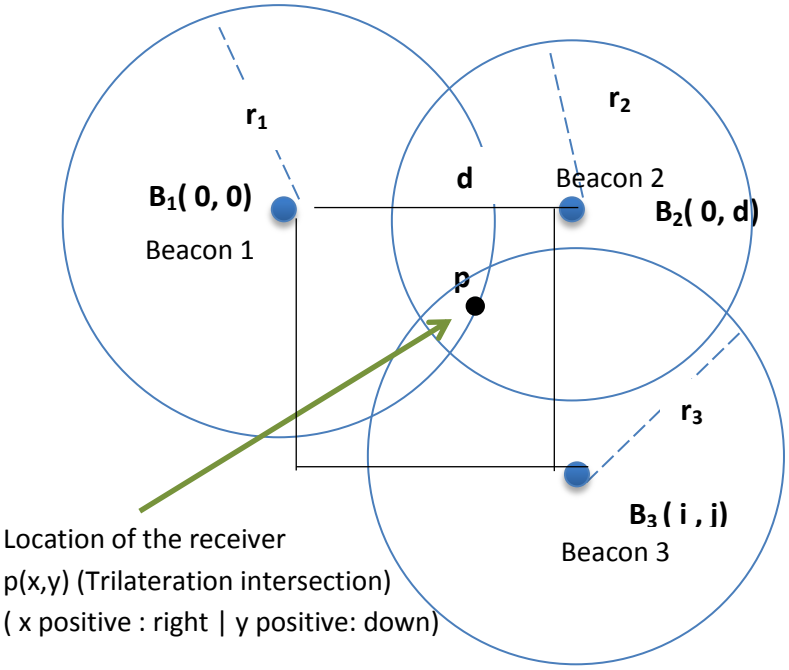


Figure 4.23 : Trilateration with iBeacons – practical location of the receive

In these situations, the receiver location can be estimated by finding a point that minimizes the distance to all of the circles, using Least Squares Estimation (Stüber & Caffrey, 1999). In a simple case, such as this where the iBeacons are are on the z=0 plane with one iBeacon at the coordinate system origin and another on the x-axis, it is possible to use a geometric solution based on vectors between the iBeacons.

Given the iBeacon locations B_1, B_2 and B_3 measured as vectors from the coordinate system origin and assuming a 2D system; i.e. z = 0, the unit vector, $\hat{\mathrm{e}}_\mathrm{x}$, from B1 to B2 is given by :

(4.49) $$\hat{e}_x = \frac{(\ B_2 - B_1\ )}{\|\ B_2 - B_1\ \|}$$

(4.50) $$\mathrm{i} = \hat{\mathrm{e}}_\mathrm{x}\,(\mathrm{B}_3 - \mathrm{B}_1\ )$$

where i is the signed magnitude of the x component of vector from B1 to B3.

The unit vector in the y direction is also given by :

(4.51) $\hat{e}_y = \frac{(B_3 - B_1 - i\hat{e}_x)}{\| B_3 - B_1 - i\hat{e}_x \|}$

The third basis unit vector $\hat{e}_z$ can be determined by the cross product of $\hat{e}_x$ and $\hat{e}_y$.

(4.52) $\hat{e}_z = \hat{e}_x \times \hat{e}_y$

The distance between the iBeacons, B1 and B2 (d) can be determined from

(4.53) $d = \| B_2 - B_1 \|$

Furthermore j is the signed magnitude of the y component vector from B_1 to B_3 and can be determined by:

(4.54) $j = \hat{e}_y . (B_3 - B_1)$

The receiver location estimate is then given by (4.55) as:

(4.55) $P = B_1 + x.\hat{e}_x + y.\hat{e}_y \pm z.\hat{e}_z$

The intersections of the surfaces of three spheres is found by formulating the equations for the three sphere surfaces and then solving the three equations for the three unknowns, x, y, and z. To simplify the calculations, the equations are formulated so that the centres of the spheres are on the z = 0 plane. Also, the formulation is such that one centre is at the origin, and one other is on the x-axis.

Considering the above assumptions, the trilateration intersection of three iBeacons (p) as illustrated in Figure 4.23 (when z = 0) can be calculated as in in equations (4.56) to (4.60) .

(4.56) $x = \frac{r_1{}^2 - r_2{}^2 + d^2}{2d}$

(4.57) $y = \frac{r_1{}^2 - r_3{}^2 + i^2 + j^2}{2j} - \frac{i}{j} x$

Where :

(4.58) $r_1{}^2 = x^2 + y^2$

(4.59) $r_2{}^2 = (x - d)^2 + y^2$

(4.60) $r_3{}^2 = (x - i)^2 + (y - j)^2$

d = B_1 B_2 : The distance between iBeacon 1 and 2

p (x,y) : coordinates of the trilateration intersection

B_3 (i, j) : is the coordinate of B_3

r_1, r_2, r_3 the sphere radii of the iBeacons

The above was implemented in an Excel spreadsheet with a worksheet cell representing each of the tile locations. Refer to Appendix C for the results of the trilateration calculations. As the actual coordinates of each tile centre are known this was used to calculate the experimentally measured location for each tile centre. In these calculations, a scaling factor of 0.5 in each direction was used to convert from tile grid number (tile dimensions of 0.5 x 0.5 metres) to metres. Finally, the mean errors were calculated considering the distance between the measured and actual locations in metres (see Table 4 for the final results of these experiments).

Minimum Error	Maximum Error	Mean Error
0.21 (m)	13.97 (m)	2.2 (m)

Table 4: Trilateration Error

4.3 Discussion and Conclusion

The experiments carried out with GPS_Visual SLAM system with sets of experimental measurements (presented in 4.1.4) demonstrate the potential for utilising visual images from the real-world in obtaining more accurate GPS location on a mobile device. The fusion of the GPS and Visual SLAM was found to enhance the performance of the GPS-only system. Visual-SLAM also allows for some continuity of tracking after the initial visual target has been lost from view. However, it was found that using the PointCloud.io library, continuous tracking under these conditions was difficult to maintain and lost easily.

Nevertheless, given a number of reference images in a given location, an initial approximate GPS location reading would bound the image subset that needs to be searched using visual-SLAM to refine the GPS location.

In the second experiment with iBeacons, the results indicated a lower accuracy for micro-location estimation than would be hoped for, although somewhat better in accuracy than raw GPS. This questions Apple's claim in using iBeacons alone for micro-location applications. However, iBeacons have the advantage over GPS of working in both indoor and outdoor environments.

In summary, both experiments with GPS_Visual SLAM and iBeacons using RSSI measurement provided results showing that these too have their own limitations. GPS_Visual SLAM improved the typical low range accuracy of GPS, but was difficult to maintain when relying on its own extensible map of the environment especially in circumstances where landmarks lack many unique visual features. On the other hand, iBeacons also offer better accuracy than GPS alone and have the added advantage of working in indoor environments. However, RSS proved to be an unreliable measure of actual distance for micro-location purposes. In the above experiment, only three iBeacons were employed. A higher density of iBeacons could possibly be used to increase accuracy. Also, iBeacons could be employed in a hybrid system, fusing with data from accelerometers and gyroscopes.

CHAPTER FIVE

5 Geometric Models and Mathematical Tools for Camera Modelling and Representation of 3D Moving Objects

This chapter introduces the mathematical models and tools for image formation, three-dimensional representation of moving objects and related geometric constraints. This chapter also provides a background for better understanding of the operating principles, mathematical and geometric models, measurement of the camera's intrinsic and extrinsic parameters, three-dimensional representation of moving objects and how to model camera pose with reference to real-world objects. In addition, geometric constraints arising from matching feature points are outlined. These constraints are used in 3D pose estimation. The mathematical notations and models in this chapter are mainly adopted from (Ma, et al., 2004).

5.1 Image Formation

Image formation refers to the process of constructing an image corresponding to a physical object (Ma, et al., 2004). Computer vision algorithms first require development of a suitable model for image formation. Such models combine physical and mathematical constraints in order to produce a manageable interpretation for solving computer vision problems.

For centuries, the study of image formation has attracted the interest of the artistic reproduction and composition society more than that of mathematics and engineering. Understanding the geometry of the image, which includes various

models for projecting three dimensional world objects onto a two dimensional plane such as a canvas (3D/2D mapping), is implicit in many branches of the visual arts. The root of formalising the geometry of image formation into mathematical models can be tracked back to the work of Euclid in the 4th century B.C. Examples of applying perspective projection can also be seen in the frescoes and mosaics of Pompeii from the 1st century B.C (see Figure 5.1).

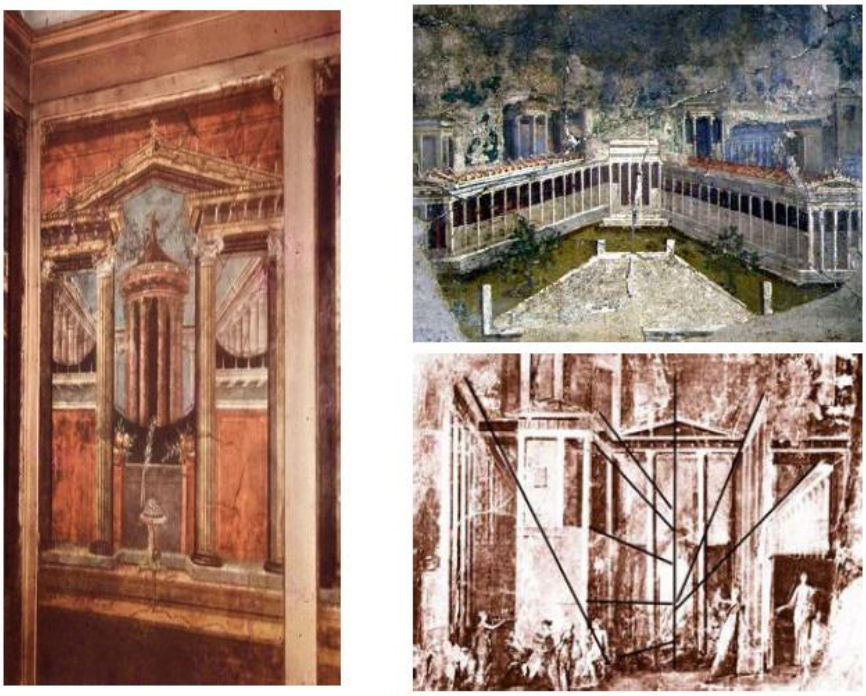


Figure 5.1 : Frescoes from the first century B.C. in Pompeii. More (left) or less (right) correct perspective projection is visible in the paintings
Source: (Ma, et al., 2004)

The early Renaissance painters developed systematic methods for determining perspective projection of three dimensional landscapes. The treatise of 'Della Pictura' published by Leon Battisa Alberti, is an example of a very early exploitation of perspective projection. He was an artist who was also proficient in engineering and architecture. He emphasised the importance of the *eye's view* of the world capturing correctly the geometry of the projection process, which is the basis of image geometry. However, consideration of geometry is not the only important part of the image formation process. In order to obtain an image, one needs to decide not only where to draw a point but also what

brightness value to assign to it. The interaction of light was at the core of the studies by Leonardo Da Vinci, and his ideas are vibrantly expressed in his surviving notes. Caravaggio and Raphael also exhibited sophisticated skills in rendering light and colour. There is also some evidence to suggest that some Renaissance artists used camera-like devices (camera obscura) (Hockney, 2001).

5.2 Geometric Models for Image Formation

A computer-based grayscale image can be envisaged as a two-dimensional brightness array. Similarly a colour image can be specified by three such arrays each representing one of the red, green and blue primary colours. In other words an image is represented by a map I on a 2D region Ω, assigning a positive real value to each point in this region. For the case of a camera, Ω is a planar, rectangular region formed on photographic medium or a CCD (Charge-Coupled Device) sensor. Equation (5.1) formulates the mapping process:

(5.1) $$\boldsymbol{I}: \boldsymbol{\Omega} \subset \mathbb{R}^{\mathbf{2}} \rightarrow \mathbb{R}^{+}, \qquad (\boldsymbol{x}, \boldsymbol{y}) \mapsto \boldsymbol{I}(\boldsymbol{x}, \boldsymbol{y})$$

In order to quantify the image formation, the value of $I(x, y)$, which is often referred to as the *image intensity* or *brightness,* must be specified at each point of (x, y). This parameter can be specified in units of power per area (W/m^2). The image pixel intensity at point (x, y) is obtained by integrating energy both with respect to time and space. The length of time depends on the shutter interval of a camera or the integration time in a CCD array. The integration space is formed by the part(s) of the object(s) contributing to the formation of the image pixel at point (x, y) and depends on various factors such as the shape of the object, the optic characteristics of the imaging device and so on.

5.3 Camera Imaging Models

A digital camera consists of two main parts; an optical system (lens system) and an image sensor, either using CCD or CMOS (Complementary Metal Oxide Semiconductor) technology. An image sensor converts the light to a digital image/map, consisting of a 2D array of pixels, each occupying a certain size on the surface area of the imaging sensor. This process is referred to as digitisation. Image pixels in a single image are normally equal in shape and size.

A camera is specified by a set of intrinsic and extrinsic parameters. The relationship between the metric camera coordinate system and the 2D array of pixels is defined by the camera intrinsic parameters, which represent the linear image distortion, the sensor geometry and its sampling characteristics. Nonlinear distortion also affects image formation, however this factor is not considered in this thesis. Readers are invited to refer to section 3.3.3 of (Ma, et al., 2004) for further details. Distortion coefficients and intrinsic camera parameters are camera-specific and remain constant for a given camera-lens system. The process of estimating these parameters is sometimes referred to as camera calibration.

On the other hand, extrinsic camera parameters describe the position and orientation of the camera in a fixed reference frame. Ultimately, the purpose of camera tracking is to estimate these parameters.

5.3.1 Imaging through Lenses

Any optical sensor (e.g. a camera) is composed of a set of lenses used for controlling the direction and propagation of light, by means of diffraction, refraction and reflection. For simplicity, the effects of diffraction and reflection in a lens system are neglected here and only the refraction is considered,

although the full description of a purely refractive lens is also beyond the aims and scope of this thesis. However, for more details please refer to (Born & Wolf, 1999).

Throughout this thesis, the thin lens model, which is a mathematical model defined by an optical axis and a focal plane, is considered. The optical axis is perpendicular to the focal plane and at its intersection with the plane forms a circular aperture centred at the optical centre. The thin lens model has two parameters, the focal length f and lens diameter, the latter of which can be ignored considering the thin lens assumption.

The thin lens function is categorised by two main properties. First is that all rays entering the lens parallel to the optical axis intersect on the optical axis at distance f from the optical centre; i.e. they intersect on the focal centre of the lens. The second property is that the rays entering the optical centre are undeflected.

Let X be a 3D point in space mapped to point X' on the image plane using the above two lens properties. Figure 5.2 illustrates the mapping process. In this image X is at a distance Z along the optical axis not too far from the optical centre. Point X' is the image of point X, formed at distance Z' from the optical centre. Referring to Figure 5.2, the fundamental equation of a thin lens can be expressed as follows:

(5.2) $$\frac{\mathbf{1}}{\boldsymbol{Z}} = \frac{\mathbf{1}}{\boldsymbol{Z'}} + \frac{\mathbf{1}}{\boldsymbol{f}}$$

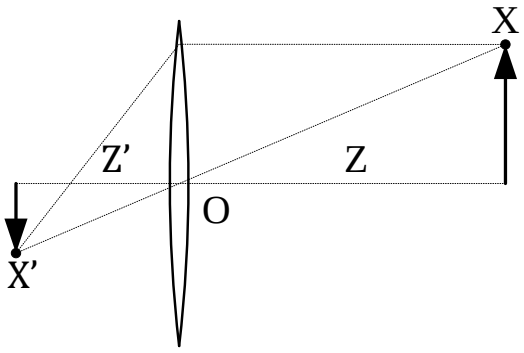


Figure 5.2 Thin Lens Model

5.3.2 Perspective Pinhole Cameras

The *Pinhole Perspective* - also called central *perspective projection model* – was first proposed as a convenient mathematical concept by Brunelleschi at the beginning of the 15th century. The perspective projection can be created by taking a box, pricking a small hole in one side of it with a pin, and then replacing the opposite side with a translucent plate acting as the image plane. By holding that box in a dimly lit room, with the pinhole facing a light source (i.e. a candle), one can see an inverted image of the candle appearing on the translucent plate (

Figure 5.3). This image is produced by light rays emitted from the scene in front of the box. If the pinhole were to be reduced down to an infinitesimally small point, then each scene point would have only one corresponding point on the image plane and exactly one light ray would pass through the scene point, the pinhole and the corresponding point on the image plane. This model, despite its simplicity, provides an acceptable approximation for many imaging applications.

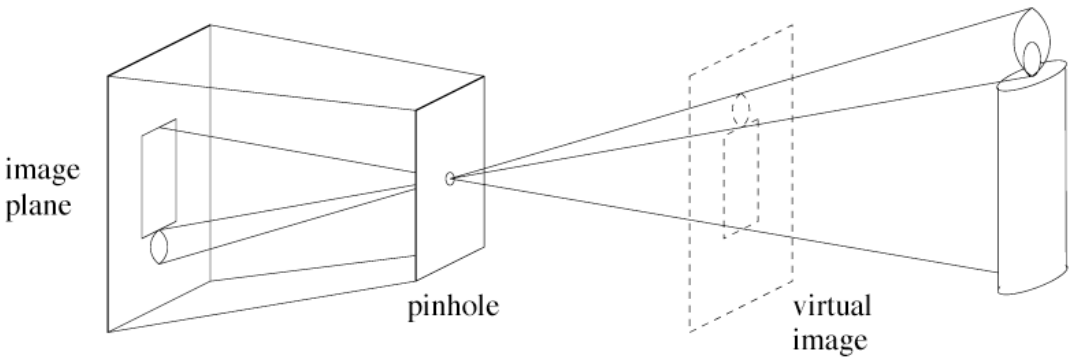


Figure 5.3 : Perspective Pinhole - source: (Forsyth & Ponce, 2003, p. 4)

Considering the thin lens model described earlier, we can also refer to a pinhole camera as a modified thin lens model. When the aperture of a thin lens is theoretically reduced to zero, all the rays are forced to go through the optical centre O, and therefore remain undeflected. The Pinhole camera is

mathematically modelled and the derived equations used as the basis for most computer vision computations. A point in 3D space is projected to the image plane by drawing a ray from the 3D point in the world towards the optical centre. The intersection of this ray and the image plane represents the image of the 3D point (see Figure 5.4). The shortest distance between the optical centre and the image plane is the focal length of the camera denoted as f.

To obtain a mathematical model, the 3D camera coordinate system C is defined here. In this coordinate system the origin is the camera's optical centre and the Z axis is along the vector perpendicular to the image plane facing towards the optical centre from the image plane. Figure 5.4 depicts the camera coordinate system.

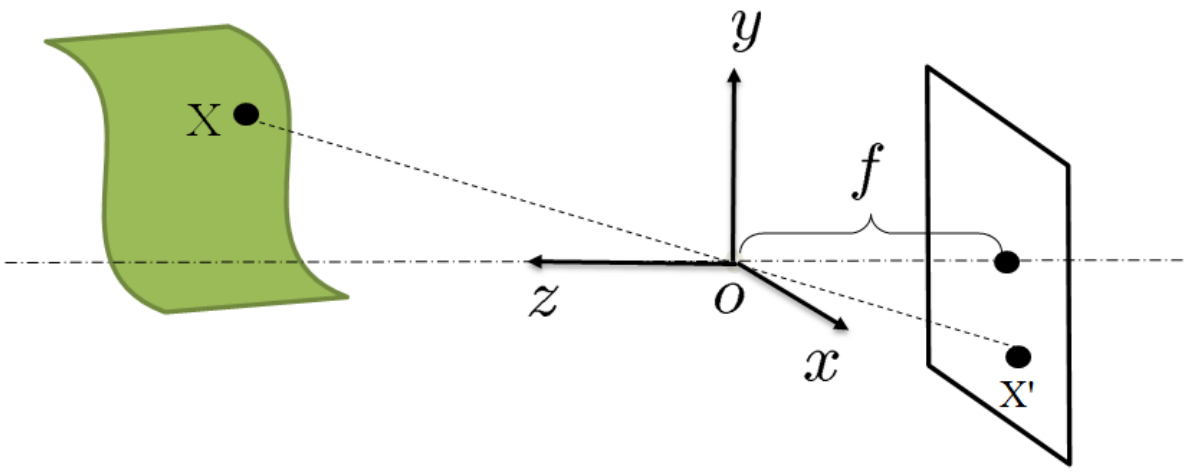


Figure 5.4 : Pinhole Model

Let X denote a 3D point in the scene and X′ the associated image point on the image plane.

(5.3) $$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \end{pmatrix}, \quad \boldsymbol{X'} = \begin{pmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \\ \boldsymbol{z'} \end{pmatrix}$$

Coordinates of both points are referenced to the camera coordinate system. X' lies on the image plane, therefore:

(5.4) $\boldsymbol{z'} = -\mathrm{f}$

Points X,O and X′ are collinear, therefore:

(5.5) $$OX = \lambda' OX' \implies \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \lambda' \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$$

Substituting (5.4) into (5.5) results in the following equations for projective camera imaging:

(5.6) $$\begin{pmatrix} x' \\ y' \end{pmatrix} = -\frac{f}{z}\begin{pmatrix} x \\ y \end{pmatrix}, \qquad \lambda' = -\frac{z}{f}$$

The negative sign in the above equation indicates that an object's image appears upside down on the image plane. To eliminate this effect and simplify the model, the image plane is flipped from behind the optical centre at $(z = -f)$ to the front at $(z = +f)$. The result is transferring the image point $(x', y')^T$ to $(-x', -y')^T$. This is referred to as Frontal Camera Pinhole Model as illustrated in Figure 5.5. Equation (5.7) shows the relationship between points X and X′ in the frontal camera model.

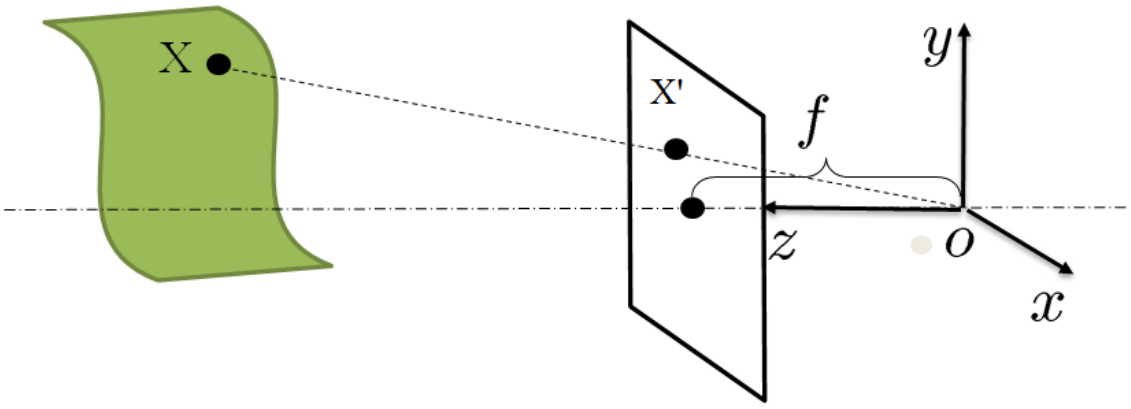


Figure 5.5 : Frontal Pinhole Model

(5.7) $$\begin{pmatrix} x' \\ y' \end{pmatrix} = +\frac{f}{z}\begin{pmatrix} x \\ y \end{pmatrix}$$

In homogenous coordinates the system the above can be described in matrix format as follows:

(5.8) $$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \quad \lambda = \frac{1}{z}$$

Letting p denote the homogeneous pixel coordinates of the image point X', the above equation can be re-written as follows:

(5.9) $$p = \lambda K_f \; \Pi_0 \; X$$

(5.10) $$p = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}, \; X = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \; K_f = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}, \; \Pi_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

5.3.3 Camera Parameters

5.3.3.1 Intrinsic Camera Parameters

The coordinates of a digital image are typically specified in pixels indexed from the top left corner (Figure 5.6). The parameters necessary to link the pixel coordinates of an image point to the corresponding coordinates in the camera reference frame are referred to as intrinsic camera parameters (see (5.10)).

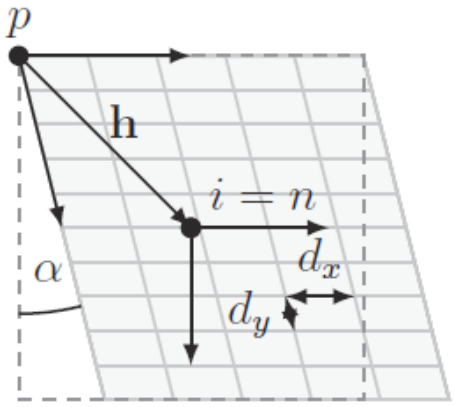


Figure 5.6 Pixel Coordinates

The intrinsic camera parameters are derived from perspective projection, the transformation between the image plane coordinates and pixel coordinates, and finally the geometric distortion introduced by the optics. The image pixels are characterised by their horizontal and vertical dimensions, dx and dy. A point p on the image plane with homogenous coordinates $(x', y', 1)^T$ in the metric system has the following coordinates in pixel units

(5.11) $$\begin{pmatrix} x_s \\ y_s \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

In the above equation s_x and s_y are the pixel densities along x and y axes.

(5.12) $$\begin{pmatrix} s_x \\ s_y \end{pmatrix} = \begin{pmatrix} 1/dx \\ 1/dy \end{pmatrix}$$

If the image plane is skewed as shown in Figure 5.6, the skew angle α also affects the intrinsic camera parameters and (5.12) ca be reformulated as :

(5.13) $$\begin{pmatrix} x_s \\ y_s \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & s_\alpha & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}, \qquad s_\alpha \propto \tan\alpha$$

Assuming that the centre of image is at $(o_x,\ o_y)^T$ in pixel units, the coordinates of point p in the image coordinate frame are :

(5.14) $$\begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = K_s \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}, \qquad K_s = \begin{pmatrix} s_x & s_\alpha & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{pmatrix}$$

Combining (5.8) and (5.14) yields:

(5.15) $$\begin{pmatrix} \mathrm{x_p} \\ \mathrm{y_p} \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} \mathrm{s_x} & \mathrm{s_\alpha} & \mathrm{o_x} \\ 0 & \mathrm{s_y} & \mathrm{o_y} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

The camera intrinsic matrix, K, is defined as it follows:

(5.16) $$K = \begin{pmatrix} \mathrm{s_x} & \mathrm{s_\alpha} & \mathrm{o_x} \\ 0 & \mathrm{s_y} & \mathrm{o_y} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} f\mathrm{s_x} & f\mathrm{s_\alpha} & \mathrm{o_x} \\ 0 & f\mathrm{s_y} & \mathrm{o_y} \\ 0 & 0 & 1 \end{pmatrix}$$

Matrix K converts the metric coordinates of a 3D point in space with reference to camera frame to pixel coordinates in the image coordinate system:

(5.17) $$\begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = \lambda K \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \lambda K \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

5.3.3.2 Extrinsic Camera Parameters – Camera Pose

The camera coordinate system as described in section 6.1.1 views 3D points in space differently to the fixed world coordinate system. The relationship between these two systems is defined by the camera extrinsic parameters. These parameters, in addition to the camera intrinsic parameters, are necessary in order to establish an accurate correspondence between the coordinates of a 3D point in space and the projected 2D point on the image plane. Figure 5.7 shows a camera undergoing a translation and rotation, represented by vector T and matrix R respectively.

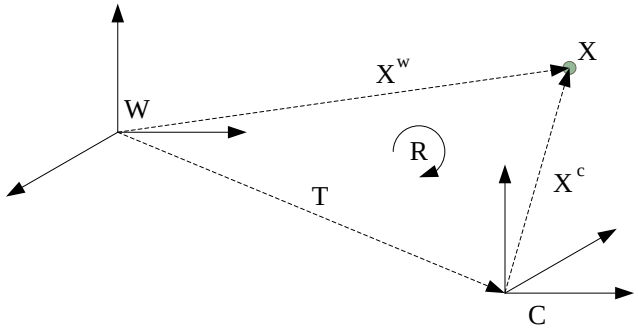


Figure 5.7 : Camera transformation

Let X^w and X^c represent the coordinates of a 3D point in space with reference to the world and camera frames, respectively. $X^w$and X^c hold the following relationship:

(5.18) $$X^w = RX^c + T$$

where T is the coordinate of the camera origin with respect to the world frame and R is the rotation matrix formed by rotation around x, y and z axes

(5.19) $\mathrm{T} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}$

(5.20) $\boldsymbol{R_x} = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{cos(\theta_x)} & \boldsymbol{-sin(\theta_x)} \\ \mathbf{0} & \boldsymbol{sin(\theta_x)} & \boldsymbol{cos(\theta_x)} \end{pmatrix}$

(5.21) $\boldsymbol{R_y} = \begin{pmatrix} \boldsymbol{cos(\theta_y)} & \mathbf{0} & \boldsymbol{sin(\theta_y)} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \boldsymbol{-sin(\theta_y)} & \mathbf{0} & \boldsymbol{cos(\theta_y)} \end{pmatrix}$

(5.22) $\boldsymbol{R_z} = \begin{pmatrix} \boldsymbol{cos(\theta_z)} & \boldsymbol{-sin(\theta_z)} & \mathbf{0} \\ \boldsymbol{sin(\theta_z)} & \boldsymbol{cos(\theta_z)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$

(5.23) $\boldsymbol{R = R_x\, R_y\, R_z}$

Matrix R can also be considered as a set of three vectors as follows:

(5.24) $\boldsymbol{R} = \begin{pmatrix} \overrightarrow{\boldsymbol{r_x}} & \overrightarrow{\boldsymbol{r_y}} & \overrightarrow{\boldsymbol{r_z}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r_{11}} & \boldsymbol{r_{12}} & \boldsymbol{r_{13}} \\ \boldsymbol{r_{21}} & \boldsymbol{r_{22}} & \boldsymbol{r_{23}} \\ \boldsymbol{r_{31}} & \boldsymbol{r_{32}} & \boldsymbol{r_{33}} \end{pmatrix}$

where $\overrightarrow{r_x}$, $\overrightarrow{r_y}$ and $\overrightarrow{r_z}$ are the unit vectors of the camera coordinate system (i_c, j_c and k_c) as seen by the world coordinate system.

(5.25) $\begin{pmatrix} \overrightarrow{\boldsymbol{r_x}} & \overrightarrow{\boldsymbol{r_y}} & \overrightarrow{\boldsymbol{r_z}} \end{pmatrix} = (\boldsymbol{i_c^w} \quad \boldsymbol{j_c^w} \quad \boldsymbol{k_c^w})$

In order to transform the coordinates of a 3D point in space from the world frame to the camera frame, equation (5.18) can be reformulated as:

(5.26) $X^c = R^{-1}(X^w - T)$

The camera extrinsic matrix is therefore defined such as follows:

(5.27) $M = \begin{pmatrix} R^{-1} & -R^{-1}T \\ 0 & 1 \end{pmatrix}$

Since R is a rotation matrix and therefore orthogonal, the inverse matrix is the same as its transpose, leading to the following:

(5.28) $$M = \begin{pmatrix} R^T & -R^T T \\ 0 & 1 \end{pmatrix}$$

5.3.3.3 Combining Extrinsic and Intrinsic Camera Parameters

Suppose a 3D point X in space is seen from the world and camera frames as X^w and X^c with the following homogeneous coordinates:

(5.29) $$X^w = \begin{pmatrix} x^w \\ y^w \\ z^w \\ 1 \end{pmatrix}, \qquad X^c = \begin{pmatrix} x^c \\ y^c \\ z^c \\ 1 \end{pmatrix}$$

By combining the intrinsic and extrinsic camera matrices the coordinates of the 2D image point associated with 3D point X can be derived as it follows:

(5.30) $$\begin{pmatrix} \mathrm{x_p} \\ \mathrm{y_p} \\ 1 \end{pmatrix} = \lambda K \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} M \begin{pmatrix} x^w \\ y^w \\ z^w \\ 1 \end{pmatrix}$$

5.4 Geometry of Two Views

Vision data gathered from multiple cameras looking at the same scene in space can reveal valuable geometric information about the position and pose of the cameras with respect to the scene.

Epipolar geometry is the geometry of stereo vision, when two cameras look at slightly offset views of the same scene. Epipolar geometry imposes a number of geometrical constraints between the 3D points in space and their 2D projection on the camera image plane. Such constraints are essential for depth analysis and image-based camera tracking.

From a geometrical point of view the mathematical model for a moving camera at two locations is essentially the same as two individual cameras at these two locations. Therefore, similar to stereo-imaging, epipolar geometry also applies to a moving monocular camera. The use of epipolar geometry in the camera tracking solution proposed in this work will be fully described in Chapter 6. This section however outlines the principles behind epipolar geometry.

5.4.1 Epipolar Geometric Constraints

Assume point X is a fixed 3D point seen as 2D image points X^{c_1} and X^{c_2} by the cameras C_1 and C_2 (see Figure 5.8 for details). C_1 and C_2 also represent the optical centres of the cameras. The three points X, C_1 and C_2 form a plane referred to as the epipolar plane. The intersection of the epipolar plane with image planes I_1 and I_2 form the epipolar lines l_1 and l_2. Figure 5.9 shows that the image of X^{c_1} is a line on the C_2 image plane. This line is the epipolar line l_2. This is because points X, X' and X'' and generally all points on the line specified by points X and C_1 produce the same 2D image at point X^{c_1}. Therefore a single point on image 1 corresponds to a line on image 2. This analogy applies to the epipolar line l_1 on image 1, which corresponds to single point X^{c_2} on image 2.

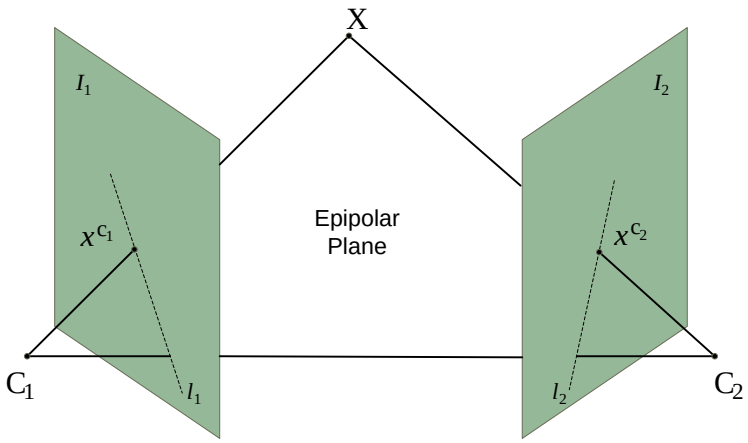


Figure 5.8 Epipolar Geometry

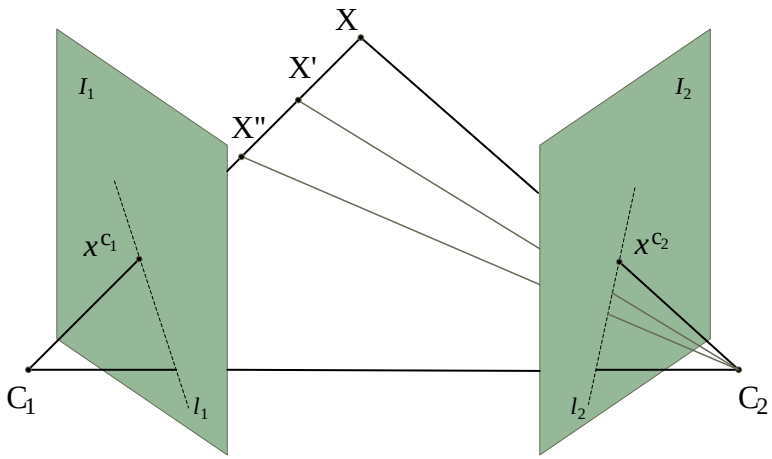


Figure 5.9 Epipolar line

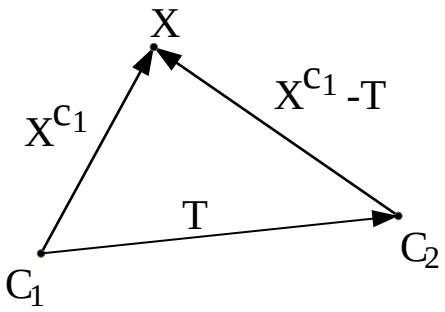


Figure 5.10 Epipolar Plane

Figure 5.10 shows the epipolar plane with X^{c_1} representing the 3D coordinates of point X in camera 1 coordinate system and T the translation vector between the two camera positions. The epipolar plane can be described by its normal vector, which is parallel to the cross product of vectors T and X^{c_1}.

(5.31) $\vec{n} \propto T \otimes X^{c_1}$

Vector $X^{c_1} - T$ is on the epipolar plane and is perpendicular to its normal vector, therefore their dot product is zero:

(5.32) $(X^{c_1} - T) \odot (T \otimes X^{c_1}) = 0$

5.4.2 Essential and Fundamental Matrices

The epipolar constraints described above lead to the definition of the Essential Matrix, which encapsulates the camera extrinsic parameters. This matrix is closely related to another important matrix, referred to as the Fundamental Matrix, which relates corresponding feature points in two images. This section describes these matrices and their role in camera tracking.

Considering vector algebra, the cross product and dot product of two vectors A and B can be expressed as follows:

(5.33) $A \otimes B = \hat{A}B$

(5.34) $A \odot B = A^T B$

$\hat{\mathrm{A}}$ is the skew-symmetric matrix and A^{T} is the transpose of vector A. The skew-symmetric matrix of vector A is defined as in (5.35) and has the property set out in (5.36):

(5.35) $\mathrm{A} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \Rightarrow \hat{A} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & a_1 \\ a_2 & a_1 & 0 \end{pmatrix}$

(5.36) $\hat{A}^T = -\hat{A}$

Using (5.33) and (5.34), equation (5.32) can be re-written as follows:

(5.37) $(X^{c_1} - T) \odot (T \otimes X^{c_1}) = (X^{c_1} - T)^T \hat{T} X^{c_1} = 0$

Using equation (5.27), $(\mathrm{T} - \mathrm{X}^{c_1})$ can be expressed as in (5.38). Substituting this in (5.37) results in (5.39):

(5.38) $X^{c_1} - T = RX^{c_2}$

(5.39) $X^{c_2 T} R^T \hat{T} X^{c_1} = 0$

$\mathrm{R}^{\mathrm{T}}\hat{\mathrm{T}}$ is defined as the Essential Matrix. This matrix plays an important role in two-view camera geometry.

(5.40) $E \triangleq R^T \hat{T} \implies X^{c_2 T} E X^{c_1} = 0$

The camera matrices may be retrieved from the essential matrix up to a scale and four-fold ambiguity. That is there are four possible solutions, except for overall scale, which cannot be determined. It can be shown that a 3×3 matrix can only be an essential matrix if and only if two of its singular values are equal and the third one is zero (Hartley & Zisserman, 2004).

By knowing the essential matrix and applying Singular Value Decomposition (SVD) the rotation matrix can be determined. Suppose the essential matrix is decomposed using the singular value decomposition method:

(5.41) $E = U \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} V^T$

It can be shown that there are four possible factorisations for $\hat{T}$ and R as follows:

(5.42) $S = +UZU^T \; or \; S = -UZU^T$, where $S = \hat{T}$

(5.43) $R = UWV^T or \; R = UW^TV^T$

Matrices Z and W are defined below:

(5.44) $W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

W is an orthogonal and Z a skew-symmetric matrix. Equation (5.43) shows that if the essential matrix is known, by applying singular value decomposition, the only two possible solutions for the rotation matrix can be found.

Equation (5.40) deals with 3D coordinates in calibrated camera view. However the un-calibrated 2D data is often the only available information. As mentioned in section 5.3.3.1 the coordinates of a 3D point in space with respect to the camera frame is related by the camera intrinsic matrix (K), to the corresponding 2D feature point in pixel coordinates. Assuming p^{c_1} and p^{c_2} are the 2D feature points in pixel coordinates, K is applied as follows:

(5.45) $X^{c_1} = K^{-1}p^{c_1}, \quad X^{c_2} = K^{-1}p^{c_2}$

Here the Fundamental Matrix is defined as in (5.46) and equation (5.40) is re-formulated in (5.47):

(5.46) $F \triangleq {K^{-1}}^T E K^{-1}$

(5.47) ${p^{c_2}}^T F p^{c_1} = {p^{c_2}}^T {K^{-1}}^T E K^{-1} p^{c_1} = 0$

There a number of methods used for computing the value of fundamental matrix. A set of at least 8 feature points are required to calculate F. The 8-point algorithm is commonly used for estimating F (Ma, et al., 2004). Once F is known and by knowing matrix K, essential matrix can be calculated using (5.48).

(5.48) $E \triangleq K^T F K$

The essential matrix can then be used to calculate the camera extrinsic parameters up to a scaling factor (see equations (5.42) and (5.43)).

5.5 Summary

This chapter introduces the mathematical models for image formation, three-dimensional representation of moving objects, camera parameters and related geometric constraints. These models are used as the basis for the image-based part of the proposed hybrid pose tracking system presented in chapter 6.

CHAPTER SIX

6 Inertial Visual Hybrid Tracking System Framework

Camera tracking is the process of determining the position and orientation of a camera with respect to a fixed frame of reference, referred to as the world frame. The combination of 3D position and 3D orientation of the camera is referred to as the 6 Degree of Freedom (DOF) camera pose. The applications and current methods for camera tracking have been outlined in previous chapters. In this chapter the proposed method for hybrid inertial-visual tracking is described in detail.

The proposed hybrid-tracking system relies on two types of sensor; namely, an Inertial Measurement Unit (IMU) and an image sensor. The IMU captures acceleration and angular velocity data, while the image sensor supplies pixel information for sets of tracked feature points in the 3D space. The IMU and image sensors provide motion and vision data with respect to their local reference frames, referred to as the IMU and vision reference frames, respectively. Such data need to be transformed to the world frame using rigid-body transformation as explained in section 6.1.1.3.

Both IMU and image sensors are influenced by measurement noise and error, which affect the accuracy of pose estimation. The noise and error cause an IMU-only tracking system to drift significantly over time, making it unsuitable for sole use in pose tracking. On the other hand, camera-only tracking systems not only suffer from noise and measurement error, but also manifest an inherent deficiency, which is the inability to estimate all 6 DOFs. Vision-based tracking systems can provide up to 5 DOFs, meaning that the estimate of the camera

position can only be provided up to a scaling factor. This is due to the fact the image formation in effect is a 3D to 2D transformation, resulting in missing a dimension. This dimension cannot be recovered unless additional information is provided through a 3D model or the presence of an object with known dimensions in the scene. By combining the information gathered through the two types of sensor, the shortcomings of either system can be addressed and a more accurate and robust estimate of the camera pose provided.

The problem of camera-pose tracking is analogous to finding hidden states of a system, defined by the state-space model in the context of recursive Bayesian filtering (see section 3.4.1). In a recursive filter the current state of the system is predicted using the past states and the input signals. The observation data are then used to correct the prediction and provide an estimate of the current system state. In the proposed solution the state space inputs are provided by the IMU and the observation data by the vision-based system.

The observation model is defined based on the properties of Focus of Expansion (FoE) (see section 6.3.3). It has a non-linear relationship to the current system state, which cannot be effectively linearised around the current state. Therefore Kalman Filter (EF) or Extended Kalman Filter (EKF) are not suitable methods for state estimation (see Chapter 3 for a description of these methods). This leads to the use of particle filtering in the proposed system, which is a known solution for recursive filtering for non-linear state-space models.

Particle filtering consists of particle selection based on a proposal distribution, followed by particle evaluation using a likelihood function and finally state estimation based on weighted particles. This process has been described in Chapter 3 and forms the basis of the hybrid pose-tracking system proposed here.

The process of determining optical flow's Focus of Expansion (FoE) provides a measure of the effectiveness of the vision-based system in the overall pose tracking performance. This was used to determine whether or not the vision-based system, at any particular time, provides sufficient information to influence the pose estimate provided by the IMU tracker. These measures have been utilised in developing a mechanism to automatically select the best tracking method from IMU-only, hybrid or hybrid with past state correction at any time (see section 6.4.9). A state correction mechanism has been developed to correct the past state of the system, where new reliable information becomes available (see section 6.4.8).

As described in Chapter 5, the camera orientation can be estimated by the vision-based system using the properties of the essential matrix (see section 6.3.4). Therefore in order to reduce the complexity and computational cost of particle filtering-based state estimation, the camera orientation is removed from the state-space vector and instead is estimated solely by the vision-based system. Consequently the state-space vector is defined by the camera pose and linear velocity.

Section 6.1 of this chapter first provides an overview of the proposed system. Section 6.2 outlines the main tasks of the IMU-based system in state estimation. Section 6.3 describes the actions the vision-based system must undertake. This mainly includes image capture, feature detection, feature tracking, estimation of rotation matrix and providing FoE data. Finally data fusion and state estimation have been detailed in section 6.4.

6.1 System Overview

The proposed solution has been developed by making use of particle filtering, where particles are the best candidates for the system state at any particular time. The overall system consists of an inertial-based, a vision-based, and a Stochastic Data Fusion (SDF) component. Figure 3.2 illustrates a block diagram of the system architecture.

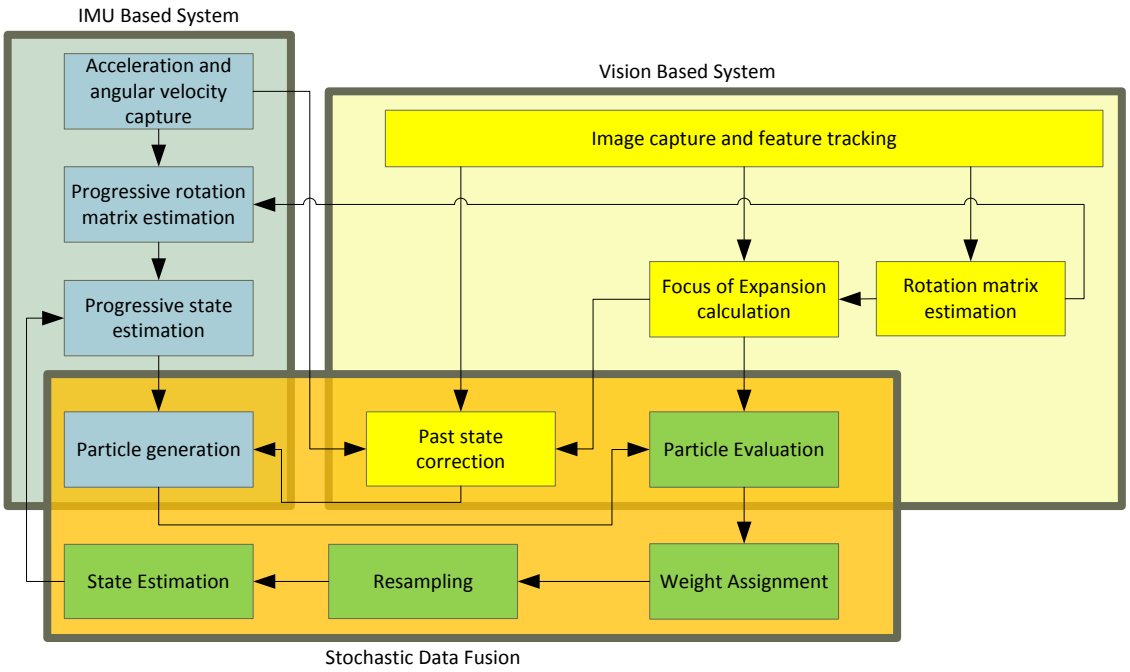


Figure 6.1 System Block Diagram

This block diagram shows that the SDF has overlapping functional blocks with the IMU and vision based systems. This is due to the fact that the processes of particle selection and evaluation are significantly dependant on the IMU and vision data, respectively.

A typical IMU may operate at a 100Hz sampling rate or more, whereas the vision-based system generally samples at 50Hz or less. The essence of the system proposed here is to use IMU data for camera pose estimation while there is no image data and combine IMU and vision data, whenever a new image is captured.

6.1.1 Definitions and Assumptions

In order to effectively formulate all physical and geometric models used for creating the framework presented here, certain definitions and notations have been adopted throughout this chapter as follows.

6.1.1.1 Principal Coordinate Systems

The camera tracking process starts from a known camera pose with respect to the world frame, meaning that there is a known transformation matrix between the camera frame at the initial camera pose and the world frame. Consequently if the camera pose at a future time is determined with respect to the initial camera reference frame, such pose can easily be transformed to the pose with respect to the world frame via a known fixed transformation matrix. The world frame is assumed to have its Z axis in opposite direction to the gravity vector. Figure 6.2 shows the camera reference frames C_t at time t and C_0 at time $t = 0$. In this work all motion and geometry data are with respect to one of the three principal coordinate systems; namely, world, camera and IMU reference frames.

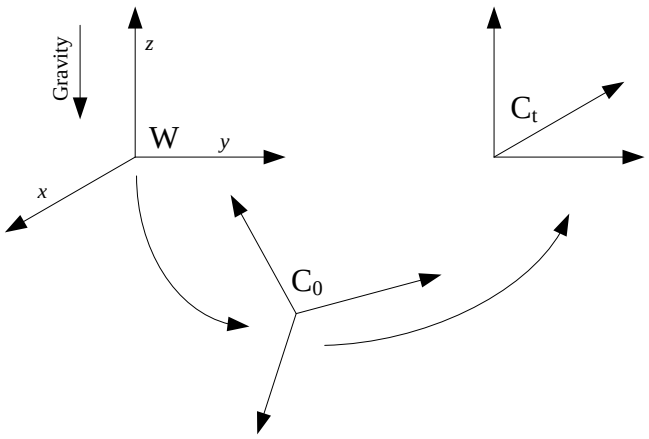


Figure 6.2 World Frame vs Camera Frame

6.1.1.2 Reference Frames

A point Q in 3D space is seen by each reference frame as a different coordinate vector, Q^r, where r may be a letter superscript from the set of letters below.

(6.1) $\boldsymbol{r} = \{\boldsymbol{c}, \boldsymbol{i}, \boldsymbol{k}, \boldsymbol{p}, \boldsymbol{w}\}$

Letters c, i, k, p, w refer to reference frames associated with the current camera pose, IMU, key frame, previous camera pose, and world, respectively. Figure 6.3 shows point Q as seen in the world and camera frames.

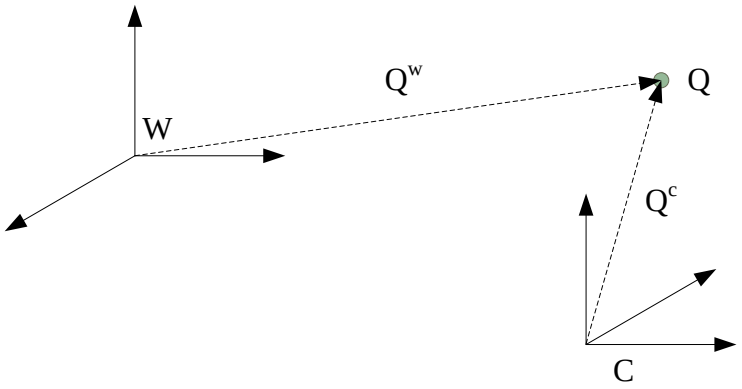


Figure 6.3 Point $\boldsymbol{Q}$ Seen from the World and Camera Frames

The transformation matrix converting a coordinate vector from reference frame r_1 to reference frame r_2 is similarly defined by $G^{r_2 r_1}$. r_1 and r_2 can be one of the letters defined in (6.1). The rigid body transformation is explained in detail in section 6.1.1.3.

6.1.1.3 Rigid Body Transformation

The camera and the associated IMU motion are considered in the context of rigid-body motion. The motion trajectory of a rigid body with respect to a fixed reference frame can be fully described by the motion trajectory of a single point on the rigid body. The motion trajectory of other points on the rigid body can be determined by knowing the relative location of each point to a reference point

on the body. Here, without loss of generality, it is assumed that the camera frame coincides with the moving body frame.

In order to estimate the camera pose, the camera needs to view a number of fixed points in the 3D space. While the camera is moving the 2D projection of these points on the image plane will also be moving, helping the tracking system to recover the pose of the camera itself.

Let us define $Q^w = (x^w, y^w, z^w)^T$ and $Q^c = (x^c, y^c, z^c)^T$ as the 3D coordinate vectors of a point Q in 3D space with respect to world and camera frames. These coordinate vectors hold the following relationship, where R^{wc} and T^{wc} are the rotation matrix and translation vector between the two reference frames.

(6.2) $\boldsymbol{Q^w = R^{wc} Q^c + T^{wc}}$

T^{wc} is the coordinate vector of the origin of the body frame in the world frame. R^{wc} is a 3x3 matrix whose columns are the unit vectors of the camera frame in the world frame.

(6.3) $\boldsymbol{T^{wc} = X_c^w}$

(6.4) $\boldsymbol{R^{wc} = (i_c^w \quad j_c^w \quad k_c^w)}$

C is the origin of camera and $\mathrm{i_c^w}$, $\mathrm{j_c^w}$ and $\mathrm{k_c^w}$ are the unit vectors of the camera frame with respect to the world frame. In order to concatenate the rotation matrix and translation vector into a single matrix, the coordinate vectors are defined in homogeneous coordinate system as follows:

(6.5) $$\boldsymbol{Q^w = \begin{pmatrix} x^w \\ y^w \\ z^w \\ 1 \end{pmatrix}}$$

(6.6) $$\boldsymbol{Q^c = \begin{pmatrix} x^c \\ y^c \\ z^c \\ 1 \end{pmatrix}}$$

(6.7) $\boldsymbol{Q^w = G^{wc} Q^c}$

(6.8) $\boldsymbol{G^{wc}} = \begin{pmatrix} \boldsymbol{R^{wc}} & \boldsymbol{T^{wc}} \\ \overline{\boldsymbol{0}}_{\boldsymbol{1x3}} & \boldsymbol{1} \end{pmatrix}$

Where G^{wc} is a 4x4 transformation matrix, converting a homogeneous point in the camera frame to the world frame. R^{wc} is an orthogonal matrix, meaning that:

(6.9) $\boldsymbol{R^{wc}R^{wcT}} = \boldsymbol{I}$

(6.10) $\boldsymbol{R^{wc-1}} = \boldsymbol{R^{wcT}}$

The conversion from the world frame to the body frame is the inverse of the above transformation matrix:

(6.11) $\boldsymbol{Q^{c}} = \boldsymbol{G^{cw}Q^{w}}$

(6.12) $\boldsymbol{G^{cw}} = \boldsymbol{G^{wc-1}} = \begin{pmatrix} \boldsymbol{R^{wcT}} & \boldsymbol{-R^{wcT}T^{wc}} \\ \overline{\boldsymbol{0}}_{\boldsymbol{1x3}} & \boldsymbol{1} \end{pmatrix}$

The camera tracking system presented here provides an estimate for T^{wc} and R^{wc} when a new image becomes available. Matrices G^{wc} and G^{cw} are derived from equations (6.8) and (6.12) and are used for converting local motion information to the world frame and vice versa. The transformation matrix between any two camera views r_2 and r_1 can be calculated as follows:

(6.13) $\boldsymbol{G^{wr_1}} = \boldsymbol{G^{wr_2}G^{r_2r_1}} \quad \Longrightarrow$

(6.14) $\boldsymbol{G^{r_2r_1}} = \boldsymbol{G^{wr_2-1}G^{wr_1}} = \boldsymbol{G^{r_2w}G^{wr_1}}$

In general the reference frames of the IMU and camera might not be the same. However as they are both parts of the camera rigid body, their relationship can be described by a constant transformation matrix, G^{ci}.

(6.15) $\boldsymbol{Q^{c}} = \boldsymbol{G^{ci}Q^{i}}$

Q^c and Q^i are the homogeneous coordinates of point Q in the camera and IMU frames, respectively. G^{ci} is assumed known with pre-determined values. Therefore in order to convert the motion information in the IMU frame to the world frame, G^{wi} matrix as defined below is used.

(6.16) $\boldsymbol{Q^w = G^{wi} Q^i}$

(6.17) $\boldsymbol{G^{wi} = G^{wc} G^{ci}}$

6.1.1.4 Vector Transformation

Suppose vector H is defined by two points, Q_1 and Q_2 in 3D space.

(6.18) $\boldsymbol{H = Q_2 - Q_1}$

This vector is seen as H^w and H^c in the world and camera reference frames,

(6.19) $\boldsymbol{H^w = Q_2^w - Q_1^w,\ H^c = Q_2^c - Q_1^c}$

Applying the camera-to-world frame transformation yields:

(6.20) $\boldsymbol{H^w = Q_2^w - Q_1^w = R^{wc} Q_2^c + T^{wc} - (R^{wc} Q_1^c + T^{wc}) \implies}$

(6.21) $\boldsymbol{H^w = R^{wc}(Q_2^c - Q_1^c) = R^{wc} H^c}$

Consequently the translation vector has no effect on vector transformation from one reference frame to another.

6.2 Inertial-Based System

The inertial-based system is mainly used for generating particles. However the tracker also needs to provide an estimate of the system states between two consecutive images, where IMU data is the only available source of motion data. This is done using a state-space model. In this section the kinematic motion equations for the camera are described and an estimate of the camera pose based on the IMU data is provided.

6.2.1 Motion Equations

The IMU incorporates an accelerometer and a gyroscope. Both devices output data with respect to the IMU local reference frame which, as explained in section 6.1.1.3, can be transformed to the camera reference frame using G^{ci} transformation matrix. Referring to the concept of motion kinematics, an external force F on an object with mass m, creates an acceleration $\mathrm{a} = \mathrm{F}/\mathrm{m}$. The acceleration is the second derivative of object displacement, x. The first derivative of the object displacement is called velocity, referred to as v.

(6.22) $\boldsymbol{a}(\boldsymbol{t}) = \ddot{\boldsymbol{x}}(\boldsymbol{t})$

(6.23) $\boldsymbol{v}(\boldsymbol{t}) = \dot{\boldsymbol{x}}(\boldsymbol{t})$

This leads to the following equations for deriving velocity and displacement using integration of the acceleration over the time interval $[\mathrm{t}_1 \quad \mathrm{t}]$:

(6.24) $\boldsymbol{v}(\boldsymbol{t}) = \int_{\boldsymbol{t_1}}^{\boldsymbol{t}} \boldsymbol{a}(\boldsymbol{t})\boldsymbol{dt} + \boldsymbol{v}(\boldsymbol{t_1})$

(6.25) $\boldsymbol{x}(\boldsymbol{t}) = \int_{\boldsymbol{t_1}}^{\boldsymbol{t}} \boldsymbol{v}(\boldsymbol{t})\boldsymbol{dt} + \boldsymbol{x}(\boldsymbol{t_1})$

For a linear motion with constant acceleration, the displacement and velocity take a simple form as follows

(6.26) $\boldsymbol{v}(\boldsymbol{t}) = \boldsymbol{v}(\boldsymbol{t_1}) + \boldsymbol{a}.(\boldsymbol{t} - \boldsymbol{t_1})$

(6.27) $\boldsymbol{x}(\boldsymbol{t}) = \boldsymbol{x}(\boldsymbol{t_1}) + \boldsymbol{v}(\boldsymbol{t}).(\boldsymbol{t} - \boldsymbol{t_1}) + \frac{\boldsymbol{1}}{\boldsymbol{2}}\boldsymbol{a}.(\boldsymbol{t} - \boldsymbol{t_1})^{\boldsymbol{2}}$

In real-life applications, it is unlikely to have a constant acceleration. Therefore in order to simplify the equations (6.24) and (6.25), the motion can be approximated by a constant acceleration between two sufficiently close times, t_1 and t_2, with time difference $\delta\mathrm{t}$. Therefore during this period the linear equations (6.26) and (6.27) apply. This process is referred to as piecewise linearisation.

(6.28) $\boldsymbol{v(t+\delta t) = v(t) + a(t).\delta t}$

(6.29) $\boldsymbol{x(t+\delta t) = x(t) + v(t).\delta t + \frac{1}{2}a(t).\delta t^2}$

In a system with an integrated IMU, the value of a is sampled at regular sampling times. In such a system δt is the sampling interval between two consecutive IMU sampling times and the constant value of a can simply be approximated by the acceleration at time t. The accuracy of this approximation however depends on the sampling interval being sufficiently small so that variations of acceleration during the interval have a negligible effect on the overall accuracy of the system. Equations (6.28) and (6.29) are the basis for state space representation of kinematic motion (see section 3.4.1 for details).

Similar analogy applies to the angular displacement, where the angular velocity is the first derivative of angular displacement.

(6.30) $\boldsymbol{\omega(t) = \dot{\theta}(t)}$

The gyroscope integrated into the IMU supplies the angular velocity data at regular time intervalsδt. If piecewise linearisation is applied, equation (6.30) for short time intervals can be written as follows.

(6.31) $\boldsymbol{\theta(t+\delta t) = \theta(t) + \omega(t).\delta t}$

Equations (6.28), (6.29) and (6.31) apply to all three directions of motion along and around x, y and z axes as follows:

(6.32) $\boldsymbol{X^w_{t+\delta t} = X^w_t + V^w_t \delta t + \frac{1}{2}A^w_t \delta t^2}$

(6.33) $\boldsymbol{V^w_{t+\delta t} = V^w_t + A^w_t \delta t}$

(6.34) $\boldsymbol{\Theta^w_{t+\delta t} = \Theta^w_t + \Omega^w_t \delta t}$

In this motion model X, V, A, Θ and Ω are the 3D linear displacement, velocity, acceleration, angular displacement and angular velocity vectors, respectively. This is the set of kinematic motion equations with respect to the world frame,

which is used for estimating the new states of the system, $(X^w_{t+\delta t}, V^w_{t+\delta t}, \Theta^w_{t+\delta t})$, by knowing the past states $(X^w_t, V^w_t, \Theta^w_t)$ as well as the control inputs (A^w_t, Ω^w_t). X^w_t, V^w_t and Θ^w_t are the 3D position, linear velocity and orientation of the camera with respect to the world frame. A^w_t, Ω^w_t are the current acceleration and angular velocity vectors.

6.2.1.1 Effect of Rotation Matrix

The IMU data need to be transformed to the world frame before they can be used in the above kinematic motion model. Since the acceleration is a vector, the translation vector has no effect on its transformation (see section 6.1.1.4 for more details). Therefore only the orientation of the IMU affects the conversion from the IMU frame to the world frame. This rotation is represented by matrix R^{wi}, which is linked to the camera reference frame as per (6.17).

(6.35) $\boldsymbol{A^w_t = R^{wi} A^i_t}$

(6.36) $\boldsymbol{R^{wi} = R^{wc} R^{ci}}$

Substituting (6.17) into (6.32) and (6.33) results in:

(6.37) $\boldsymbol{X^w_{t+\delta t} = X^w_t + V^w_t \delta t + \frac{1}{2} R^{wi} A^i_t \delta t^2}$

(6.38) $\boldsymbol{V^w_{t+\delta t} = V^w_t + R^{wi} A^i_t \delta t}$

Converting the orientation is somewhat less straightforward. Here, instead of using the orientation vector, the rotation matrix and its incremental change must be used. If the current orientation of the IMU is represented by matrix, R^{wi}_t, the new orientation by matrix $R^{wi}_{t+\delta t}$, and the differential rotation between the two by $R_{t,\delta t}$, these three matrices hold the following relationship:

(6.39) $\boldsymbol{R^{wi}_{t+\delta t} = R_{t,\delta t} R^{wi}_t}$

Due to the small change of orientation between two successive IMU samples, matrix $R_{\delta t}$ can be approximated using the incremental orientation vector,

consisting of three differential elements, $\delta\phi, \delta\theta,$ and $\delta\psi$, between the two states.

(6.40) $$\begin{pmatrix} \boldsymbol{\delta\phi} \\ \boldsymbol{\delta\theta} \\ \boldsymbol{\delta\psi} \end{pmatrix} = \boldsymbol{\Delta\Theta_t^w} = \boldsymbol{\Omega_t^w \delta t}$$

(6.41)

$$\boldsymbol{R_{t,\delta t}} = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{cos(\delta\phi)} & \boldsymbol{-sin(\delta\phi)} \\ \mathbf{0} & \boldsymbol{sin(\delta\phi)} & \boldsymbol{cos(\delta\phi)} \end{pmatrix} \begin{pmatrix} \boldsymbol{cos(\delta\theta)} & \mathbf{0} & \boldsymbol{sin(\delta\theta)} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \boldsymbol{-sin(\delta\theta)} & \mathbf{0} & \boldsymbol{cos(\delta\theta)} \end{pmatrix} \begin{pmatrix} \boldsymbol{cos(\delta\psi)} & \boldsymbol{-sin(\delta\psi)} & \mathbf{0} \\ \boldsymbol{sin(\delta\psi)} & \boldsymbol{cos(\delta\psi)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

The camera orientation can be found directly from the rotation matrix by knowing the sequence of rotation. Therefore the focus in this work is on finding the rotation matrix only. The rotation matrix between two consecutive images is estimated using tracked feature points (see section 6.3.4). In between two images, the rotation matrix is estimated recursively using equations (6.39) to (6.41).

6.2.1.2 The Effect of Gravity

The accelerometer measures proper acceleration, i.e. acceleration without gravity. For example if the camera system is at rest on a desktop with gravity facing downwards, the desk exerts an upward force equal to gravity to keep the camera system still on the desk; therefore the accelerometer measures the upward force only. On the other hand if the camera system is experiencing a free fall, the accelerometer shows zero, as the whole body of the accelerometer (case and proof mass) move together; hence no displacement between the two will be observed by the sensor. Gravity has no effect on the orientation. Equations (6.42) to (6.44) provide the complete Kinematics motion model for the camera pose after inclusion of gravity effect.

(6.42) $$\boldsymbol{X_{t+\delta t}^w} = \boldsymbol{X_t^w} + \boldsymbol{V_t^w \delta t} + \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{R_t^{wi} A_t^i} + \boldsymbol{G})\boldsymbol{\delta t^2}$$

(6.43) $$\boldsymbol{V_{t+\delta t}^w} = \boldsymbol{V_t^w} + (\boldsymbol{R_t^{wi} A_t^i} + \boldsymbol{G})\boldsymbol{\delta t}$$

(6.44) $$\boldsymbol{R_{t+\delta t}^{wi}} = \boldsymbol{R_{t,\delta t} R_t^{wi}}$$

Vector G is a constant vector, and without loss of generality it is assumed that the two components of G are zero and the last component in the Z direction is g, typically $9.81\ ms^{-2}$. These motion equations are used for estimating camera pose using an IMU as detailed in the next section and then for generating particles as detailed in section 6.4.3.

6.2.2 IMU-Based Pose Estimation

Between the time of the current image (t_n) and the previous one (t_{n-1}), no vision data is available; however the IMU still continues to gather motion data. Therefore between t_{n-1} and t_n the pose is estimated using the IMU data and the Kinematics motion equations (6.42) to (6.44). Suppose N_i is the number of IMU data sets in the time interval $[t_{n-1}, t_n]$, with δt time between two IMU samples. Using $X^w_{t_{n-1}}$, $V^w_{t_{n-1}}$ and $R^{wi}_{t_{n-1}}$ as the initial position vector, and velocity vector, and the rotation matrix at time t_{n-1} the pose at $t_{n-1}+\delta t$, $t_{n-1}+2\delta t$ and $t_{n-1}+3\delta t$ are calculated and then by the application of mathematical induction the pose at time $t_n = t_{n-1} + N_i\delta t$ is formulated. Equations (6.45) to (6.47) give the system states at time $t_{n-1}+\delta t$ and $t_{n-1}+2\delta t$, respectively.

(6.45)
$$\begin{pmatrix} \boldsymbol{X^w_{t_{n-1}+\delta t}} \\ \boldsymbol{V^w_{t_{n-1}+\delta t}} \\ \boldsymbol{R^{wi}_{t_{n-1}+\delta t}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X^w_{t_{n-1}} + V^w_{t_{n-1}}\delta t + \frac{\delta t^2}{2}\left(R^{wi}_{t_{n-1}} A^i_{t_{n-1}} + G\right)} \\ \boldsymbol{V^w_{t_{n-1}} + \delta t\left(R^{wi}_{t_{n-1}} A^i_{t_{n-1}} + G\right)} \\ \boldsymbol{\delta R_{t_{n-1}} R^{wi}_{t_{n-1}}} \end{pmatrix}$$

(6.46)
$$\begin{pmatrix} \boldsymbol{X^w_{t_{n-1}+2\delta t}} \\ \boldsymbol{V^w_{t_{n-1}+2\delta t}} \\ \boldsymbol{R^{wi}_{t_{n-1}+2\delta t}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X^w_{t_{n-1}+\delta t} + V^w_{t_{n-1}+\delta t}\delta t + \frac{\delta t^2}{2}\left(R^{wi}_{t_{n-1}+\delta t} A^i_{t_{n-1}+\delta t} + G\right)} \\ \boldsymbol{V^w_{t_{n-1}+\delta t} + \delta t\left(R^{wi}_{t_{n-1}+\delta t} A^i_{t_{n-1}+\delta t} + G\right)} \\ \boldsymbol{\delta R_{t_{n-1}+\delta t} R^{wi}_{t_{n-1}+\delta t}} \end{pmatrix}$$

δR_t is the differential rotation matrix from time t and $t+\delta t$. Substituting (6.45) into (6.46) results in the simpler expression below:

(6.47) $$\begin{pmatrix} \boldsymbol{X^{w}_{t_{n-1}+2\delta t}} \\ \boldsymbol{V^{w}_{t_{n-1}+2\delta t}} \\ \boldsymbol{R^{wi}_{t_{n-1}+2\delta t}} \end{pmatrix} =$$

$$\begin{pmatrix} \boldsymbol{X^{w}_{t_{n-1}} + 2V^{w}_{t_{n-1}}\delta t + \frac{\delta t^2}{2}\sum_{m=1}^{2}(5-2m)\left(R^{wi}_{t_{n-1}+m\delta t}\, A^{i}_{t_{n-1}+m\delta t} + G\right)} \\ \boldsymbol{V^{w}_{t_{n-1}} + \delta t\sum_{m=1}^{2}\left(R^{wi}_{t_{n-1}+m\delta t}\, A^{i}_{t_{n-1}+m\delta t} + G\right)} \\ \boldsymbol{(\prod_{m=1}^{2}\delta R_{t_{n-1}+m\delta t})R^{wi}_{t_{n-1}}} \end{pmatrix}$$

This process is repeated for time $\mathrm{t_{n-1}} + 3\delta\mathrm{t}$, with the outcome summarised in equation (6.48).

(6.48) $$\begin{pmatrix} \boldsymbol{X^{w}_{t_{n-1}+3\delta t}} \\ \boldsymbol{V^{w}_{t_{n-1}+3\delta t}} \\ \boldsymbol{R^{wi}_{t_{n-1}+3\delta t}} \end{pmatrix} =$$

$$\begin{pmatrix} \boldsymbol{X^{w}_{t_{n-1}} + 3V^{w}_{t_{n-1}}\delta t + \frac{\delta t^2}{2}\sum_{m=1}^{3}(7-2m)\left(R^{wi}_{t_{n-1}+m\delta t}\, A^{i}_{t_{n-1}+m\delta t} + G\right)} \\ \boldsymbol{V^{w}_{t_{n-1}} + \delta t\sum_{m=1}^{3}\left(R^{wi}_{t_{n-1}+m\delta t}\, A^{i}_{t_{n-1}+m\delta t} + G\right)} \\ \boldsymbol{(\prod_{m=1}^{3}\delta R_{t_{n-1}+m\delta t})R^{wi}_{t_{n-1}}} \end{pmatrix}$$

This process is repeated until all N_i IMU data sets are considered. By applying mathematical induction the result at time following $\boldsymbol{t_n = t_{n-1} + N_i\delta t}$ is:

$$(6.49)\begin{pmatrix} \boldsymbol{X}^{\boldsymbol{w}}_{\boldsymbol{t_{n-1}+N_i\delta t}} \\ \boldsymbol{V}^{\boldsymbol{w}}_{\boldsymbol{t_{n-1}+N_i\delta t}} \\ \boldsymbol{R}^{\boldsymbol{wi}}_{\boldsymbol{t_{n-1}+N_i\delta t}} \end{pmatrix} =$$

$$\begin{pmatrix} \boldsymbol{X}^{\boldsymbol{w}}_{\boldsymbol{t_{n-1}}} + \boldsymbol{N_i V}^{\boldsymbol{w}}_{\boldsymbol{t_{n-1}}}\boldsymbol{\delta t} + \frac{\boldsymbol{\delta t^2}}{\boldsymbol{2}}\sum_{\boldsymbol{m=1}}^{\boldsymbol{N_i}}(\boldsymbol{2N_i + 1 - 2m})\left(\boldsymbol{R}^{\boldsymbol{wi}}_{\boldsymbol{t_{n-1}+m\delta t}}\boldsymbol{A}^{\boldsymbol{i}}_{\boldsymbol{t_{n-1}+m\delta t}} + \boldsymbol{G}\right) \\ \boldsymbol{V}^{\boldsymbol{w}}_{\boldsymbol{t_{n-1}}} + \boldsymbol{\delta t}\sum_{\boldsymbol{m=1}}^{\boldsymbol{N_i}}\left(\boldsymbol{R}^{\boldsymbol{wi}}_{\boldsymbol{t_{n-1}+m\delta t}}\boldsymbol{A}^{\boldsymbol{i}}_{\boldsymbol{t_{n-1}+m\delta t}} + \boldsymbol{G}\right) \\ \left(\prod_{\boldsymbol{m=1}}^{\boldsymbol{N_i}}\boldsymbol{\delta R}_{\boldsymbol{t_{n-1}+m\delta t}}\right)\boldsymbol{R}^{\boldsymbol{wi}}_{\boldsymbol{t_{n-1}}} \end{pmatrix}$$

In this equation $\mathrm{A}^{\mathrm{i}}_{\mathrm{t_{n-1}+m\delta t}}$ and $\mathrm{R}_{\mathrm{t_{n-1}+m\delta t,\delta t}}$ are the control inputs. $\mathrm{A}^{\mathrm{i}}_{\mathrm{t_{n-1}+m\delta t}}$ is the 3D acceleration measured by the IMU. $\mathrm{R}_{\mathrm{t_{n-1}+m\delta t,\delta t}}$ is the differential rotation matrix, which is directly calculated using (6.41) and the 3D angular velocity measured by the gyroscope at time $\mathrm{t_{n-1}} + \mathrm{m\delta t}$. The initial states of the system, $\mathrm{X}^{\mathrm{w}}_{\mathrm{t_{n-1}}}$, $\mathrm{V}^{\mathrm{w}}_{\mathrm{t_{n-1}}}$ and $\mathrm{R}^{\mathrm{wi}}_{\mathrm{t_{n-1}}}$, are the estimated system states provided by hybrid tracking at time $\mathrm{t_{n-1}}$.

This formulation has been presented in a way to suit the particle generation process, which will be explained later on in section 6.4.3. However for applications where the camera pose between two consecutive images is required, the following equations can be used for state progression, where m is a number from 1 to N_i.

$$(6.50)\begin{pmatrix} \boldsymbol{X}^{\boldsymbol{w}}_{\boldsymbol{t_{n-1}+m\delta t}} \\ \boldsymbol{V}^{\boldsymbol{w}}_{\boldsymbol{t_{n-1}+m\delta t}} \\ \boldsymbol{R}^{\boldsymbol{wi}}_{\boldsymbol{t_{n-1}+m\delta t}} \end{pmatrix} =$$

$$\begin{pmatrix} \boldsymbol{X}^{\boldsymbol{w}}_{\boldsymbol{t_{n-1}+(m-1)\delta t}} + \boldsymbol{V}^{\boldsymbol{w}}_{\boldsymbol{t_{n-1}+(m-1)\delta t}}\boldsymbol{\delta t} + \frac{\boldsymbol{\delta t^2}}{\boldsymbol{2}}\left(\boldsymbol{R}^{\boldsymbol{wi}}_{\boldsymbol{t_{n-1}+(m-1)\delta t}}\boldsymbol{A}^{\boldsymbol{i}}_{\boldsymbol{t_{n-1}+(m-1)\delta t}} + \boldsymbol{G}\right) \\ \boldsymbol{V}^{\boldsymbol{w}}_{\boldsymbol{t_{n-1}+\delta t}} + \boldsymbol{\delta t}\left(\boldsymbol{R}^{\boldsymbol{wi}}_{\boldsymbol{t_{n-1}+(m-1)\delta t}}\boldsymbol{A}^{\boldsymbol{i}}_{\boldsymbol{t_{n-1}+(m-1)\delta t}} + \boldsymbol{G}\right) \\ \boldsymbol{\delta R}_{\boldsymbol{t_{n-1}+(m-1)\delta t}}\boldsymbol{R}^{\boldsymbol{wi}}_{\boldsymbol{t_{n-1}+(m-1)\delta t}} \end{pmatrix}$$

6.3 The Vision-Based System

The vision based system is responsible for capturing images of the scene and processing image data to assist the pose tracking process. The core of the vision-based system is feature detection, tracking and analysis. A feature point is the 2D image of a visually distinctive 3D point in space, which is formed on the image plane based on the pin-hole camera principle (for more information refer to Chapter 5). The feature points are used to derive the rotation matrix when a new image is captured. The feature points and tracking information are also used to form the observation model in the context of state space and recursive filtering.

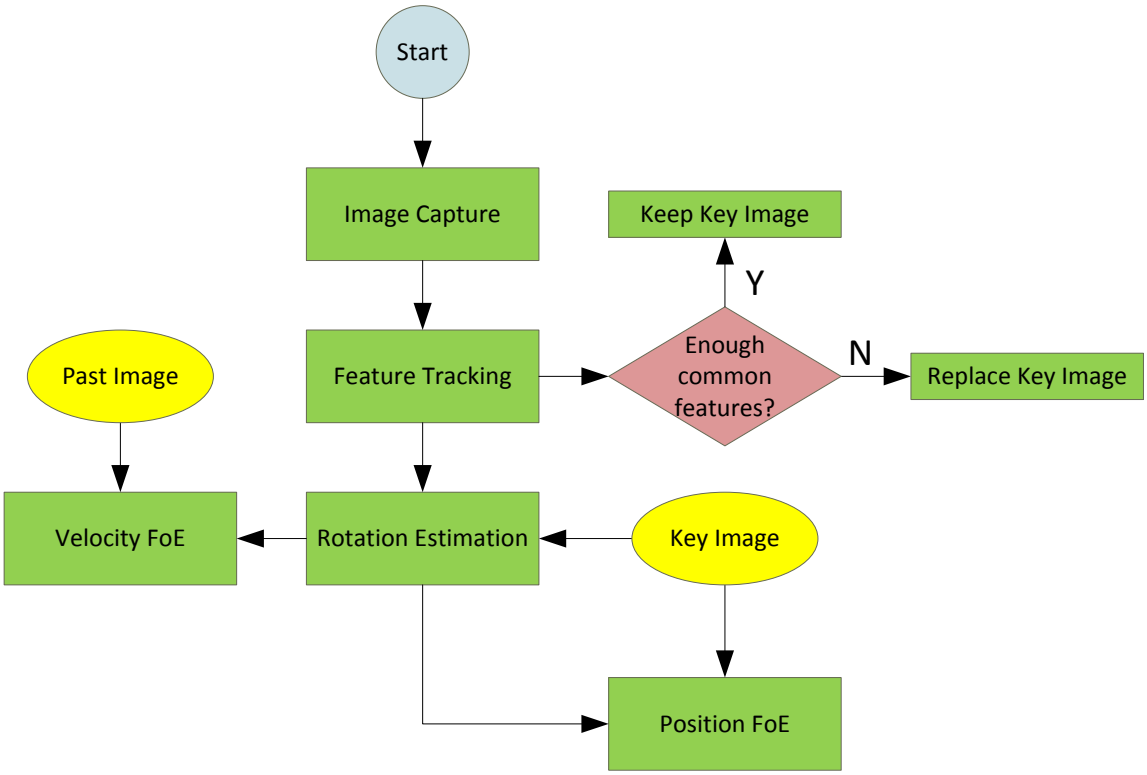


Figure 6.4 The Vision-Based System

Figure 6.4 provides a block diagram of the vision-based system. The following section describes in detail the process of image capture, feature detection and tracking. It also provides an overview of the method used for calculating the rotation matrix. In addition, the concept of Focus of Expansion (FoE), which has

extensively been used in the proposed hybrid tracking system, is described in detail. In the last section the criteria for replacing the key image and also a mechanism for replacing the key image when needed is outlined.

6.3.1 Image Capture

The system requires a monocular camera to capture images of the scene at regular intervals. The tracking algorithm needs three images to operate; namely current, past and key images. Upon system start up an image is captured and considered to be the first key image. The feature points are then detected and used as the basis for the tracking algorithm. Thereafter a new image is obtained at every image sampling time and the associated feature points are detected and tracked.

The key image plays an important role in the proposed tracking system. It acts as a reference for tracking and remains unchanged while there are adequate shared feature points between the current and key images. Here is an outline summary of the image capture process;

- At time t_1 the first image (I_{t_1}) is captured and used as the key image (I_{t_k}). As this is the first key image, t_1 and t_k are the same. However during tracking the key image may need changing (see section 6.3.5). Therefore the key image is referred to as I_{t_k}, which takes into account the occasional change of key image.
- A set of feature points, F_k, is found in the key image and used as the basis for optical-flow tracking. A minimum of 12 feature points is recommended in order to provide enough feature points for the application of the 8-point algorithm, without having to frequently change the key image.
- At time t_2 another image is captured (I_2) and image features are tracked and gathered in a new set of feature points, called F_2.

- Thereafter every new image (I_n) goes through the feature tracking process and the detected feature points are saved in F_k. The number of tracked feature points in the new image is used to decide, whether or not the key image must be replaced by a new image, which shares more feature points with the current image. Section 6.3.5 provides the details of when and how the key image must be replaced.

6.3.2 Feature Tracking

The proposed system uses a SIFT feature detector to detect feature points in the key image. In order to minimise drift in the tracking system, it is best to keep the key image unchanged as long as practically possible. To meet this objective, feature points towards the centre of the image are considered for tracking as they are more likely to remain in the subsequent images. Therefore a Region of Interest (ROI) is defined for the key image, which concentrates on the central area of the image.

If the images are obtained by a wide-angle camera, they need to be rectified before being used for feature detection. However this process leaves parts of the image close to the borders blurred, making them less suitable for this algorithm. Therefore the use of central ROI is also beneficial in this case as it discards the blurred regions of the image.

The features detected in the key image must be tracked so that corresponding feature points in subsequent images can be found. Optical flow tracking is a technique widely used for this purpose. There are various methods for tracking optical flow, however the Kanade-Lucas-Tomasi (KLT) is one of the most widely used and is employed in this work for tracking features points. This method works based on searching for feature points in the vicinity of each feature point in the first image and finding the corresponding feature points in the second image. A complete account of this method is outside the scope of this work,

however the readers are invited to refer to (Lucas & Kanade, 1981) for more information.

6.3.3 Focus of Expansion

Focus of Expansion (FoE) is a concept used in conjunction with optical flow tracking. In this section three definitions in relation to optical flow tracking are explained. Firstly the ‘flow line’ is defined as the line connecting 2D images of a point in 3D space, as seen from two camera viewpoints. Figure 6.5 shows the flow lines in red for a static scene viewed by a moving camera. The flow lines have been identified in the Region of Interest (RoI) specified by a red rectangle.

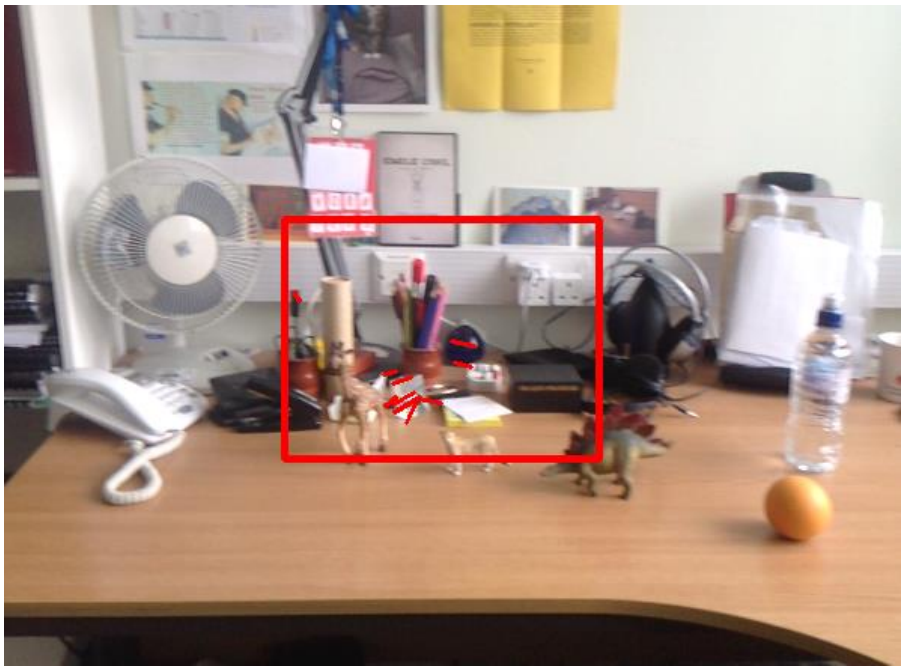


Figure 6.5 Flow Lines

Secondly the ‘flow velocity vector’ is defined as the 2D velocity vector of a feature point on the image plane. The 2D image of a static point in space, moving with the movement of the camera, constitutes a velocity vector. For two adjacent images, the image velocity vector can be approximated by the feature point displacement vector divided by the time elapsed between the two images.

The third and most important concept is the Focus of Expansion (FoE). Focus of expansion is closely linked to optical flow and refers to a point on the image plane where all flow lines meet. In order to understand the concept of focus of expansion, it is important to recall one of the principles of perspective geometry, which states that images of parallel lines meet at a single point in the image plane. This point is referred to as the vanishing point. Figure 6.6 illustrates the vanishing point for a perspective view of parallel lines.

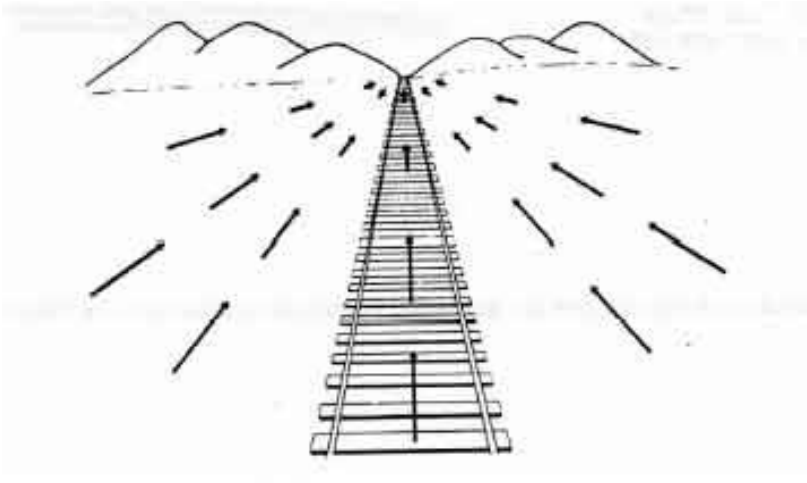


Figure 6.6 Vanishing Point

The same analogy applies to a moving camera. If a camera undergoes a translational movement, T, from the camera point of view it appears that the static 3D points all move parallel to each other in the opposite direction by translation vector $-T$. The relative movement of each feature point with respect to the camera frame forms a flow line. Referring to the concept of vanishing point explained above, since the movements of all feature points in 3D space with respect to the camera frame are parallel, their images on the image plane must meet at a single point (except for some cases as explained in section 6.3.3.3.1). The images of parallel lines are the flow lines, and using the same analogy as used for vanishing point, must meet at a single point, which is referred to as the Focus of Expansion (FoE) or Focus of Contraction (FoC). FoE

refers to the case where flow lines move away from a central point as seen in Figure 6.7. FoC relates to scenarios where the flow lines move towards a central point as illustrated in Figure 6.8. Figure 6.7 and Figure 6.8 illustrate the images of a rectangle specified by 4 corner points A, B, C and D. These corner points move to A', B', C' and D' as the camera moves away or towards the scene. AA', BB', CC' and DD' are the flow lines, which depending on the direction of the camera movement, meet at FoE or FoC.

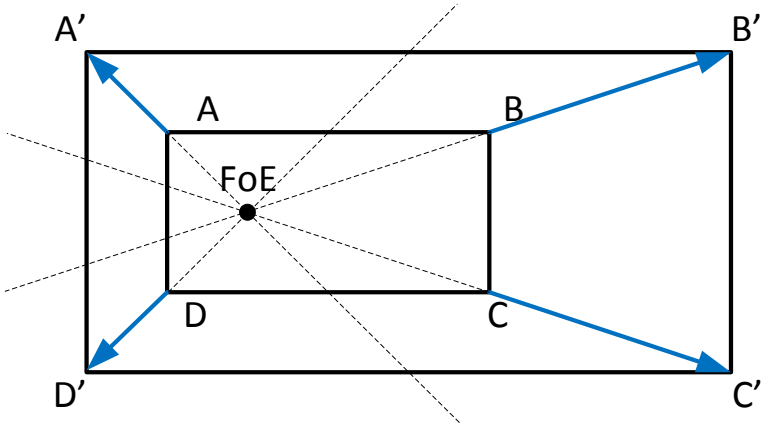


Figure 6.7 Focus of Expansion – Camera moves towards the object

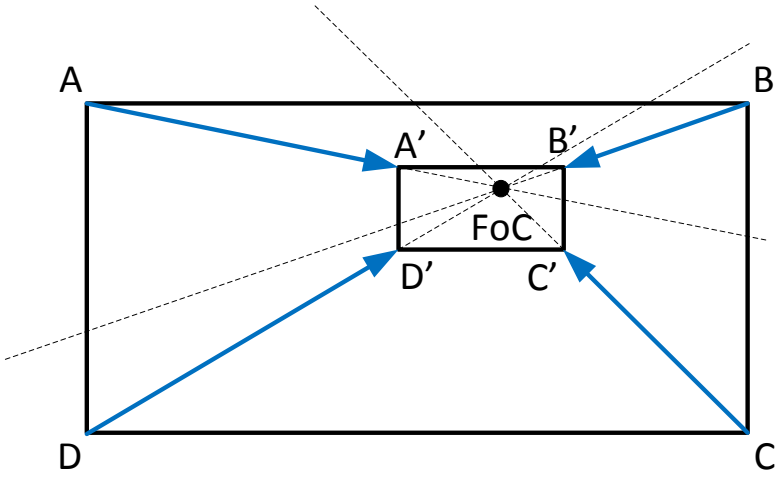


Figure 6.8 Focus of Contraction – Camera moves away from the object

6.3.3.1 Finding the Focus of Expansion Point

In this work both FoC and FoE cases are treated in the same way and there is no need to differentiate between them. Therefore for ease of reference both are referred to as the Focus of Expansion (FoE). Although in theory the FoE must be a single point, in practice, due to the image noise and feature detection error, the flow lines may not all meet at exactly the same point. Therefore a point on the image plane, which has the minimum overall distance to all flow lines, is considered to be the FoE point. A flow line is characterised by two associated feature points $A_m = (x_{m,A}, y_{m,A})$ and $\mathrm{A'_m} = (x_{m,A'}, y_{m,A'})$ on the image plane as per equation (6.51) or a simpler form in equation (6.52). Index $m \in (1 \dots N_f)$ refers to the feature point number.

(6.51) $\frac{y-y_{m,A}}{x-x_{m,A}} = \frac{y_{m,A'}-y_{m,A}}{x_{m,A'}-x_{m,A}}$

(6.52) $\boldsymbol{a_m x + b_m y + c_m = 0}$,

(6.53) $\boldsymbol{a_m = \frac{y_{m,A'}-y_{m,A}}{x_{m,A'}-x_{m,A}},\ b_m = -1, c_m = -x_{m,A}\frac{y_{m,A'}-y_{m,A}}{x_{m,A'}-x_{m,A}} + y_{m,A}}$

Suppose the coordinates of the FoE point on the 2D image is $(x_{FOE}, y_{FOE})^T$. The point is on the image plane, therefore its distance along the camera z axis is the focal length, f. The 3D coordinates of the FoE point with respect to the camera frame is given by:

(6.54) $\boldsymbol{FoE^c = \begin{pmatrix} x_{FOE} \\ y_{FOE} \\ f \end{pmatrix}}$

The distance between the FoE and a flow line, as shown in Figure 6.9, can be calculated using the equation (6.55):

(6.55) $\boldsymbol{d_m = \frac{a_m x_{FoE} + b_m y_{FoE} + c_m}{\sqrt{a_m^2 + b_m^2}}}$, $\boldsymbol{m \in (1 \dots N_f)}$

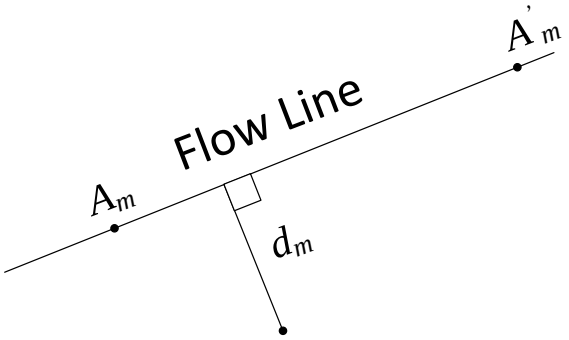


Figure 6.9 FoE to Flow Line Distance

The overall distance from the FoE to all flow lines is therefore calculated as:

(6.56) $\boldsymbol{d^2 = \sum_{m=1}^{N_f} \frac{(a_m x + b_m y + c_m)^2}{a_m^2 + b_m^2}, \qquad m \in (1 \dots N_f)}$

The FoE is the point, where d^2 has the minimum value. To identify this point the partial derivatives with respect to x and y are calculated. The coordinates of the FoE are where these partial derivatives become zero.

(6.57) $\boldsymbol{\frac{\partial d^2}{\partial x} = \left(\sum_{m=1}^{N_f} \frac{2a_m^2}{a_m^2 + b_m^2}\right) x + \left(\sum_{m=1}^{N_f} \frac{2a_m b_m}{a_m^2 + b_m^2}\right) y + \sum_{m=1}^{N_f} \frac{2a_m c_m}{a_m^2 + b_m^2} = 0}$

(6.58) $\boldsymbol{\frac{\partial d^2}{\partial y} = \left(\sum_{m=1}^{N_f} \frac{2a_m b_m}{a_m^2 + b_m^2}\right) x + \left(\sum_{m=1}^{N_f} \frac{2b_m^2}{a_m^2 + b_m^2}\right) y + \sum_{m=1}^{N_f} \frac{2b_m c_m}{a_m^2 + b_m^2} = 0}$

Equations (6.57) and (6.58) can be written in matrix form as per (6.59), resulting in the FoE value as in (6.60):

(6.59) $$\begin{pmatrix} \sum_{m=1}^{N_f} \frac{2a_m^2}{b_m^2 + c_m^2} & \sum_{m=1}^{N_f} \frac{2a_m b_m}{b_m^2 + c_m^2} \\ \sum_{m=1}^{N_f} \frac{2a_m b_m}{b_m^2 + c_m^2} & \sum_{m=1}^{N_f} \frac{2b_m^2}{b_m^2 + c_m^2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \sum_{m=1}^{N_f} \frac{2a_m c_m}{b_m^2 + c_m^2} \\ \sum_{m=1}^{N_f} \frac{2b_m c_m}{b_m^2 + c_m^2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

(6.60) $$\begin{pmatrix} x_{FoE} \\ y_{FoE} \end{pmatrix} = -\begin{pmatrix} \sum_{m=1}^{N_f} \frac{2a_m^2}{b_m^2 + c_m^2} & \sum_{m=1}^{N_f} \frac{2a_m b_m}{b_m^2 + c_m^2} \\ \sum_{m=1}^{N_f} \frac{2a_m b_m}{b_m^2 + c_m^2} & \sum_{m=1}^{N_f} \frac{2b_m^2}{b_m^2 + c_m^2} \end{pmatrix}^{-1} \begin{pmatrix} \sum_{m=1}^{N_f} \frac{2a_m c_m}{b_m^2 + c_m^2} \\ \sum_{m=1}^{N_f} \frac{2b_m c_m}{b_m^2 + c_m^2} \end{pmatrix}$$

In this work, two types of FoE have been considered, namely $FoE_{t_n}^c$ and $FoE_{t_n}^k$. $FoE_{t_n}^c$ is the FoE calculated at time t_n on the image taken at time t_{n-1}, using

two consecutive images, one taken at time t_{n-1} and one at the present time, t_n. $FoE_{t_n}^{k}$ is the FoE calculated at time t_n on the key image, using two images, one taken at time of key image and one at present time, t_n.

6.3.3.2 Properties of Focus of Expansion

Suppose a camera goes through a translational movement specified by $T = (T_x, T_y, T_z)^T$, while 3 points A, B and C in the 3D space are in its field of view. From the camera point of view, the camera is static and the points have moved parallel to each other in the opposite direction by $-T$ to points A', B' and C'. It can be shown that the line connecting the centre of the camera to the FoE ($\overrightarrow{OF}$) is parallel to the translation vector between the two images (Burger & Bhanu, 1990). Figure 6.10 illustrates this concept.

(6.61) $\overrightarrow{\boldsymbol{OF}} \parallel \boldsymbol{T}$

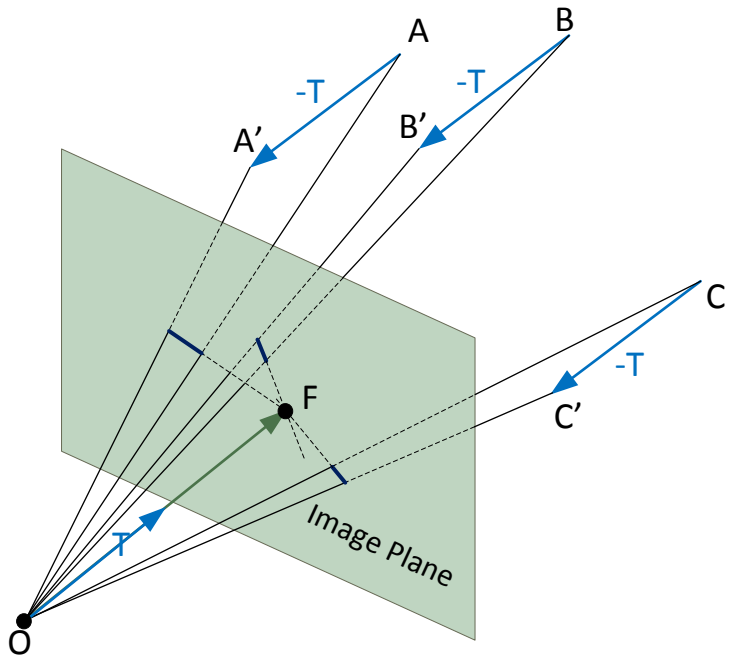


Figure 6.10 Focus of Expansion

When two consecutive images are considered, the time interval between the two images is small, therefore the linear speed $v = (v_x, v_y, v_z)^T$ can be

estimated using the translation vector, where Δt is the time interval between the two consecutive images:

(6.62) $\boldsymbol{v} = \frac{\boldsymbol{T}}{\Delta \boldsymbol{t}}$

This suggests that the camera speed vector is also parallel to the $\overrightarrow{OF}$ vector associated with two consecutive images.

(6.63) $\overrightarrow{\boldsymbol{OF}} \parallel \boldsymbol{T} \parallel \boldsymbol{v}$

$\overrightarrow{\mathrm{OF}}$ is the 3D coordinate of FoE with respect to the camera frame. As the distance between the image plane and the camera origin equals the focal length f, $\overrightarrow{\mathrm{OF}}$ has the following 3D coordinates:

(6.64) $\overrightarrow{\mathbf{OF}} = \begin{pmatrix} \boldsymbol{x_{FOE}} \\ \boldsymbol{y_{FOE}} \\ \boldsymbol{f} \end{pmatrix}$

$\overrightarrow{OF}$ and T are parallel, therefore their coordinates are related to each other as per (6.65). Consequently the translation vector can be used to derive the coordinates of FOE^T as per equation (6.66). Similarly the velocity vector can be used to derive the coordinates of FoE between two consecutive images as per equation (6.67):

(6.65) $\frac{\boldsymbol{x_{FOE}}}{\boldsymbol{T_x}} = \frac{\boldsymbol{y_{FOE}}}{\boldsymbol{T_y}} = \frac{\boldsymbol{f}}{\boldsymbol{T_z}}$

(6.66) $\boldsymbol{x_{FOE^T}} = \boldsymbol{f}\frac{\boldsymbol{T_x}}{\boldsymbol{T_z}}, \quad \boldsymbol{y_{FOE^T}} = \boldsymbol{f}\frac{\boldsymbol{T_y}}{\boldsymbol{T_z}}$

(6.67) $\boldsymbol{x_{FOE^v}} = \boldsymbol{f}\frac{\boldsymbol{v_x}}{\boldsymbol{v_z}}, \quad \boldsymbol{y_{FOE^v}} = \boldsymbol{f}\frac{\boldsymbol{v_y}}{\boldsymbol{v_z}}$

Properties of FoE are used to evaluate the position and velocity vectors of the generated particles in the proposed algorithm (see section 6.4.4 for details).

6.3.3.3 Quality of Focus of Expansion

Although, in theory, the FoE must be a single point, in practice due to noise and error, the flow lines may not all meet at the same point. Also in some cases, where the velocity or translation vector is parallel to the image plane, the images of the flow lines are parallel and therefore do not meet at a single point.

The estimated $FoE_{t_n}^{c}$ compared to $FoE_{t_n}^{k}$ is more likely to suffer from noise and error. This is due to the fact that flow lines are shorter and noise and error effects are more noticeable. In order to evaluate the accuracy and validity of FoE, two measures are considered. The first measure is the angle of movement with respect to the image plane. The second measure is the average distance from the FoE to flow lines. These measures are explained in the following two subsections.

6.3.3.3.1 Angle of Movement

When the camera translation vector or the camera linear velocity vector is parallel to the image plane, or there is a small angle between them, the flow lines are effectively parallel to each other and do not meet at a single point. Figure 6.11 illustrates this issue.

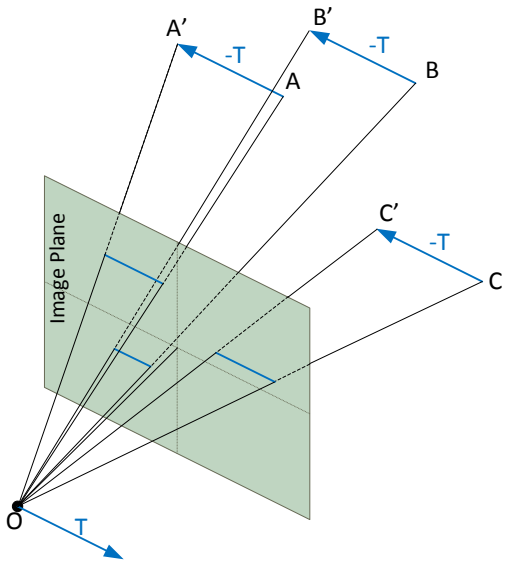


Figure 6.11 Parallel Flow Lines

The angle between the image plane and the translation vector at time t_n is calculated using equation (6.68). Figure 6.12 shows this angle.

$$if \ \measuredangle T, N = 90^o \ \ then \ \ T \odot N = 0$$

Therefore:

(6.68) $$\boldsymbol{\theta_T} = \frac{\boldsymbol{\pi}}{\boldsymbol{2}} - \boldsymbol{cos}^{\boldsymbol{-1}} \frac{\boldsymbol{T^w} \odot \boldsymbol{N^w}}{\|\boldsymbol{T^w}\|}$$

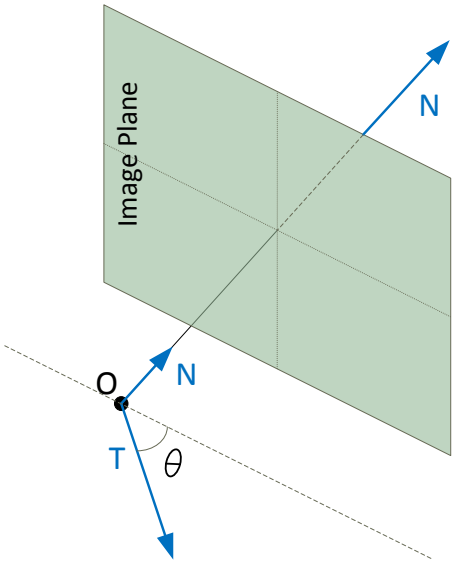


Figure 6.12 Angle of camera movement

N^w is the unit normal vector of the image plane with respect to the world frame. T^w is the translation vector between the two images with respect to the world frame.

Similarly to the translation vector, if the camera velocity vector is parallel to the image plane, a single FoE point cannot be formed. The angle between the velocity vector and the image plane is derived from the following equation:

(6.69) $$\boldsymbol{\theta_v} = \frac{\boldsymbol{\pi}}{\boldsymbol{2}} - \boldsymbol{cos}^{\boldsymbol{-1}} \frac{\boldsymbol{v^w N^w}}{\|\boldsymbol{v^w}\|}$$

v^{w} is the translation vector between the two images with respect to the world frame. θ_T and θ_v angles are used to determine whether or not a single FoE point exists. Small θ_T and θ_v angles indicate that an associated FoE point does not exist.

6.3.3.3.2 Average Distance Between FoE Point and The Flow Lines

The average distance of the calculated FoE to all optical flow lines is a good measure for the quality of FoE. Providing that the FoE point exists, a high average FoE distance indicates that the error in the location of feature points have led to a poor flow line estimation and therefore flow lines have not met at a single point. For a FoE point, the average distance based on equation (6.56) is as follows:

(6.70) $$\boldsymbol{d_{FOE}} = \frac{\mathbf{1}}{\boldsymbol{m}} \sqrt{\sum_{\boldsymbol{m=1}}^{\boldsymbol{N_f}} \frac{(\boldsymbol{a_m x_{FOE} + b_m y_{FOE} + c_m})^2}{\boldsymbol{a_m^2 + b_m^2}}}$$

$(x_{FOE}, y_{FOE})^T$ is the coordinate vector of the FoE point, N_f is the number of feature points and (a_m, b_m, c_m) are the parameters of the flow line associated with feature number m as per equation (6.52).

6.3.4 Estimating Camera Orientation

As outlined in section 6.1, the camera orientation can be derived from the rotation matrix, and vice versa. Therefore, in this work, the focus is on finding the rotation matrix instead of orientation. The vision-based system is used for this purpose. To do so, the feature points from the key image are tracked and identified in the current image. The fundamental matrix F between the two images is then formed using the 8- point algorithm (see Chapter 5). By knowing the camera intrinsic matrix, K, the essential matrix (E) can be derived from the fundamental matrix as follows:

(6.71) $$\boldsymbol{F} = \boldsymbol{K^{-T} E K^{-1}}$$

The essential matrix, E, defines the relationship between corresponding feature points in two calibrated camera views. Referring to Chapter 5, the essential matrix can be decomposed using Singular Value Decomposition (SVD) so that:

(6.72) $$\boldsymbol{E} = \boldsymbol{U}\begin{pmatrix}\mathbf{1} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{pmatrix}\boldsymbol{V^T}$$

U and V are 3D vectors resulting from the application of SVD to the fundamental matrix. Once U and V are known, matrix W is defined as per (6.74) and then the rotation matrix between the current and key image frames at time t_n is computed as follows:

(6.73) $$\boldsymbol{R_{t_n}^{kc}} = \mathbf{UWV^T}$$

(6.74) $$\boldsymbol{W} = \begin{pmatrix}\mathbf{0} & \mathbf{-1} & \mathbf{0}\\ \mathbf{1} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{1}\end{pmatrix}$$

In order to arrive at the rotation matrix with respect to the world reference frame, the rotation matrix of the key image ($\mathrm{R}_{\mathrm{t_n}}^{\mathrm{wk}}$) must be multiplied by the above rotation matrix ($\mathrm{R}_{\mathrm{t_n}}^{\mathrm{kc}}$) resulting in rotation matrix $R_{t_n}^{wc}$ for transformation from the camera frame to the world frame.

(6.75) $$\mathbf{R_{t_n}^{wc}} = \mathbf{R^{wk}R_{t_n}^{kc}}$$

This method provides two possible solutions $\bar{R}_{t_n}^{wc}$ and $\bar{\bar{R}}_{t_n}^{wc}$, reflecting the fact the application of SVD provides two possible solutions; $\bar{R}_{t_n}^{kc}$ and $\bar{\bar{R}}_{t_n}^{kc}$. In order to select the correct rotation matrix, the resultant rotation matrix is compared with the estimate provided by the IMU system (see equation

(6.49) and the matrix closest to the IMU estimate, $(R_{\boldsymbol{t_{n-1}+N_i\delta t}}^{wi})\boldsymbol{R^{ic}}$, is considered to be the correct rotation matrix. In order to carry out this comparison, the mean square error between individual components of $R_{\boldsymbol{t_{n-1}+N_i\delta t}}^{wi}$ and $\bar{R}_{t_n}^{wc}$ on one hand and $R_{\boldsymbol{t_{n-1}+N_i\delta t}}^{wi}$ and $\bar{\bar{R}}_{t_n}^{wc}$ on the other is determined. The matrix with the least

error is considered to be the correct rotation matrix. Here matrices A and B are defined as follows:

(6.76) $\mathbf{A} = (\mathbf{R}^{\mathbf{wi}}_{\mathbf{t_{n-1}+N_i\delta t}})\boldsymbol{R^{ic}} - \overline{\mathbf{R}}^{\mathbf{wc}}_{\mathbf{t_n}}$

(6.77) $\mathbf{B} = (\mathbf{R}^{\mathbf{wi}}_{\mathbf{t_{n-1}+N_i\delta t}})\boldsymbol{R^{ic}} - \overline{\overline{\mathbf{R}}}^{\mathbf{wc}}_{\mathbf{t_n}}$

The mean square error is given by $\mathrm{d_A}$ and $\mathrm{d_B}$ for matrices A and B, where $\mathrm{a_{lj}}$ and $\mathrm{b_{lj}}$ are their individual components.

(6.78) $\boldsymbol{d_A} = \frac{\sqrt{\sum_{l=1}^{3}\sum_{j=1}^{3} \boldsymbol{a_{lj}^2}}}{\mathbf{9}}$

(6.79) $\boldsymbol{d_B} = \frac{\sqrt{\sum_{l=1}^{3}\sum_{j=1}^{3} \boldsymbol{b_{lj}^2}}}{\mathbf{9}}$

If $\mathrm{d_A} < \mathrm{d_B}$, $\overline{\mathrm{R}}^{\mathrm{wc}}_{\mathrm{t_n}}$ is the correct rotation matrix, otherwise $\overline{\overline{\mathrm{R}}}^{\mathrm{wc}}_{\mathrm{t_n}}$ is considered to be correct.

6.3.5 Replacing Key Image

If the number of tracked feature points in the current image falls below a threshold, N_{min}^{th}, the key image must be replaced by a more recent one which, by virtue of closer temporal proximity, should have considerably more feature points in common with the current image frame. The new key image must be selected in such a way to ensure the Focus of Expansion associated with the key and current images is of a good quality (see section 6.3.3.3). Once a new key image is selected, the optical-flow tracking program is reset to start the tracking process from the feature points in the new key image.

6.4 Stochastic Data Fusion (SDF)

So far in this chapter the IMU-based kinematic motion equations and the associated pose estimation mechanism have been described in detail in section 6.2. Furthermore, the feature detection and tracking routines along with methods for estimating orientation as well as Focus of Expansion (FoE) were outlined in section 6.3. In this section the data gathered from the IMU and vision based systems are fused to provide an accurate and robust pose tracking.

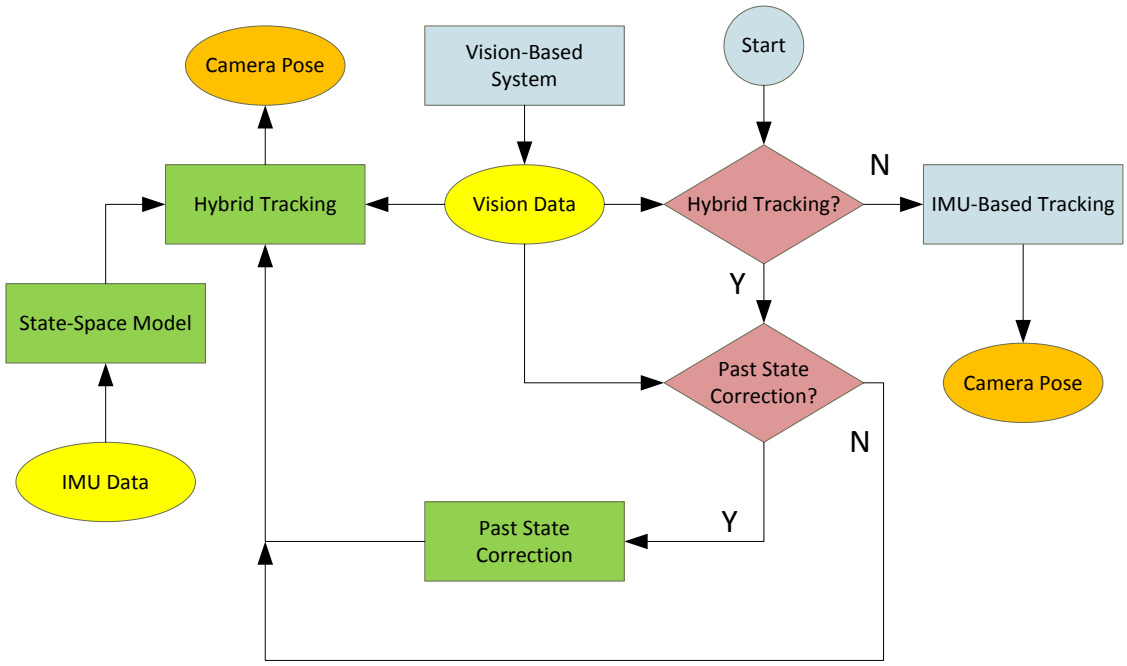


Figure 6.13 Pose Tracking with Tracking Source Selection

The core of the proposed hybrid tracking method is a data fusion system based on particle filtering. The data fusion system relies on the properties of FoE, and vision data to operate. The quality of FoE is used as a measure to determine whether or not the vision data provide sufficient information for hybrid tracking. The system incorporates a mechanism for selecting the tracking method, whether pure IMU-based or hybrid tracking, with or without past state correction. Figure 6.13 provides a block diagram of the overall tracking solution. The diagram shows that the camera pose can either be the output of the IMU-

based or hybrid tracking and the decision on the method of tracking is made based on the vision data.

The hybrid tracking method proposed here works on the basis of recursive filtering techniques. A recursive particle filter is defined in the context of a state-space model. The following section provides a brief overview of the state-space model and its use in state estimation using kinematic motion equations, then particle filtering, as a recursive filtering technique, is outlined. The remainder of the section describes the details of the proposed particle filter-based tracking method. Subsequently all relevant issues in order to achieve effective hybrid tracking are outlined and suitable solutions are provided

6.4.1 State Space Model and Recursive Filtering

A state-space model describes a system with a set of input, output and state vectors represented by $U(t), Y(t)$ and $S(t)$ as shown in Figure 6.14. $S'(t)$ is the first derivative of $S(t)$.

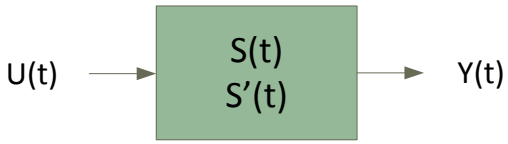


Figure 6.14 State-Space Model

The model relates $U(t), Y(t), S(t)$ and $S'(t)$ vectors in two sets of equations as follows (Ristic, et al., 2004):

(6.80) $\boldsymbol{S'(t) = f(S(t), U(t))}$

(6.81) $\boldsymbol{Y(t) = h(S(t), U(t))}$

These equations provide state update and observation models, respectively. The discrete representation of these equations is:

(6.82) $\boldsymbol{S(t + \delta t) = f(S(t), U(t))}$

(6.83) $\boldsymbol{Y(t) = h(S(t), U(t))}$

The above models formulate the case for a perfect system, where there is no error in modelling or measurement of the output or the control inputs. In practical applications, however, such effects need to be considered. Therefore equations (3.3) and (3.4) are re-written in order to include noise as well as measurement and modelling error (η and ξ). Here noise and error are collectively referred to as noise.

(6.84) $\boldsymbol{S(t + \delta t) = f(S(t), U(t), \eta)}$

(6.85) $\boldsymbol{Y(t) = h(S(t), U(t), \xi)}$

In general $f(.)$ and $g(.)$ are nonlinear functions. However when they are linear the model can be described as follows:

(6.86) $\boldsymbol{S(t + \delta t) = A(t)S(t) + B(t)U(t) + \eta(t)}$

(6.87) $\mathbf{Y(t) = C(t)S(t) + D(t)U(t) + \xi(t)}$

Here the recursive filtering techniques come into play. As described in Chapter 3 these methods try to estimate the correct state of the system using the noisy and erroneous measurement and process model. These methods predict the current system state using equation (3.1), considering past system states, $S(t)$, control input, $U(t)$, and noise model η. Once new observation data become available, equation (3.2) is used to correct the state prediction and update filter parameters.

There are three well-known recursive filtering methods; namely, Kalman Filter (KF), Extended Kalman Filter (EKF) and Particle Filter (PF). These have been described in some detail in Chapter 3. Kalman Filter only applies to linear systems (equations (3.11) and (3.12)) with normal probability distribution for measurement and observation noise (η and ξ). Extended Kalman Filter (EKF) is applicable when $f(.)$ and $g(.)$ are non-linear but can be linearised around the

current system state. Particle filtering is used when $f(.)$ or $g(.)$ or both and non-linear and cannot be linearised effectively.

Here, for ease of reference, the kinematic motion equations provided in section 6.2.1 are presented again.

(6.88) $$\begin{pmatrix} \boldsymbol{X^w_{t_{n-1}+\delta t}} \\ \boldsymbol{V^w_{t_{n-1}+\delta t}} \\ \boldsymbol{R^{wi}_{t_{n-1}+\delta t}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X^w_{t_{n-1}} + V^w_{t_{n-1}}\delta t + \frac{\delta t^2}{2}\left(R^{wi}_{t_{n-1}} A^i_{t_{n-1}} + G\right)} \\ \boldsymbol{V^w_{t_{n-1}} + \delta t\left(R^{wi}_{t_{n-1}} A^i_{t_{n-1}} + G\right)} \\ \boldsymbol{R_{t_{n-1},\delta t} R^{wi}_{t_{n-1}}} \end{pmatrix}$$

In order to define the motion equations in the context of state-space, rotation matrices R^{wi}_t and $R^{wi}_{t+\delta t}$ need to be converted to Euler angles. Alternatively this equation can be expressed using quaternions, which uniquely describe any rotation around a 3D point in space with an axis of rotation going through the point ($\vec{v}$) and an angle of rotation (θ). The quaternion of such a rotation is defined as follows (see Figure 6.15 for details):

(6.89) $$\boldsymbol{Q} = \left(\boldsymbol{cos}\left(\frac{\boldsymbol{\theta}}{\boldsymbol{2}}\right), \boldsymbol{sin}\left(\frac{\boldsymbol{\theta}}{\boldsymbol{2}}\right)\vec{\boldsymbol{v}}\right)$$

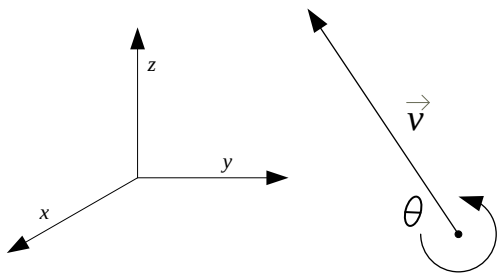


Figure 6.15 Quaternion Representation

The quaternion, as a means of describing rotation, is closely linked to the angular velocity. Angular velocity vector $\vec{\Omega}^i_t$ can be expressed as a quaternion by forming a 4D vector as follows:

(6.90) $$\boldsymbol{\Omega^i_{q,t}} = \left(\mathbf{0}, \boldsymbol{\vec{\Omega}^i_t}\right)$$

It can be shown that the first derivative of a quaternion can be expressed using the quaternion multiplication of angular velocity quaternion $\vec{\Omega}_t^i$ and current quaternion.

(6.91) $$\dot{\boldsymbol{Q}}_{\boldsymbol{t}}^{\boldsymbol{w}} = \frac{\mathbf{1}}{\mathbf{2}} \boldsymbol{Q}_{\boldsymbol{t}}^{\boldsymbol{w}} \otimes \boldsymbol{\Omega}_{\boldsymbol{q,t}}^{\boldsymbol{i}}$$

Note that angular velocity is with respect to the camera frame, which is the output of the gyroscope. Operator $\otimes$ represents quaternion multiplication. Using this notation and defining the state vector $S_t^w = (X_t^w, V_t^w, Q_t^w)^T$ the state-space representation of the camera motion will be:

(6.92) $$\begin{pmatrix} \boldsymbol{X}_{\boldsymbol{t_{n-1}+\delta t}}^{\boldsymbol{w}} \\ \boldsymbol{V}_{\boldsymbol{t_{n-1}+\delta t}}^{\boldsymbol{w}} \\ \boldsymbol{Q}_{\boldsymbol{t_{n-1}+\delta t}}^{\boldsymbol{w}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}_{\boldsymbol{t_{n-1}}}^{\boldsymbol{w}} + \boldsymbol{V}_{\boldsymbol{t_{n-1}}}^{\boldsymbol{w}} \boldsymbol{\delta t} + \frac{\boldsymbol{\delta t^2}}{\mathbf{2}} \left(\boldsymbol{R}_{\boldsymbol{t_{n-1}}}^{\boldsymbol{wi}} \boldsymbol{A}_{\boldsymbol{t_{n-1}}}^{\boldsymbol{i}} + \boldsymbol{G} \right) \\ \boldsymbol{V}_{\boldsymbol{t_{n-1}}}^{\boldsymbol{w}} + \boldsymbol{\delta t} \left(\boldsymbol{R}_{\boldsymbol{t_{n-1}}}^{\boldsymbol{wi}} \boldsymbol{A}_{\boldsymbol{t_{n-1}}}^{\boldsymbol{i}} + \boldsymbol{G} \right) \\ \frac{\mathbf{1}}{\mathbf{2}} \boldsymbol{\delta t} \boldsymbol{Q}_{\boldsymbol{t_{n-1}}}^{\boldsymbol{w}} \otimes \boldsymbol{\Omega}_{\boldsymbol{q,t_{n-1}}}^{\boldsymbol{i}} \end{pmatrix}$$

This is clearly a non-linear model, which should either be solved using EKF or PF. The EKF-based tracking methods, such as EKF-SLAM require several key frames and have to continuously build a map of the area, with a considerable computational cost (see Chapter 3).

The solution proposed here however requires 3 images only and has a simple but non-linear observation function (see section 6.4.4). As described earlier, in this method the camera orientation is estimated using the vision-based system only (see section 6.3.4), therefore orientation state is removed from the state-space model above, leaving a simpler state-space model, with only 6 elements defined as:

(6.93) $$\boldsymbol{S}_{\boldsymbol{t}}^{\boldsymbol{w}} = \begin{pmatrix} \boldsymbol{X}_{\boldsymbol{t_n}}^{\boldsymbol{w}} \\ \boldsymbol{V}_{\boldsymbol{t_n}}^{\boldsymbol{w}} \end{pmatrix}$$

Considering (6.92) and the above definition for state vector, S_t^w , the state-space model is defined as follows:

(6.94) $$\begin{pmatrix} \mathbf{X}^{\mathbf{w}}_{\mathbf{t_{n-1}+\delta t}} \\ \mathbf{V}^{\mathbf{w}}_{\mathbf{t_{n-1}+\delta t}} \end{pmatrix} = \begin{pmatrix} \mathbf{I}_{3\times3} & \boldsymbol{\delta t}.\,\mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{pmatrix} \begin{pmatrix} \mathbf{X}^{\mathbf{w}}_{\mathbf{t_{n-1}}} \\ \mathbf{V}^{\mathbf{w}}_{\mathbf{t_{n-1}}} \end{pmatrix} + \begin{pmatrix} \frac{\boldsymbol{\delta t}^2}{2}.\,\mathbf{I}_{3\times3} \\ \boldsymbol{\delta t}.\,\mathbf{I}_{3\times3} \end{pmatrix} \left(\mathbf{R}^{\mathbf{wi}}_{\mathbf{t_{n-1}}}\,\mathbf{A}^{\mathbf{i}}_{\mathbf{t_{n-1}}} + \mathbf{G}\right) \Rightarrow$$

(6.95) $$\mathbf{S}^{\mathbf{w}}_{\mathbf{t_{n-1}+\delta t}} = \mathbf{A}\mathbf{S}^{\mathbf{w}}_{\mathbf{t_{n-1}}} + \mathbf{B}\left(\mathbf{R}^{\mathbf{wi}}_{\mathbf{t_{n-1}}}\,\mathbf{A}^{\mathbf{i}}_{\mathbf{t_{n-1}}} + \mathbf{G}\right)$$

Matrices A and B are defined as:

(6.96) $$\mathbf{A} = \begin{pmatrix} \mathbf{I}_{3\times3} & \boldsymbol{\delta t}.\,\mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{pmatrix},\ \mathbf{B} = \begin{pmatrix} \frac{\boldsymbol{\delta t}^2}{2}.\,\mathbf{I}_{3\times3} \\ \boldsymbol{\delta t}.\,\mathbf{I}_{3\times3} \end{pmatrix}$$

Taking into account the new definition, and using equation (6.49) the system state at time $\boldsymbol{t_n}$ is estimated to be:

(6.97) $$\mathbf{S}^{\mathbf{w}}_{\mathbf{t_n}} = \mathbf{A}\mathbf{S}^{\mathbf{w}}_{\mathbf{t_{n-1}}} + \mathbf{B}\Delta\mathbf{S}^{\mathbf{w}}_{\mathbf{t_{n-1}}}$$

(6.98) $$\Delta\mathbf{S}^{\mathbf{w}}_{\mathbf{t_{n-1}}} = \begin{pmatrix} \sum_{\mathbf{m=1}}^{\mathbf{N_i}} (\mathbf{2N_i + 1 - 2m}) \left(\mathbf{R}^{\mathbf{wi}}_{\mathbf{t_{n-1}+m\delta t}}\,\mathbf{A}^{\mathbf{i}}_{\mathbf{t_{n-1}+m\delta t}} + \mathbf{G}\right) \\ \sum_{\mathbf{m=1}}^{\mathbf{N_i}} \left(\mathbf{R}^{\mathbf{wi}}_{\mathbf{t_{n-1}+m\delta t}}\,\mathbf{A}^{\mathbf{i}}_{\mathbf{t_{n-1}+m\delta t}} + \mathbf{G}\right) \end{pmatrix}$$

Equation (6.98) is used as the basis for particle generation, as outlined in section 6.4.3.

6.4.2 Framework for PF-Based Hybrid Tracking

Due to the non-linear nature of the suggested observation model (see section 6.4.4), the only applicable method for filtering and data fusion is particle filtering. However by reducing the number of states to 6 (see (6.94)) and proposing simple yet effective particle selection and evaluation methods, a robust, reliable and accurate hybrid tracking solution with a manageable computation cost has

been developed. This section outlines the framework for the PF-based hybrid tracking.

An overview of the PF-based hybrid tracking is presented first, followed by a detailed description of the particle generation and evaluation methods. In the subsequent subsections, the procedure for assigning weight to particles, the re-sampling algorithm, and finally the method for estimating the current state of the system based on particles are provided. After the application of particle filtering, the process designed for past state correction is described in detail and also a method for replacing the key image is presented.

The principle of operation for any particle filter method is based on selecting particles from a proposal distribution, assigning a weight to each particle using a likelihood function, applying re-sampling if necessary, and finally providing the system state using the particles and their associated weights. The particle filtering method employed in this work is based on the Sequential Importance Sampling (SIS) algorithm described in Chapter 3. In this section the SIS algorithm is briefly outlined and then a block diagram of the proposed PF-based tracking method is provided.

6.4.2.1 Summary of the SIS-based Particle Filtering Method

The SIS method consists of a sampling and evaluation stage, as well as a re-sampling stage, as follows.

6.4.2.1.1 SIS - Sampling and Evaluation

When a new image is captured, the following algorithm is executed for each particle:

D. Draw a new particle from a posterior probability distribution function (PDF) (section 6.4.3):
   $P_{p,t_n} \sim \ p(S_{t_n}|P_{p,t_{n-1}})$

$P_{p,t_{n-1}}$is the past particle, $P_{p,t_n}$is the current particle and $p(S_{t_n}|P_{p,t_{n-1}})$ is the PDF for the current system state, S_{t_n}.

E. Calculate weights using likelihood function (sections 6.4.4 and 6.4.5):

$\widehat{W}_{p,t_n} = p(Z_{t_n}|P_{p,t_n})\widehat{W}_{p,t_{n-1}}$

$p(Z_{t_n}|P_{p,t_n})$ is the probability of having the current observation data, Z_{t_n}, providing that the system state is specified by particle P_{p,t_n}

F. Normalise weights (section 6.4.5)

$W_{p,t_n} = \widehat{W}_{p,t_n} / \sum \widehat{W}_{p,t_n}$

G. Calculate number of effective particles (section 6.4.6)

$N_{eff} = 1/\sum W_{p,t_n}^2$

H. If the number of effective particles, N_{eff}, is less than a threshold, run the resampling algorithm before proceeding to the next stage (section 6.4.6)

I. Estimate the current state of the system (section 6.4.7)

6.4.2.1.2 Resampling

A. Sort the particles based on their weights in descending order.

B. Keep the first N_{eff} particles and replace the remaining with the first N_{eff} particles.

C. Normalise weights

D. Calculate number of effective particles

$N_{eff} = 1/\sum W_{p,t_n}^2$

E. If the number of effective particles, N_{eff}, is less than a threshold, repeat the resampling process.

6.4.2.2 Block Diagram of the PF-based Hybrid Tracking

Figure 6.16 shows a block diagram of the PF-Based Tracker. The particle generation block receives tracking information from the IMU and by knowing the IMU characteristics and past state of the system provides a number of particles for evaluation. The particle evaluation block evaluates the particles using the current camera orientation and image feature points in the current, past and key

images. The outcome is provided as the particle score, which is then used by the weight-assignment block for assigning an importance weight to each particle and normalising them at the end. The normalised weights are used to calculate the number of effective particles and, if necessary, take the particles through the resampling block. The final particles are passed on to the state estimation block. The next block corrects the past state of the system if the conditions are met. Finally the key image is replaced if required. Each of these blocks is described in detail in the following subsections.

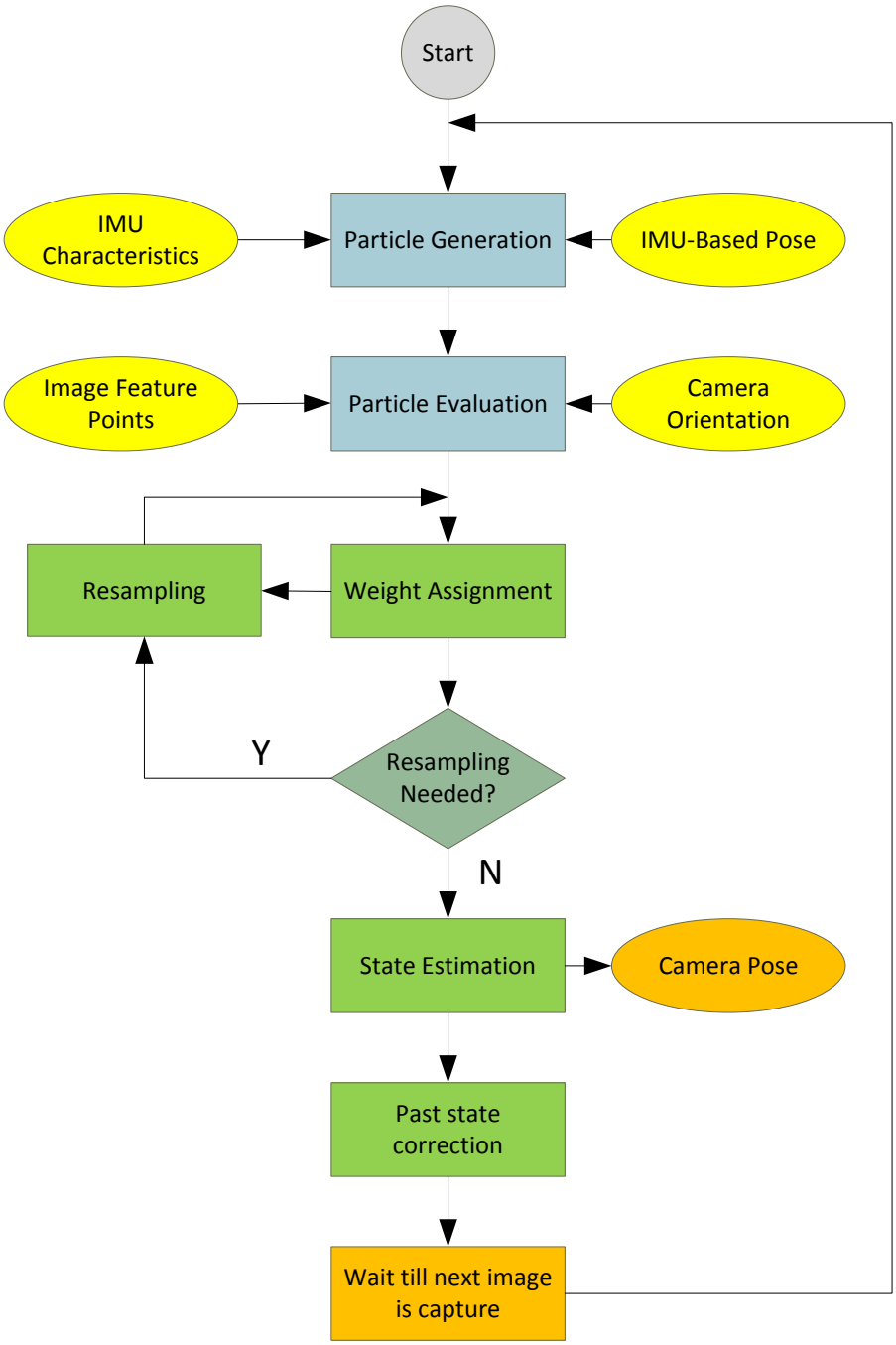


Figure 6.16 Block Diagram of the PF-Based Tracker

6.4.3 Particle Generation

At time t_n, when a new image is captured, N_p particles are drawn from the associated proposal distributions, which are formed based on the past particles,

control inputs and noise model. The basis of particle generation is the camera position (X) and velocity (V) estimated by the IMU at time $t_n = t_{n-1} + N_i \delta t$ using equation (6.97). As explained above, the PF state space vector is defined to only contain the camera position and velocity, therefore particle P^w_{p,t_n}, has an associated position vector P^w_{X,p,t_n} and velocity vector P^w_{V,p,t_n} as follows.

(6.99) $\boldsymbol{P^w_{p,t_n}} = \begin{pmatrix} \boldsymbol{P^w_{X,p,t_n}} \\ \boldsymbol{P^w_{V,p,t_n}} \end{pmatrix}$

Here, $p \in (1 \dots N_p)$ is the particle number, t_n represents the current time and w the world reference frame. The particles are drawn from proposal distributions formed based on the past particles as follows. $\mathrm{p}(\mathrm{S^w_{t_n}}|\mathrm{P^w_{p,t_{n-1}}})$ is the posterior probability distribution for the p^{th} particle, $p \in (1 \dots N_p)$.

(6.100) $\mathbf{P^w_{p,t_n}} \sim \boldsymbol{p}(\boldsymbol{S^w_{t_n}}|\boldsymbol{P^w_{p,t_{n-1}}})$

The proposal distribution is formed using equation (6.97). According to this equation, if the past state of the system is the p^{th} particle at time $\mathrm{t_{n-1}}$, the estimated current state will be $\tilde{\mathrm{S}}^{\mathrm{w}}_{\mathrm{t_n}}$ as follows:

(6.101) $\mathbf{\tilde{S}^w_{t_n}} = \boldsymbol{A P^w_{p,t_{n-1}}} + \boldsymbol{B \Delta S^w_{t_{n-1}}}$

The same analogy can be used for generating new particles.

(6.102) $\mathbf{P^w_{p,t_n}} = \mathbf{A P^w_{p,t_{n-1}}} + \mathbf{B \Delta P^w_{p,t_{n-1}}}$

Therefore in order to generate new particles, $\Delta P^w_{p,t_{n-1}}$ for each particle is drawn from a proposal distribution, centring at $\Delta S^w_{t_{n-1}}$ with a variance based on the IMU characteristics.

(6.103) $\Delta\mathbf{P^w_{p,t_n}} \sim \mathbf{N}(\Delta\mathbf{S^w_{t_{n-1}}}, \boldsymbol{\sigma_{t_n}})$, $\mathbf{p} \in (\mathbf{1} \dots \mathbf{N_p})$

The variance σ_{t_n}, must be chosen in a way that allows for possible error and noise generated by the IMU accelerometer. A typical accelerometer converts

the acceleration to a voltage and then using an A/D converter to a digital value. The resolution of the A/D conversion affects the measurement accuracy. For example a 16-bit A/D measuring in the range $\pm 2g$ (g is approx. $9.82 m/s^2$), can only measure with a measurement accuracy of $61 \mu g$. The analogue section of the device also typically has measurement noise, offset error, gain error and a non-linearity factor. The power spectral density of measurement noise is often defined by $\mu g/\sqrt{Hz}$. The total noise power depends on the measurement bandwidth. A typical accelerometer has an on-board low pass filter, which limits the signal frequency to 100Hz or less. The offset error is the sensor output when the acceleration is zero. This value is defined in mg units (InvenSense IMU Datasheet (InvenSense, 2003)).

The digital output of the sensor must be divided by a conversion factor in order to determine the actual acceleration. The conversion factor is affected by gain and non-linearity error factors. The gain error is the percentage error in the slope of the conversion line for converting digital sensor output to acceleration. The non-linearity error is also defined as a percentage and gives a measure of how linear the conversion curve is. By combining all these effects, the noise and error model for an accelerometer is defined as follows:

(6.104) $\mathbf{e_A^i = e_{noise}^i + e_{offset}^i + e_{gain}^i A_{t_n}^i}$

e_A^i is the total error due to the accelerometer noise and measurement error. $A_{t_n}^i$ is the measured acceleration and e_{noise}^i, e_{offset}^i and e_{gain}^i are equivalent noise, offset and gain error margins. $e_A^i, e_{\text{noise}}^i$, $e_{\text{offset}}^i, e_A^i$ and $A_{t_n}^i$ are 3D vectors. e_{gain}^i is a 3×3 diagonal matrix with the elements of the main diagonal being the gain error in x, y and z directions.

$e_{A_j}^i$ as the possible total deviation of measured acceleration from the actual value is used in conjunction with equation (6.98) to estimate the possible error caused in calculating $\Delta S_{t_{n-1}}^w$.

(6.105) $\boldsymbol{\sigma_{t_n}} = \mathbf{e}_{\Delta S^{w}_{t_{n-1}}} = \begin{pmatrix} \sum_{m=1}^{N_i}(2N_i + 1 - 2m)\left(\mathbf{R}^{wi}_{t_{n-1}+m\delta t}\mathbf{e}^{i}_{A_{t_{n-1}+m\delta t}}\right) \\ \sum_{m=1}^{N_i}\left(\mathbf{R}^{wi}_{t_{n-1}+m\delta t}\mathbf{e}^{i}_{A_{t_{n-1}+m\delta t}}\right) \end{pmatrix}$

$\sigma_{t_n}$is the variance in the proposal distribution (6.103). Once the proposal distribution is known, Inverse-Transform Sampling (ITS) (Princeton, 2014) is used to draw particles from the proposal distribution. To do so for each particle, first a random number is chosen from a uniform distribution:

(6.106) $\mathbf{r_{p,k}} \in [\mathbf{0} \quad \mathbf{1}],\ \ \mathbf{p} \in (\mathbf{1} \dots \mathbf{N_p}),\ \ \mathbf{k} \in (\mathbf{1} \dots \mathbf{6})$

Then, using the ITS algorithm, a real number for each element of the particle vector is found so that the Cumulative Distribution Function (CDF) of the proposal distribution function is r_{pk}. p is the particle index, N_p is the number of particles and k is the state index.

(6.107) $\mathbf{r_{p,k}} = \mathbf{CDF^{-1}}\left(\mathbf{N}(\Delta \mathbf{S^{w}_{k,t_n}},\ \boldsymbol{\sigma}_{\mathbf{k,t_n}})\right),\quad \mathbf{p} \in (\mathbf{1} \dots \mathbf{N_p}),\ \ \mathbf{k} \in (\mathbf{1} \dots \mathbf{6})$

6.4.4 Particle Evaluation

The particles generated by the system need to be evaluated and weighted based on the likelihood of their correctness. The particles are generated using the particle generation method outlined in section 6.4.3 and are evaluated by the vision-based system as presented in this section. Each particle (P^{W}_{p,t_n}) is a vector consisting of three position vectors and three linear velocity elements, $P^{w}_{X,p,t_n}, P^{w}_{V,p,t_n}$, as below:

(6.108) $\mathbf{P^{W}_{p,t_n}} = \begin{pmatrix} \mathbf{P^{w}_{X,p,t_n}} \\ \mathbf{P^{w}_{V,p,t_n}} \end{pmatrix},\quad \mathbf{p} \in (\mathbf{1} \dots \mathbf{N_p})$

The speed and position vectors of a particle are evaluated using the focus of expansion between two images which, in the case of speed, is the current and past images and, in the case of position, the current and key images.

6.4.4.1 Evaluation of the Position Vector of a Particle using FoE

Each particle P_{p,t_n}^{W} has a position vector P_{X,p,t_n}^{W}, which represents the likely position of the camera origin with respect to the world frame. The position vector of each particle is evaluated using the focus of expansion between the current and key images.

Referring to section 6.3.3.2, the translation vector between two images is parallel to the line connecting the optical centre of the camera to the FoE point, providing that there is no rotation element between the two images. Therefore, in order to evaluate the position vector of a particle, the translation vector between the key image and each particle, T_{p,t_n}^{w}, must be formed using equation (6.109), where $X_{t_k}^{w}$ is a known vector. Figure 6.17 illustrates the relationship between particles and the key image.

(6.109) $\mathbf{T}_{\mathbf{p,t_n}}^{\mathbf{w}} = \mathbf{P}_{\mathbf{X,p,t_n}}^{\mathbf{w}} - \mathbf{X}_{\mathbf{t_k}}^{\mathbf{w}}$

T_{p,t_n}^{w} is with respect to the world frame and needs to be transformed to the key image reference frame using R^{wk}, so that it can be used for FoE analysis.

(6.110) $\mathbf{T}_{\mathbf{p,t_n}}^{\mathbf{k}} = \mathbf{R}^{\mathbf{kw}}\mathbf{T}_{\mathbf{p,t_n}}^{\mathbf{w}} = \mathbf{R}^{\mathbf{wk}^{-1}}\mathbf{T}_{\mathbf{p,t_n}}^{\mathbf{w}}$

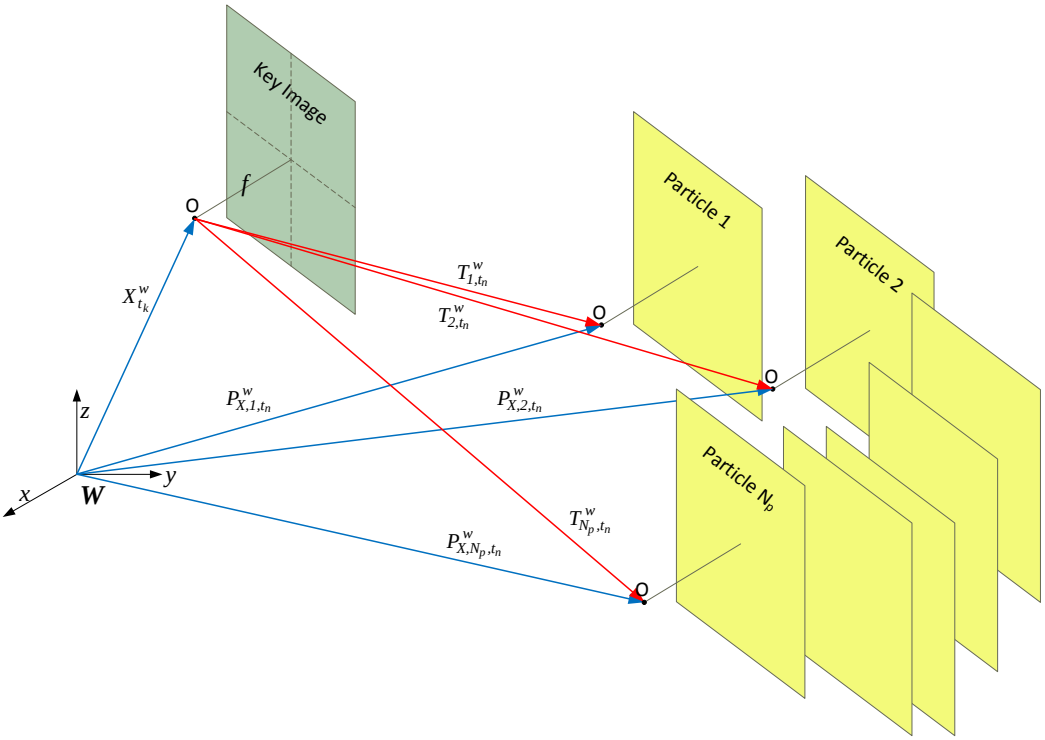


Figure 6.17 Translation between the key image and each particle

In order to use the focus of expansion to evaluate the particles' position vector, the current image together with its feature points must be rotated so that it has the same orientation as the key image. This requires the rotation matrix $R_{t_n}^{kc}$ between the current and key images, which can be provided using equation (6.73) in section 6.3.4. This rotated image is referred to as the intermediate image.

Suppose there is a set of 2D feature points in the key image for which a set of corresponding feature points in the current image exists, with $N_{k,f}$ being the number of shared feature points. These two sets of feature points are referred to as F_k and F_c, respectively. The set of feature points in the intermediate image is referred to as F_i. Figure 6.18 shows the key, current and intermediate images, together with the corresponding feature points F_k, F_c and F_i.

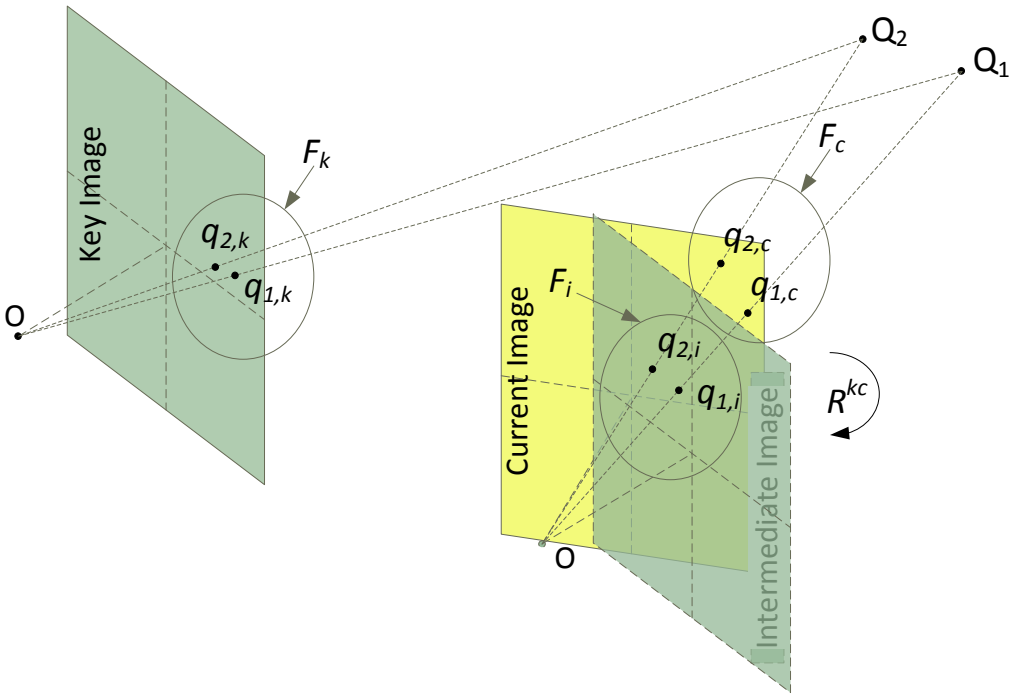


Figure 6.18 Key, current and the intermediate images

Suppose Q_m is a 3D point in the world frame, seen as Q_m^k, Q_m^c and Q_m^i in the key, current and intermediate camera views respectively. These three points have corresponding feature points q_m^k, q_m^c and q_m^i in their respective images which, considering the pinhole camera model, have the following relationship to their 3D counterparts.

(6.111) $\mathbf{q_m^l} = \frac{\mathbf{f}}{\mathbf{Z_m^l}} \mathbf{Q_m^l}$, $\mathbf{m} \in (\mathbf{1} \dots \mathbf{N_{k,f}}), \mathbf{l} \in (\mathbf{i}, \mathbf{k}, \mathbf{c})$

Z_m^l is the distance of the 3D point along the camera axis in the respective reference frame and f is the focal length of the camera. Using the rotation matrix between the current and key camera frames ($R_{t_n}^{kc}$), the coordinates of the Q_m^c in the intermediate frame will be:

(6.112) $\mathbf{Q_m^i} = \mathbf{R_{t_n}^{kc}} \mathbf{Q_m^c}$, $\mathbf{m} \in (\mathbf{1} \dots \mathbf{N_{k,f}})$

Note that the key and intermediate images have the same orientation, therefore $R_{t_n}^{kc} = R_{t_n}^{ic}$. Substituting (6.111) into (6.112) results in the following:

(6.113) $\frac{\mathbf{Z_m^i}}{\mathbf{f}}\ \mathbf{q_m^i} = \frac{\mathbf{Z_m^c}}{\mathbf{f}}\mathbf{R_{t_n}^{kc}}\ \mathbf{q_m^c}$, $\mathbf{m} \in (\mathbf{1} \dots \mathbf{N_{k,f}})$

(6.114) $\mathbf{q_m^i} = \frac{\mathbf{Z_m^c}}{\mathbf{Z_m^i}}\mathbf{R_{t_n}^{kc}}\ \mathbf{q_m^c}$, $\mathbf{m} \in (\mathbf{1} \dots \mathbf{N_{k,f}})$

Z_m^c/Z_m^i is a scaling factor, which affects all three elements of q_m^i in a homogeneous coordinate system in the same way. Therefore it can be omitted from the equation, resulting in the following equation for finding corresponding feature points in the intermediate image.

(6.115) $\mathbf{q_m^i} = \mathbf{R_{t_n}^{kc}}\ \mathbf{q_m^c}$, $\mathbf{m} \in (\mathbf{1} \dots \mathbf{N_{k,f}})$

This equation is used to transform all feature points in F_c to the corresponding feature points in F_i. F_k and F_i are then used to find the $FoE_{t_n}^k$ point between key and intermediate images. This is done by the application of the method outlined in section 6.3.3.1.

According to FoE properties (section 6.3.3.2), the line connecting $FoE_{t_n}^k$ to the centre of camera at key image frame, is parallel to the translation vector. The same concept is used for evaluating the particles, meaning that if a particle position is correct, its $\overrightarrow{OF}$ vector must be parallel to the translation vector associated with the particle. Therefore here, using (6.66) and (6.109), a virtual FoE is defined for the position vector of each particle:

(6.116) $$\widehat{\boldsymbol{FoE}}_{\boldsymbol{p,t_n}}^{\boldsymbol{k}} = \boldsymbol{f}\begin{pmatrix} \frac{\boldsymbol{T_{x,p,t_n}^c}}{\boldsymbol{T_{z,p,t_n}^c}} \\ \frac{\boldsymbol{T_{y,p,t_n}^c}}{\boldsymbol{T_{z,p,t_n}^c}} \end{pmatrix}, \quad \boldsymbol{T_{p,t_n}^c} = \begin{pmatrix} \boldsymbol{T_{x,p,t_n}^c} \\ \boldsymbol{T_{y,p,t_n}^c} \\ \boldsymbol{T_{z,p,t_n}^c} \end{pmatrix}, \quad \boldsymbol{p} \in (\mathbf{1} \dots \boldsymbol{N_p})$$

A particle, whose associated $\widehat{\mathrm{FoE}}_{\mathrm{p,t_n}}^{\mathrm{k}}$ is close to the actual image-based FoE, $\mathrm{FoE}_{\mathrm{t_n}}^{\mathrm{k}}$, is a good representation for the actual current position of the camera. This concept is used to define a score, Γ_p^X, for the position vector of each particle, based on the distance between $\widehat{\mathrm{FoE}}_{\mathrm{p,t_n}}^{\mathrm{k}}$ and $\mathrm{FoE}_{\mathrm{t_n}}^{\mathrm{k}}$ as below.

(6.117) $\boldsymbol{\Gamma_{p}^{X} = \left\| \widehat{\mathbf{FoE}}_{\mathbf{p},\mathbf{t_n}}^{\mathbf{k}} - \mathbf{FoE}_{\mathbf{t_n}}^{\mathbf{k}} \right\|, \quad \mathbf{p} \in (\mathbf{1} \dots \mathbf{N_p})}$

6.4.4.2 Evaluation of the Velocity Vector of a Particle Using FoE

Each particle P_{p,t_n}^{W} also has a velocity vector P_{V,p,t_n}^{W}, which represents the likely velocity of the camera origin with respect to the world frame. The velocity vector of each particle is evaluated using the focus of expansion between the current and past images, captured at time t_n and t_{n-1}, respectively. Referring to section 6.3.3.2 the velocity vector between two adjacent images is parallel to the line connecting the optical centre of the camera to the FoE point, providing that there is no rotation element between the two images. Therefore, in order to evaluate the velocity vector of a particle, the velocity vector must be transformed to the past image reference frame, using $R^{wc_{t_{n-1}}}$ as follows:

(6.118) $\boldsymbol{\mathbf{P}_{\mathbf{V},\mathbf{p},\mathbf{t_n}}^{\mathbf{c}_{\mathbf{t_{n-1}}}} = \mathbf{R}^{\mathbf{c}_{\mathbf{t_{n-1}}}\mathbf{w}} \mathbf{P}_{\mathbf{V},\mathbf{p},\mathbf{t_n}}^{\mathbf{w}} = \mathbf{R}^{\mathbf{wc_{t_{n-1}}}^{-1}} \mathbf{P}_{\mathbf{V},\mathbf{p},\mathbf{t_n}}^{\mathbf{w}}, \ \mathbf{p} \in (\mathbf{1} \dots \mathbf{N_p})}$

In order to use the focus of expansion to evaluate the velocity vector of a particle, the current image together with its feature points must be rotated so that it has the same orientation as the past image. This requires the rotation matrix $R^{c_{t_{n-1}}c_{t_n}}$ between the current and past images, which can be calculated using the current and past rotation matrices, $R^{wc_{t_n}}$ and $R^{wc_{t_{n-1}}}$, which themselves are estimated using equation (6.73) in section 6.3.4.

(6.119) $\boldsymbol{\mathbf{R}^{\mathbf{c}_{\mathbf{t_{n-1}}}\mathbf{c}_{\mathbf{t_n}}} = \mathbf{R}^{\mathbf{wc}_{\mathbf{t_{n-1}}}^{-1}} \mathbf{R}^{\mathbf{wc}_{\mathbf{t_n}}}}$

Suppose there is a set of 2D feature points in the past image for which a set of corresponding feature points in the current image exists, with $N_{c,f}$ being the number of shared feature points. These two sets of feature points are referred to as $F_{c_{t_{n-1}}}$ and $F_{c_{t_n}}$, respectively. The set of feature points in the intermediate image is referred to as F_i.

Suppose Q_m is a 3D point in the world frame, seen as $Q_m^{c_{t_{n-1}}}, Q_m^{c_{t_n}}$ and Q_m^i in the key, current and intermediate camera views, respectively. These three points have corresponding feature points $q_m^{c_{t_{n-1}}}, q_m^{c_{t_n}}$ and q_m^i in their respective images, which considering the pinhole camera model have the following relationship to the 3D counterparts.

(6.120) $\boldsymbol{q_m^l = \frac{f}{Z_m^l} Q_m^l}$, $\boldsymbol{m \in (1 \dots N_{c,f}), l \in (c_{t_{n-1}}, c_{t_n}, i)}$

$\boldsymbol{Z_m^l}$ is the distance of the 3D point along the camera axis in the respective reference frame and f is the focal length of the camera. Referring to (6.115) the feature points in the current image can be transformed to intermediate image using the following equation:

(6.121) $\boldsymbol{q_m^i = R^{c_{t_{n-1}} c_{t_n}} q_m^c}$, $\boldsymbol{m \in (1 \dots N_{c,f})}$

This equation is used to transform all feature points in $F_{c_{t_n}}$ to the corresponding feature points in F_i. $F_{c_{t_{n-1}}}$ and F_i are then used to find the $FoE_{t_n}^c$ point between the past and intermediate images. This is done by the application of the method outlined in section 6.3.3.1.

According to FoE properties (Section 6.3.3.2), the line connecting $FoE_{t_n}^c$ to the camera origin at time t_{n-1}, is parallel to the velocity vector. The same concept is used for evaluating the particles, meaning that if a particle velocity vector is correct, its $\overrightarrow{OF}$ vector must be parallel to the velocity vector associated with the particle. Therefore, using (6.67) and (6.118), a virtual FoE is defined for the velocity vector of each particle:

(6.122) $$\widehat{\mathbf{FoE}}_{\mathbf{p,t_n}}^{\mathbf{c}} = \mathbf{f}\begin{pmatrix} \frac{\mathbf{P}_{\mathbf{V,x,p,t_n}}^{\mathbf{c}}}{\mathbf{P}_{\mathbf{V,z,p,t_n}}^{\mathbf{c}}} \\ \frac{\mathbf{P}_{\mathbf{V,y,p,t_n}}^{\mathbf{c}}}{\mathbf{P}_{\mathbf{V,z,p,t_n}}^{\mathbf{c}}} \end{pmatrix}, \quad \mathbf{P}_{\mathbf{V,p,t_n}}^{\mathbf{c}} = \begin{pmatrix} \mathbf{P}_{\mathbf{V,x,p,t_n}}^{\mathbf{c}} \\ \mathbf{P}_{\mathbf{V,y,p,t_n}}^{\mathbf{c}} \\ \mathbf{P}_{\mathbf{V,z,p,t_n}}^{\mathbf{c}} \end{pmatrix}, \quad \mathbf{p} \in (\mathbf{1} \dots \mathbf{N_p})$$

A particle, which its associated $\widehat{\mathrm{FoE}}_{\mathrm{p,t_n}}^{\mathrm{c}}$, is close to the actual image-based FoE, $\mathrm{FoE}_{\mathrm{t_n}}^{\mathrm{c}}$, is a good representation for the actual current velocity of the camera.

This concept is used to define a score, $\boldsymbol{\Gamma_p^V}$, for the velocity vector of each particle, based on the distance between $\widehat{\mathrm{FoE}}^{\mathrm{c}}_{\mathrm{p,t_n}}$ and $\mathrm{FoE}^{\mathrm{c}}_{\mathrm{t_n}}$ as below.

(6.123) $\boldsymbol{\Gamma_p^V = \left\| \widehat{FoE}^{c}_{p,t_n} - FoE^{c}_{t_n} \right\|, \quad p \in (1 \dots N_p)}$

6.4.5 Weight Assignment

The score values Γ_p^V and Γ_p^X, calculated during particle evaluation (see sections 6.4.4.1 and 6.4.4.2), are used for assigning weights to a particle. Particles in this work have two components, namely; 3D position and speed vectors. The score for the position or velocity vectors of a particle represents the degree of correctness for that particular particle component. When a score is low the associated position or velocity vector is more likely to be a correct vector. On the other hand a high score represents particles, which are less likely to be near the current state of the system. This leads to the definition of the particle weight as the inverse of score value. The position and velocity vectors of a particle are assigned a separate weight as defined below:

(6.124) $\boldsymbol{W_{X,p,t_n} = {1}/{\Gamma_p^X} \quad , \quad p \in (1 \dots N_p)}$

(6.125) $\boldsymbol{W_{V,p,t_n} = {1}/{\Gamma_p^V} \quad , \quad p \in (1 \dots N_p)}$

Once the particle weights are calculated, they are normalised, so that they can be used for weighted averaging.

(6.126) $\boldsymbol{\widetilde{W}^X_{p,t_n} = \frac{W^X_{p,t_n}}{\sum_{p=1}^{N_p} (W^X_{p,t_n})^2} \quad , \quad p \in (1 \dots N_p)}$

(6.127) $\boldsymbol{\widetilde{W}^V_{p,t_n} = \frac{W^V_{p,t_n}}{\sum_{p=1}^{N_p} (W^V_{p,t_n})^2} \quad , \quad p \in (1 \dots N_p)}$

6.4.6 Re-sampling

Some particles have a low weight and do not contribute much to the final state estimation. Such particles must be replaced by particles with higher value

weights. To do so, the number of effective particles is calculated using the method outlined in section 6.4.2.1.

(6.128) $$\boldsymbol{N_{eff}^{X}} = \frac{\mathbf{1}}{\sum_{p=1}^{N_p}(\widetilde{\boldsymbol{W}}_{p,t_n}^{X})^2}$$

(6.129) $$\boldsymbol{N_{eff}^{V}} = \frac{\mathbf{1}}{\sum_{p=1}^{N_p}(\widetilde{\boldsymbol{W}}_{p,t_n}^{V})^2}$$

N_{eff}^{X} and N_{eff}^{V} are the number of effective particles for the position and velocity vectors of particles. A threshold level (e.g 70%) is defined to determine the adequacy of the number of effective particles. If the number of effective particles is less than the threshold level, the resampling algorithm is executed.

6.4.7 State Estimation

The state of the system is calculated using a weighted summation of system particles as follows:

(6.130) $$\begin{pmatrix} \boldsymbol{X_{t_n}^{w}} \\ \boldsymbol{V_{t_n}^{w}} \end{pmatrix} = \begin{pmatrix} \sum_{p=1}^{N_p} \widetilde{\boldsymbol{W}}_{p,t_n}^{X} \boldsymbol{P}_{X,p,t_n}^{w} \\ \sum_{p=1}^{N_p} \widetilde{\boldsymbol{W}}_{p,t_n}^{V} \boldsymbol{P}_{V,p,t_n}^{w} \end{pmatrix}$$

6.4.8 Past State Correction

Every new image, together with the IMU data between the current and past images, can provide valuable information about the past state of the system. This information can be used to update the past system state, which in turn can influence the current system state through the state-space model. The properties of epipolar geometry for continuous motion are used to serve this purpose. This section provides an overview of the relevant background information and presents a solution for correcting the past state of the system based on the properties of epipolar geometry for continuous motion.

6.4.8.1 Continuous Motion and Epipolar Geometry

Static points in 3D space, from the point of view of a moving camera, appear to have a 3D motion. Suppose Q^{w} is a fixed point with homogenous coordinates with respect to the world frame. As explained earlier, the same point has a different coordinate vector with respect to the camera frame, Q^{c}, which relates to the world frame representation by a transformation matrix G^{cw}:

(6.131) $\boldsymbol{Q^c = G^{cw} Q^w}$

(6.132) $\boldsymbol{G^{cw}} = \begin{pmatrix} \boldsymbol{R^{cw}} & \boldsymbol{T^{cw}} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \in \mathbb{R}^{\mathbf{4\times4}}$

(6.133) $\boldsymbol{G^{cw-1}} = \begin{pmatrix} \boldsymbol{R^{cwT}} & \boldsymbol{-R^{cwT}T^{cw}} \\ \mathbf{0} & \mathbf{1} \end{pmatrix}$

Computing the time derivative of equation (6.131) (note that Q^w is a constant value vector) and substituting into equation (6.131) results in the following:

(6.134) $\boldsymbol{\dot{Q}^c = \dot{G}^{cw} Q^w}$

(6.135) $\dot{\mathbf{Q}}^{\mathbf{c}} = \dot{\mathbf{G}}^{\mathbf{cw}}\mathbf{G}^{\mathbf{cw-1}}\,\mathbf{Q}^{\mathbf{c}}$

In order to simplify the equation, the superscripts are omitted. The transformation matrix is then replaced by the equivalent as follows:

(6.136) $\dot{\mathbf{Q}}^{\mathbf{c}} = \dot{\mathbf{G}}\,\mathbf{G}^{-\mathbf{1}}\mathbf{Q}^{\mathbf{c}}$

(6.137) $\dot{\mathbf{G}}\,\mathbf{G}^{-\mathbf{1}} = \begin{pmatrix} \dot{\mathbf{R}}\mathbf{R}^{\mathbf{T}} & \dot{\mathbf{T}} - \dot{\mathbf{R}}\mathbf{R}^{\mathbf{T}}\mathbf{T} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$

It can be shown that $\dot{R}R^T$ is the angular velocity skew-symmetric matrix and $\dot{T} - \dot{R}R^T T$ is the linear velocity vector (Ma, et al., 2004).

(6.138) $\hat{\boldsymbol{\omega}}(\mathbf{t}) = \dot{\mathbf{R}}\mathbf{R}^{\mathbf{T}} = \begin{pmatrix} \mathbf{0} & -\boldsymbol{\omega}_{\mathbf{z}} & \boldsymbol{\omega}_{\mathbf{y}} \\ \boldsymbol{\omega}_{\mathbf{z}} & \mathbf{0} & -\boldsymbol{\omega}_{\mathbf{x}} \\ -\boldsymbol{\omega}_{\mathbf{y}} & \boldsymbol{\omega}_{\mathbf{x}} & \mathbf{0} \end{pmatrix}$

(6.139) $\mathbf{v}(\mathbf{t}) = \dot{\mathbf{T}} - \hat{\boldsymbol{\omega}}(\mathbf{t})\mathbf{T} = \begin{pmatrix} \mathbf{v}_{\mathbf{x}} \\ \mathbf{v}_{\mathbf{y}} \\ \mathbf{v}_{\mathbf{z}} \end{pmatrix}$

Therefore (6.136) is re-written as follows:

(6.140) $\dot{\mathbf{Q}} = \begin{pmatrix} \widehat{\boldsymbol{\omega}}(\mathbf{t}) & \mathbf{v}(\mathbf{t}) \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \mathbf{Q}$

Here Q is redefined to be a vector with 3 elements, consequently the above equation becomes:

(6.141) $\dot{\mathbf{Q}}(\mathbf{t}) = \widehat{\boldsymbol{\omega}}(\mathbf{t})\mathbf{Q}(\mathbf{t}) + \mathbf{v}(\mathbf{t})$

6.4.8.2 A Method for Past State Correction

The epipolar geometry for continuous motion and its associated equation (6.141) form the basis of the past state correction method proposed here. Suppose $\mathrm{Q} = (\mathrm{X}, \mathrm{Y}, \mathrm{Z})^{\mathrm{T}}$ is a static point in 3D space. Equation (6.141) is expanded as follows:

(6.142) $\dot{\boldsymbol{Q}} = \begin{pmatrix} \dot{\boldsymbol{X}} \\ \dot{\boldsymbol{Y}} \\ \dot{\boldsymbol{Z}} \end{pmatrix} = \begin{pmatrix} \mathbf{0} & -\boldsymbol{\omega_z} & \boldsymbol{\omega_y} \\ \boldsymbol{\omega_z} & \mathbf{0} & -\boldsymbol{\omega_x} \\ -\boldsymbol{\omega_y} & \boldsymbol{\omega_x} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \\ \boldsymbol{Z} \end{pmatrix} + \begin{pmatrix} \boldsymbol{v_x} \\ \boldsymbol{v_y} \\ \boldsymbol{v_z} \end{pmatrix}$

(6.143) $\begin{pmatrix} \dot{\boldsymbol{X}} \\ \dot{\boldsymbol{Y}} \\ \dot{\boldsymbol{Z}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{v_x} - \boldsymbol{\omega_z Y} + \boldsymbol{\omega_y Z} \\ \boldsymbol{v_y} + \boldsymbol{\omega_z X} - \boldsymbol{\omega_x Z} \\ \boldsymbol{v_z} - \boldsymbol{\omega_y X} + \boldsymbol{\omega_x Y} \end{pmatrix}$

Note that the reference to *t* is dropped for simplicity. Assuming that $(q_x, q_y)^T$ and $(x, y)^T$ are the corresponding un-calibrated and calibrated pixel coordinates of a 3D point Q:

(6.144) $\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \frac{\mathbf{1}}{\boldsymbol{z}} \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{pmatrix} = \boldsymbol{f K^{-1}} \begin{pmatrix} \boldsymbol{q_x} \\ \boldsymbol{q_y} \end{pmatrix}$

Without loss of generality the image plane is considered to be at $f = 1$. Now, using the image velocity vector $(u, v)^T$ the above equation is formulated as follows:

(6.145) $\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} = \begin{pmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{y}} \end{pmatrix}$

Considering (6.144) the above equation can be expressed as follows:

(6.146) $\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} = \frac{\mathbf{1}}{\boldsymbol{Z}} \begin{pmatrix} \dot{\boldsymbol{X}} \\ \dot{\boldsymbol{Y}} \end{pmatrix} - \frac{\dot{\boldsymbol{Z}}}{\boldsymbol{Z}^{\mathbf{2}}} \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{pmatrix}$

Substituting equations (6.143) into (6.146) results in the equation (6.147). This equation is decomposed into translational and rotational components as per (6.148). The rotational component solely depends on the angular velocity and pixel information. The translational element only depends on the depth and linear velocity.

(6.147) $\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} = \frac{\mathbf{1}}{\boldsymbol{Z}} \begin{pmatrix} \boldsymbol{v_x} - \boldsymbol{\omega_z Y} + \boldsymbol{\omega_y Z} \\ \boldsymbol{v_y} + \boldsymbol{\omega_z X} - \boldsymbol{\omega_x Z} \end{pmatrix} - \frac{\boldsymbol{v_z} - \boldsymbol{\omega_y X} + \boldsymbol{\omega_x Y}}{\boldsymbol{Z}^{\mathbf{2}}} \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{pmatrix}$

(6.148) $\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u_T} \\ \boldsymbol{v_T} \end{pmatrix} + \begin{pmatrix} \boldsymbol{u_R} \\ \boldsymbol{v_R} \end{pmatrix}$

(6.149) $\begin{pmatrix} \boldsymbol{u_T} \\ \boldsymbol{v_T} \end{pmatrix} = \begin{pmatrix} \frac{\boldsymbol{v_x} - \boldsymbol{v_z x}}{\boldsymbol{Z}} \\ \frac{\boldsymbol{v_y} - \boldsymbol{v_z y}}{\boldsymbol{Z}} \end{pmatrix}$

(6.150) $\begin{pmatrix} \boldsymbol{u_R} \\ \boldsymbol{v_R} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\omega_y} - \boldsymbol{\omega_z y} - \boldsymbol{\omega_x xy} + \boldsymbol{\omega_y x^2} \\ -\boldsymbol{\omega_x} + \boldsymbol{\omega_z x} + \boldsymbol{\omega_y xy} - \boldsymbol{\omega_x y^2} \end{pmatrix}$

By knowing the angular velocity and the 2D coordinates of the 3D point, u_R and v_R can easily be calculated. Vector $(u, v)^T$, the image velocity vector, is also known from optical flow between the two consecutive images (see section 6.3.3). Consequently the translational component for each feature point can be formulated as follows:

(6.151) $\begin{pmatrix} \boldsymbol{u_{m,T}} \\ \boldsymbol{v_{m,T}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u_m} \\ \boldsymbol{v_m} \end{pmatrix} - \begin{pmatrix} \boldsymbol{u_{m,R}} \\ \boldsymbol{v_{m,R}} \end{pmatrix}, \ \boldsymbol{m} \in (\mathbf{1} \dots \boldsymbol{N_f})$

On the other hand, considering equation (6.149), vector $(u_{m,T}, v_{m,T})^T$ for each feature point $m \in (1 \dots N_f)$ can be expressed as follows:

(6.152) $\begin{pmatrix} \boldsymbol{u_{m,T}} \\ \boldsymbol{v_{m,T}} \end{pmatrix} = \begin{pmatrix} \frac{\frac{\boldsymbol{v_x}}{\boldsymbol{v_z}} - \boldsymbol{x_m}}{\boldsymbol{Z_m}} . \boldsymbol{v_z} \\ \frac{\frac{\boldsymbol{v_y}}{\boldsymbol{v_z}} - \boldsymbol{y_m}}{\boldsymbol{Z_m}} . \boldsymbol{v_z} \end{pmatrix}, \boldsymbol{m} \in (\mathbf{1} \dots \boldsymbol{N_f})$

Taking into account the properties of FoE, and equation (6.67), it is apparent that $\left(\frac{v_x}{v_z}, \frac{v_y}{v_z}\right)^T$ is in fact the coordinate vector of $FoE^c_{t_{n-1}} = (x_{FoE^c_{t_{n-1}}}, y_{FoE^c_{t_{n-1}}})$. Therefore (6.152) can be formulated as follows:

$$(6.153)\ \begin{pmatrix} \boldsymbol{u_{m,T}} \\ \boldsymbol{v_{m,T}} \end{pmatrix} = \begin{pmatrix} \frac{x_{FoE^c_{t_{n-1}}} - x_m}{Z_m} \boldsymbol{v_z} \\ \frac{y_{FoE^c_{t_{n-1}}} - y_m}{Z_m} \boldsymbol{v_z} \end{pmatrix}, \quad \boldsymbol{m} \in (\boldsymbol{1} \dots \boldsymbol{N_f})$$

$u_{m,T}$ and $v_{m,T}$ are derived from (6.152). x_m and y_m are the coordinates of the feature points tracked by optical flow at time t_n, and therefore known. Consequently all parameters of equation (6.153) are known, except for Z, which leads to the following equation for deriving Z.

$$(6.154)\ \begin{pmatrix} \boldsymbol{Z_{m,1}} \\ \boldsymbol{Z_{m,2}} \end{pmatrix} = -\begin{pmatrix} \frac{x_{FoE^c_{t_{n-1}}} - x_m}{u_{m,T}} \boldsymbol{v_z} \\ \frac{y_{FoE^c_{t_{n-1}}} - y_m}{v_{m,T}} \boldsymbol{v_z} \end{pmatrix}, \quad \boldsymbol{m} \in (\boldsymbol{1} \dots \boldsymbol{N_f})$$

In theory $Z_{m,1}$ and $Z_{m,2}$ should be the same; however in practice due to feature detection error these values may differ, therefore Z is considered to be the average of the two values:

$$(6.155)\ \boldsymbol{Z_{m,t_{n-1}}} = \frac{\boldsymbol{Z_{m,1}} + \boldsymbol{Z_{m,2}}}{\boldsymbol{2}}$$

The Z value can only be calculated for the previous sampling time, t_{n-1}, but not the current one, t_n. This is not useful for estimating the current state of the system, however it gives the system a chance to correct for the accumulated error over time.

Using the calibrated pixel coordinates of a feature point $(x_{m,t_{n-1}}, y_{m,t_{n-1}})^T$ at time t_{n-1} and the depth $Z_{m,t_{n-1}}$ as per (6.155), the 3D coordinate vector of point Q_m can be estimated as follows:

(6.156) $$\widehat{\boldsymbol{Q}}_{\boldsymbol{m}}^{\boldsymbol{c}_{t_{n-1}}} = \boldsymbol{Z}_{\boldsymbol{m},\boldsymbol{t}_{n-1}} \begin{pmatrix} \boldsymbol{x}_{\boldsymbol{m},\boldsymbol{t}_{n-1}} \\ \boldsymbol{y}_{\boldsymbol{m},\boldsymbol{t}_{n-1}} \\ \mathbf{1} \end{pmatrix}$$

The same technique for calculating depth is also applied to the key image:

(6.157) $$\widehat{\boldsymbol{Q}}_{\boldsymbol{m}}^{\boldsymbol{k}} = \boldsymbol{Z}_{\boldsymbol{m},\boldsymbol{t}_{\boldsymbol{k}}} \begin{pmatrix} \boldsymbol{x}_{\boldsymbol{m},\boldsymbol{t}_{\boldsymbol{k}}} \\ \boldsymbol{y}_{\boldsymbol{m},\boldsymbol{t}_{\boldsymbol{k}}} \\ \mathbf{1} \end{pmatrix}$$

Referring again to the concept of rigid body transformation (see section 6.1), the transformation equation between the key image and the image at time t_{n-1} is:

(6.158) $$\widehat{\boldsymbol{Q}}_{\boldsymbol{m}}^{\boldsymbol{k}} = \widehat{\boldsymbol{T}}_{\boldsymbol{m},\boldsymbol{t}_{n-1}}^{\boldsymbol{kC}_{t_{n-1}}} + \boldsymbol{R}_{\boldsymbol{t}_{n-1}}^{\boldsymbol{kC}_{t_{n-1}}} \widehat{\boldsymbol{Q}}_{\boldsymbol{m}}^{\boldsymbol{c}_{t_{n-1}}}, \quad \boldsymbol{m} \in (\mathbf{1} \dots \boldsymbol{N}_{\boldsymbol{f}})$$

$\mathrm{R}_{\mathrm{t_n}}^{\mathrm{kC_{t_{n-1}}}}$ is known using the method outlined in section 6.3.4. Therefore by combining equations (6.156) to (6.158), the translation vector at the time of past image ($\mathrm{t_{n-1}}$) for each feature point can be estimated as follows:

(6.159) $$\widehat{\mathbf{T}}_{\mathbf{m},\mathbf{t_{n-1}}}^{\mathbf{kC_{t_{n-1}}}} = \widehat{\mathbf{Q}}_{\mathbf{m}}^{\mathbf{k}} - \mathbf{R}_{\mathbf{t_{n-1}}}^{\mathbf{kC_{t_{n-1}}}} \widehat{\mathbf{Q}}_{\mathbf{m}}^{\mathbf{c_{t_{n-1}}}}, \quad \mathbf{m} \in (\mathbf{1} \dots \mathbf{N_f})$$

In theory $\hat{T}_{m,t_{n-1}}^{kC_{t_{n-1}}}$ should be the same for all feature points, however, due to noise and error in the system in practice these vectors may differ. Therefore the translation vectors $\hat{T}_{m,t_{n-1}}^{kC_{t_{n-1}}}$ are treated like the position vector of particles and evaluated for correctness using the properties of FoE (see section 6.4.4.1). As $\hat{T}_{m,t_{n-1}}^{kC_{t_{n-1}}}$ is a possible solution for the translation vector between the key and past images, its associated FoE must be close to $FoE_{t_{n-1}}^{k}$. According to (6.66) the associated FoE for $\hat{T}_{m,t_{n-1}}^{kC_{t_{n-1}}}$ is defined as:

(6.160) $$\widehat{\boldsymbol{FoE}}^{\boldsymbol{k}}_{\boldsymbol{m},\boldsymbol{t_{n-1}}} = \begin{pmatrix} \frac{\widehat{\boldsymbol{T}}^{\boldsymbol{kC_{t_{n-1}}}}_{\boldsymbol{x},\boldsymbol{m},\boldsymbol{t_{n-1}}}}{\widehat{\boldsymbol{T}}^{\boldsymbol{kC_{t_{n-1}}}}_{\boldsymbol{z},\boldsymbol{m},\boldsymbol{t_{n-1}}}} \\ \frac{\widehat{\boldsymbol{T}}^{\boldsymbol{kC_{t_{n-1}}}}_{\boldsymbol{y},\boldsymbol{m},\boldsymbol{t_{n-1}}}}{\widehat{\boldsymbol{T}}^{\boldsymbol{kC_{t_{n-1}}}}_{\boldsymbol{z},\boldsymbol{m},\boldsymbol{t_{n-1}}}} \end{pmatrix}$$

where

$$\widehat{\boldsymbol{T}}^{\boldsymbol{kC_{t_{n-1}}}}_{\boldsymbol{m},\boldsymbol{t_{n-1}}} = \begin{pmatrix} \widehat{\boldsymbol{T}}^{\boldsymbol{kC_{t_{n-1}}}}_{\boldsymbol{x},\boldsymbol{m},\boldsymbol{t_{n-1}}} \\ \widehat{\boldsymbol{T}}^{\boldsymbol{kC_{t_{n-1}}}}_{\boldsymbol{y},\boldsymbol{m},\boldsymbol{t_{n-1}}} \\ \widehat{\boldsymbol{T}}^{\boldsymbol{kC_{t_{n-1}}}}_{\boldsymbol{z},\boldsymbol{m},\boldsymbol{t_{n-1}}} \end{pmatrix}, \quad \boldsymbol{m} \in (\boldsymbol{1} \dots \boldsymbol{N_f})$$

$FoE^k_{t_{n-1}}$ has been calculated at time t_{n-1} and is known. The distance between $\widehat{FoE}^k_{m,t_{n-1}}$ and $FoE^k_{t_{n-1}}$ serves as a score for $\hat{T}^{kC_{t_{n-1}}}_{m,t_n}$.

(6.161) $$\boldsymbol{\Gamma^T_m} = \left\| \widehat{\boldsymbol{FoE}}^{\boldsymbol{k}}_{\boldsymbol{m},\boldsymbol{t_{n-1}}} - \boldsymbol{FoE}^{\boldsymbol{k}}_{\boldsymbol{t_{n-1}}} \right\|, \quad \boldsymbol{m} \in (\boldsymbol{1} \dots \boldsymbol{N_f})$$

A weight is then assigned to each feature point and the result is normalised.

(6.162) $$\boldsymbol{W^T_{m,t_{n-1}}} = {}^{\boldsymbol{1}}\!/_{\boldsymbol{\Gamma^T_m}} \quad , \quad \boldsymbol{m} \in (\boldsymbol{1} \dots \boldsymbol{N_f})$$

(6.163) $$\widetilde{\boldsymbol{W}}^{\boldsymbol{T}}_{\boldsymbol{m},\boldsymbol{t_{n-1}}} = \frac{\boldsymbol{W^T_{m,t_{n-1}}}}{\sum_{\boldsymbol{m=1}}^{\boldsymbol{N_f}} \left(\boldsymbol{W^T_{m,t_{n-1}}}\right)^{\boldsymbol{2}}} \quad , \quad \boldsymbol{m} \in (\boldsymbol{1} \dots \boldsymbol{N_f})$$

Finally the correct translation vector between the reference frame and previous image frame is estimated using the weighted summation:

(6.164) $$\boldsymbol{T}^{\boldsymbol{kC_{t_{n-1}}}}_{\boldsymbol{m},\boldsymbol{t_{n-1}}} = \sum_{\boldsymbol{m=1}}^{\boldsymbol{N_f}} \widetilde{\boldsymbol{W}}^{\boldsymbol{T}}_{\boldsymbol{m},\boldsymbol{t_n}} \widehat{\boldsymbol{T}}^{\boldsymbol{kC_{t_{n-1}}}}_{\boldsymbol{m},\boldsymbol{t_{n-1}}}$$

$\boldsymbol{T}^{\boldsymbol{kC_{t_{n-1}}}}_{\boldsymbol{m},\boldsymbol{t_n}}$ is a vector with respect to the key image reference frame. Therefore the estimated position of the camera at time t_{n-1} with respect to the world frame is:

(6.165) $$\widehat{\boldsymbol{X}}^{\boldsymbol{w}}_{\boldsymbol{t_{n-1}}} = \boldsymbol{T}^{\boldsymbol{wk}}_{\boldsymbol{t_k}} + \boldsymbol{R}^{\boldsymbol{wk}}_{\boldsymbol{t_n}} \boldsymbol{T}^{\boldsymbol{kC_{t_{n-1}}}}_{\boldsymbol{m},\boldsymbol{t_n}}$$

6.4.9 A Mechanism for Selecting the Method of Tracking

In this chapter a solution for hybrid tracking using fused IMU and vision data has been presented. However, when the vision system cannot provide sufficient accurate tracking information, the hybrid system will no longer be suitable and IMU-based tracking must be used on its own until the current circumstances are changed by the arrival of new images. Furthermore, the past state correction method described in section 6.4.8 does not always provide a reliable outcome, therefore certain criteria need to be met before the past state is corrected. These issues have led to the design of a mechanism and definition of a number of criteria to determine the best method for tracking at any particular time.

6.4.9.1 Criteria for Determining the Method of Tracking

The criteria for determining the method of tracking are mainly related to the quality of the FoE. This section provides a list of criteria, which must be met before hybrid tracking or past state correction is applied, otherwise the system continues with IMU-only tracking until a new image becomes available and criteria for hybrid tracking are satisfied.

6.4.9.1.1 Angle of Motion

In order to effectively evaluate position and velocity vectors of particles, $FoE_{t_n}^{k}$ and $FoE_{t_n}^{c}$ must exist (see section 6.4.4). Referring to section 6.3.3.3.1, when the translation vector, $T_{t_n}^{w,kc}$, between the key and current images is parallel to the key image plane, $FoE_{t_n}^{k}$ does not exist. To check for this condition, the angle between the key image plane and $T_{t_n}^{w,kc}$ is estimated using equation (6.68) as follows.

(6.166) $$\boldsymbol{\theta_{T,t_n} = \frac{\pi}{2} - cos^{-1} \frac{\widehat{T}_{t_n}^{w,kc} N_{t_k}^{w}}{\left\| \widehat{T}_{t_n}^{w,kc} \right\|}} \quad , \quad \boldsymbol{\widehat{T}_{t_n}^{w,kc} = X_{t_{n-1}+N_i \delta t}^{w} - X_{t_k}^{w}}$$

$T_{t_n}^{w,kc}$ is the translation vector between the current and key images with respect to the world frame. Note that this angle is calculated before the application of hybrid tracking and therefore the only available source of tracking information for the camera at the time is $X_{t_{n-1}+N_i\delta t}^{w}$ estimated by the IMU using equation (6.49) $X_{t_k}^{w}$ is the camera pose at the time of the key image acquisition and $N_{t_k}^{w}$ is the key image unit normal vector with respect to the world frame.

Similarly when the velocity vector, $v_{t_n}^{w}$, is parallel to the past image plane, $FoE_{t_n}^{c}$ does not exist. To check for this condition, the angle between the past image plane and $v_{t_n}^{w}$ is estimated using equation (6.69) as follows.

(6.167) $$\boldsymbol{\theta_{v,t_n}} = \frac{\boldsymbol{\pi}}{\boldsymbol{2}} - \boldsymbol{cos^{-1}} \frac{\widehat{\boldsymbol{v}}_{\boldsymbol{t_n}}^{\boldsymbol{w}} \boldsymbol{N}_{\boldsymbol{t_{n-1}}}^{\boldsymbol{w}}}{\|\widehat{\boldsymbol{v}}_{\boldsymbol{t_n}}^{\boldsymbol{w}}\|} \quad , \quad \widehat{\boldsymbol{v}}_{\boldsymbol{t_n}}^{\boldsymbol{w}} = \boldsymbol{V}_{\boldsymbol{t_{n-1}}+\boldsymbol{N_i}\boldsymbol{\delta t}}^{\boldsymbol{w}}$$

$\mathrm{V}_{\mathrm{t_{n-1}}+\mathrm{N_i}\delta\mathrm{t}}^{\mathrm{w}}$ is the camera speed estimated by the IMU using equation (6.49) $\mathrm{N}_{\mathrm{t_{n-1}}}^{\mathrm{w}}$is the past image unit normal vector with respect to the world frame. $\theta_{\mathrm{T,t_n}}$ and $\theta_{\mathrm{v,t_n}}$ angles greater than a threshold level, θ_{th}, indicate that the associated FoE, $\mathrm{FoE}_{\mathrm{t_n}}^{\mathrm{k}}$or $\mathrm{FoE}_{\mathrm{t_n}}^{\mathrm{c}}$ exists, therefore hybrid tracking can be used.

6.4.9.1.2 Distance between FoE and Flow Lines

Another measure used for evaluating the effectiveness of hybrid tracking is the average distance between the estimated FoE point and the flow lines (see section 6.3.3.3.2). Equation (6.70) is used to provide the average distances, $d_{FoE_{t_n}^{k}}$ for $FoE_{t_n}^{k} = (x_{FoE_{t_n}^{k}}, y_{FoE_{t_n}^{k}})^T$ and $d_{FoE_{t_n}^{c}}$ for $FoE_{t_n}^{c} = (x_{FoE_{t_n}^{c}}, y_{FoE_{t_n}^{c}})^T$.

(6.168) $$\boldsymbol{d_{FoE_{t_n}^{k}}} = \frac{\boldsymbol{1}}{\boldsymbol{m}} \sqrt{\sum_{\boldsymbol{m=1}}^{\boldsymbol{N_{k,f}}} \frac{\left(\boldsymbol{a_{k,m}} \boldsymbol{x_{FoE_{t_n}^{k}}} + \boldsymbol{b_{k,m}} \boldsymbol{y_{FoE_{t_n}^{k}}} + \boldsymbol{c_{k,m}}\right)^2}{\boldsymbol{a_{k,m}^2} + \boldsymbol{b_{k,m}^2}}}$$

(6.169) $$\boldsymbol{d_{FoE_{t_n}^{c}}} = \frac{\boldsymbol{1}}{\boldsymbol{m}} \sqrt{\sum_{\boldsymbol{m=1}}^{\boldsymbol{N_{c,f}}} \frac{\left(\boldsymbol{a_{c,m}} \boldsymbol{x_{FoE_{t_n}^{c}}} + \boldsymbol{b_{c,m}} \boldsymbol{y_{FoE_{t_n}^{c}}} + \boldsymbol{c_{c,m}}\right)^2}{\boldsymbol{a_{c,m}^2} + \boldsymbol{b_{c,m}^2}}}$$

$(a_{k,m}, b_{k,m}, c_{k,m})$ and $(a_{c,m}, b_{c,m}, c_{c,m})$ are the flow line parameters for each feature point used for estimating $\mathrm{FoE}_{t_n}^{k}$ and $\mathrm{FoE}_{t_n}^{c}$. $N_{k,f}$ and $N_{c,f}$ are the number of common feature points between the current image and key or past images, respectively.

In order to have a reliable hybrid tracking system both $d_{\mathrm{FoE}_{t_n}^{k}}$ and $d_{\mathrm{FoE}_{t_n}^{c}}$ must be smaller than a threshold value, d_{th1}. Otherwise the hybrid tracking is not effective and must be bypassed. This threshold is defined in terms of percentage of the focal length and is determined based on the performance requirements of the tracking system.

6.4.9.1.3 Criteria for Past State Correction

Provided that the FoE exists and has good quality, as explained in previous two sections, the hybrid tracking can be applied. However past state correction requires more stringent criteria to be met.

a) The threshold for acceptability of quality of $FoE_{t_{n-1}}^{c}$ based on $d_{FoE_{t_{n-1}}^{c}}$ is defined at a much lower level, d_{th2}.
b) $FoE_{t_{n-1}}^{c}$ is compared with the IMU-based FoE, which is determined as follows:

(6.170)

$$\mathbf{FoE}_{IMU,t_{n-1}}^{c} = \begin{pmatrix} \frac{\hat{v}_{x,t_{n-1}}^{c_{t_{n-1}}}}{\hat{v}_{z,t_{n-1}}^{c_{t_{n-1}}}} \\ \frac{\hat{v}_{y,t_{n-1}}^{c_{t_{n-1}}}}{\hat{v}_{z,t_{n-1}}^{c_{t_{n-1}}}} \end{pmatrix}, \quad \hat{\boldsymbol{v}}_{t_n}^{c_{t_{n-1}}} = \begin{pmatrix} \hat{v}_{x,t_{n-1}}^{c_{t_{n-1}}} \\ \hat{v}_{y,t_{n-1}}^{c_{t_{n-1}}} \\ \hat{v}_{z,t_{n-1}}^{c_{t_{n-1}}} \end{pmatrix} = \boldsymbol{R}^{wc_{t_{n-1}}{}^{-1}} \boldsymbol{V}_{t_{n-2}+N_i\delta t}^{w}$$

If the distance between $FoE_{t_{n-1}}^{c}$ and $\mathrm{FoE}_{IMU,t_{n-1}}^{c}$ is less than a threshold level, d_{th3}, $FoE_{t_{n-1}}^{c}$ is considered to have sufficient quality for the application of past state correction algorithm.

c) The difference between $Z_{m,1}$ and $Z_{m,2}$ must be small. This is checked by calculating the following parameter and comparing with a threshold

(6.171) $$\boldsymbol{\alpha_{t_{n-1}}} = \frac{\mathbf{1}}{\mathbf{N_f}} \sum_{\mathbf{m=1}}^{\mathbf{N_f}} \frac{\mathbf{Z_{m,1} - Z_{m,2}}}{\mathbf{Z_{m,1} + Z_{m,2}}}$$

6.5 Summary

This chapter presents the system architecture for a hybrid inertial-visual camera pose tracking system. A stochastic data fusion method has been employed for fusing IMU and vision data. This is based on particle filtering, where particles are selected via state-space model, kinematic motion equations and IMU characteristics. The particle evaluation is carried out using the properties of focus of expansion. In addition, a past state correction mechanism has been incorporated, which operates when new reliable image information become available. These concepts have been formulated in this chapter in a way to suit their intended application and designed algorithms.

The focus of this chapter is on pose estimation when a new image is captured. The pose information between two images is provided by the IMU data and state-space model as outlined in section 6.2.1. The algorithm for estimating the pose when a new image is captured is summarised as follows:

a) The past state correction criteria are evaluated and if applicable the state of the system at the time of the past image is updated (see section 6.4.8). Then the past particles are set to the updated past state.
b) If the image-based FoE has a good quality, particle filtering is employed to estimate the current state of the system, otherwise the IMU-based estimation is used and the following steps are skipped.
c) New particles are generated (see section 6.4.16.4.3)

d) The particles are evaluated (see section 6.4.4)
e) A weight is assigned to each particle (see section 6.4.5)
f) Re-sampling is applied if necessary (see section 6.4.6)
g) The current state of the system is estimated using the weighted summation of all particles (see section 6.4.6)
h) If the number of feature points in the current image is no longer adequate for a robust tracking, the key image is replaced by a more recent image (see section 6.3.5)

CHAPTER SEVEN

7 Results and Analysis

The hybrid tracking system proposed in previous chapters operates based on receiving sensory information from IMU and vision sensors and then fusing tracking data using focus of expansion and particle-filtering. In order to evaluate the performance of the camera tracking system, a set of IMU and vision sensory data needed to be presented to the system and the results compared with a set of ground truth pose and orientation data. This chapter sets out the framework for system validation and provides the tracking outcome and analysis using both synthetic and real data. System implementation, data generation and performance analysis were carried out using MATLAB.

7.1 Generation of Synthetic Data

In order to evaluate the performance of the system outlined in chapter 6, a set of simulation data was devised to be considered as the ground truth. This included a 3D trajectory for the camera position, a set of data for camera orientation and a set of feature points to be used by the image-based system.

7.1.1 Feature Points

The focus of the algorithm proposed in this thesis is on tracking and data fusion. Although feature tracking is an integral part of the system, the methods employed for this purpose are well established and widely used. Therefore in

this simulation a set of feature points with pre-defined 3D locations has been assumed. The camera parameters considered in this simulation were based on a typical 5MP camera such as the OV5650 camera used in the iPad4. The camera has been assumed to operate in a typical 720p mode at a maximum rate of 60fps. The 3D feature points were defined as in (7.1) using parameters a, b, c specified in (7.2). Figure 7.1 shows the 3D points. The feature points were placed in three layers at the height of 3, 4 and 5 meters. Each layer consists of 169 feature points, each 0.5mx0.5m apart, in an array of 13x13 feature points. The feature points have been arranged in such a way that their density increases from layer 1 to layer 3.

(7.1) $$\boldsymbol{F} = \begin{pmatrix} \boldsymbol{a}/(\boldsymbol{c}-\boldsymbol{2}) \\ \boldsymbol{b}/(\boldsymbol{c}-\boldsymbol{2}) \\ \boldsymbol{c} \end{pmatrix}$$

(7.2) $$\boldsymbol{a} \in (-\boldsymbol{2m}:\boldsymbol{0.5m}:\boldsymbol{4m}),\ \ \boldsymbol{b} \in (-\boldsymbol{2m}:\boldsymbol{0.5m}:\boldsymbol{4m}),\ \ \boldsymbol{c} \in (\boldsymbol{3m}:\boldsymbol{1m}:\boldsymbol{5m})$$

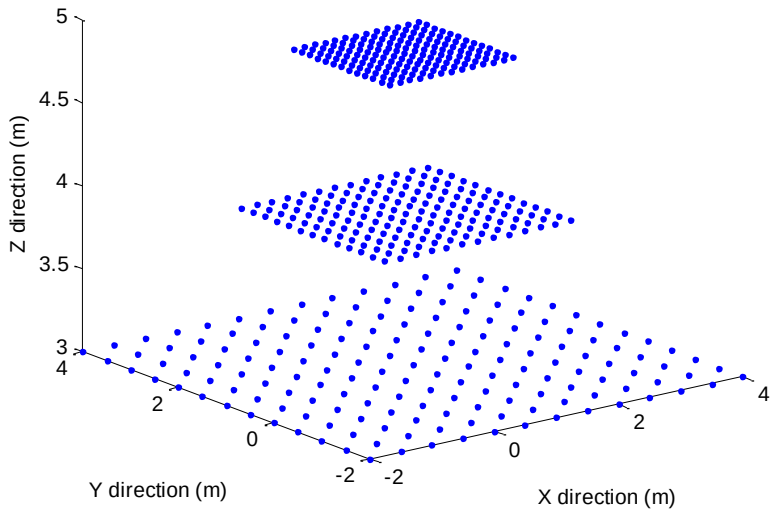


Figure 7.1 : 3D feature points in space

7.1.2 3D Position

Based on the concept of Fourier transform, a continuous signal can be written as a series of sinusoid signals. Therefore the camera position was assumed to

be a sinusoid waveform in every direction with different amplitude, frequency and phase values for the three degrees of freedom as expressed by (7.3).

$$X_t^w = \begin{pmatrix} \rho_x(1 - cos(2\pi f_x t + \varphi_x)) \\ \rho_y(1 - cos(2\pi f_y t + \varphi_y)) \\ \rho_z(1 - cos(2\pi f_z t + \varphi_z)) \end{pmatrix} \tag{7.3}$$

$$\begin{pmatrix} \rho_x \\ \rho_y \\ \rho_z \end{pmatrix} = \begin{pmatrix} 2 \\ 0.5 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \begin{pmatrix} 0.5 \\ 1 \\ 0.25 \end{pmatrix}, \quad \begin{pmatrix} \varphi_x \\ \varphi_y \\ \varphi_z \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{\pi}{3} \\ \frac{-\pi}{5} \end{pmatrix} \tag{7.4}$$

where ρ_x, ρ_y, ρ_z represent the amplitude, f_x, f_x, f_x the frequency and $\varphi_x, \varphi_y, \varphi_z$ the phase values of the camera position in three directions with respect to the world frame. In this example the parameters specified in equation (7.4) have been used. Figure 7.2 shows the camera trajectory together with the 3D points in space.

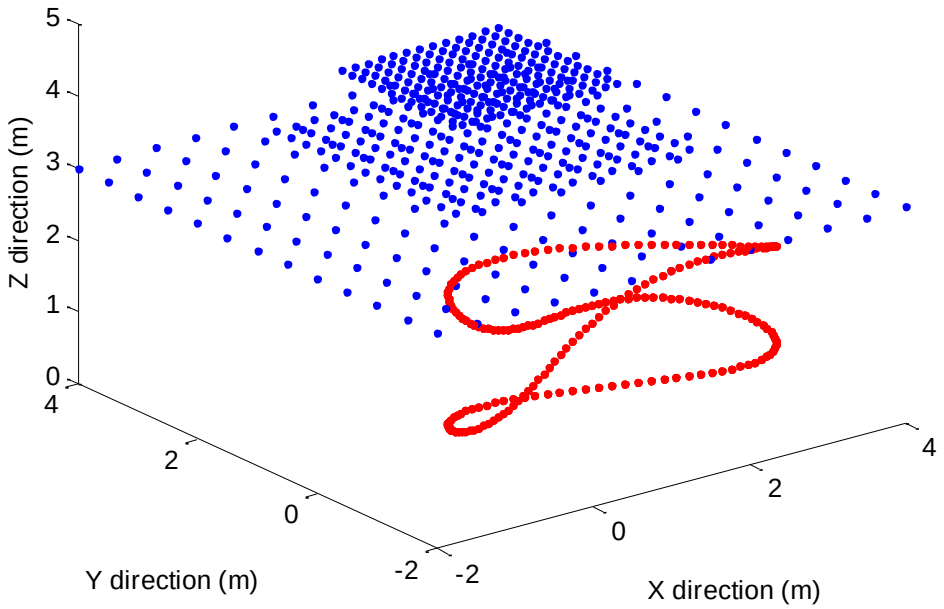


Figure 7.2 : 3D camera trajectory. Blue and Red dots represent 3D feature points and camera ground truth trajectory

7.1.3 3D Orientation

The camera orientation was simulated using quaternions. A quaternion defines the direction and angle of rotation. In this simulation the direction of rotation was defined using the cross product of two vectors. The first one was a unit vector in opposite direction to the gravity vector. The second one was the vector connecting the camera centre to a fixed point in space, which for the sake of simulation was taken to be at $(0\ \ 0\ \ 2)^T$:

(7.5) $$\mathbf{V_q} = \frac{\mathbf{V_1} \times \mathbf{V_2}}{\|\mathbf{V_1} \times \mathbf{V_2}\|}, \qquad \mathbf{V_1} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \end{pmatrix}, \quad \mathbf{V_2} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{2} \end{pmatrix} - \mathbf{X_c^w}$$

The angle of rotation was set to take a sinusoid form as in equation (7.6). Therefore the quaternion was defined as per equation (7.7).

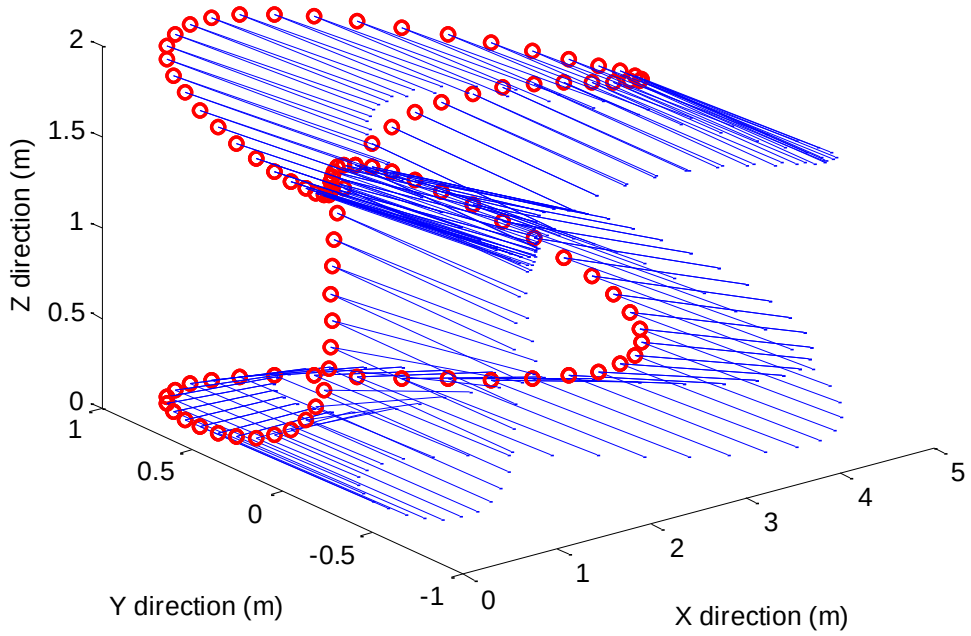


Figure 7.3 : Camera axis of rotation. 'red circles' and 'blue lines' represent the 'camera motion trajectory' and 'axis of rotation', respectively

Figure 7.3 shows the quaternion direction vector in blue lines for each point of the camera trajectory, based on the parameters as specified in equation (7.8). Figure 7.4 shows the camera orientation in three directions.

(7.6) $\theta_q = A_\theta \sin(2\pi f_\theta t)$

(7.7) $\boldsymbol{Q} = [\boldsymbol{cos}\left(\frac{\boldsymbol{\theta_q}}{\boldsymbol{2}}\right), \boldsymbol{sin}\left(\frac{\boldsymbol{\theta_q}}{\boldsymbol{2}}\right)\boldsymbol{V_q}]$

(7.8) $\boldsymbol{A_\theta = \pi/16}, \quad \boldsymbol{f_\theta = 0.5Hz}$

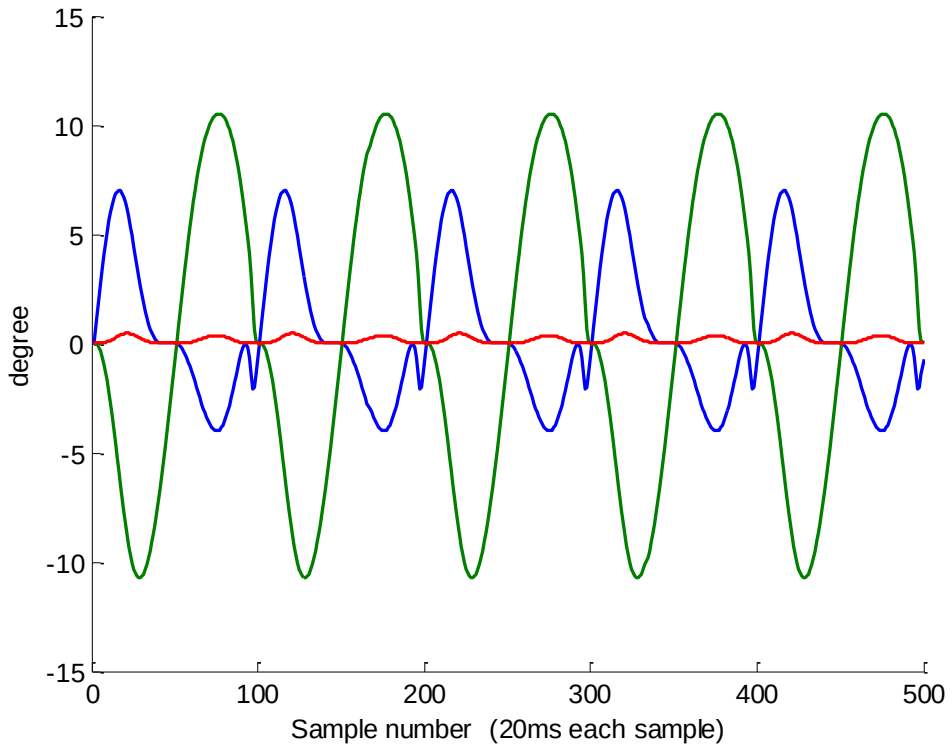


Figure 7.4 : 3D camera orientation, Roll, Pitch and Yaw, which have been illustrated by Blue, Green and Red

The rotation matrix $\mathbf{R_t^{wc}}$ can be easily derived from the quaternion (Mathworks, 2014). Assuming a quaternion is described as per equation (7.9), the rotation matrix can be determined using equation (7.10).

(7.9) $\boldsymbol{Q_t = q_0 + iq_1 + jq_2 + kq_3}$

(7.10) $$\boldsymbol{R_t^{wc}} = \begin{pmatrix} \boldsymbol{q_0^2 + q_1^2 - q_2^2 - q_3^2} & \boldsymbol{2(q_1 q_2 + q_0 q_3)} & \boldsymbol{2(q_1 q_3 - q_0 q_2)} \\ \boldsymbol{2(q_1 q_2 - q_0 q_3)} & \boldsymbol{q_0^2 - q_1^2 + q_2^2 - q_3^2} & \boldsymbol{2(q_2 q_3 + q_0 q_1)} \\ \boldsymbol{2(q_1 q_3 + q_0 q_2)} & \boldsymbol{2(q_2 q_3 - q_0 q_1)} & \boldsymbol{q_0^2 - q_1^2 - q_2^2 + q_3^2} \end{pmatrix}$$

7.1.4 2D Feature Points

Once the camera position and orientation data were produced, the 2D feature points, as seen by the camera at time t_n, were calculated. To do so, the coordinates of each 3D point in space was transformed using the translation vector ($T_{t_n}^{wc}$) and rotation matrix ($R_{t_n}^{wc}$) to find the coordinates in the camera frame. $T_{t_n}^{wc}$ is in fact $X_{t_n}^{w}$; the coordinates of the camera origin in world frame (per equation (7.3)). A pinhole camera perspective projection was then used to convert 3D points to 2D feature points. Finally a random noise was added to the coordinates of the 2D feature points to represent feature-tracking error.

(7.11) $$\boldsymbol{F_m^c} = \begin{pmatrix} \boldsymbol{X_{F,m}} \\ \boldsymbol{Y_{F,m}} \\ \boldsymbol{Z_{F,m}} \end{pmatrix} = \boldsymbol{R_{t_n}^{wc^{-1}}}\left(\boldsymbol{F_m^w - T_{t_n}^{wc}}\right),\ \boldsymbol{T_{t_n}^{wc} = X_{t_n}^{w}},\quad \boldsymbol{m = 1{:}\,N_f}$$

(7.12) $$\boldsymbol{p_{t_n,m}} = \boldsymbol{f}\begin{pmatrix} \frac{\boldsymbol{X_{F,m}}}{\boldsymbol{Z_{F,m}}} \\ \frac{\boldsymbol{Y_{F,m}}}{\boldsymbol{Z_{F,m}}} \end{pmatrix} + \begin{pmatrix} \boldsymbol{e_{n,x}} \\ \boldsymbol{e_{n,y}} \end{pmatrix}$$

$\mathrm{F_m^w}$ and $\mathrm{F_m^c}$ are the 3D coordinates of a feature point in world and current frames, respectively. Figure 7.5 shows the 2D feature points in the first image in the un-calibrated camera view. In order to simplify the simulation, hereafter, 2D feature points are considered to be in calibrated camera view. However for simulation using an un-calibrated camera view, the intrinsic matrix (K) can be used to provide simulated feature points. The tracking algorithm should in turn use (K^{-1}) to convert back to calibrated view.

7.1.5 Accelerometer Data

By knowing X_t^w in closed form, as explained earlier, the true velocity and acceleration can be calculated as described in equations (7.13) and (7.14):

$$(7.13)\ \boldsymbol{V_t^w = \dot{X}_t^w = \begin{pmatrix} 2\pi\rho_x f_x sin(2\pi f_x t + \varphi_x) \\ 2\pi\rho_y f_y sin(2\pi f_y t + \varphi_y) \\ 2\pi\rho_z f_z sin(2\pi f_z t + \varphi_z) \end{pmatrix}}$$

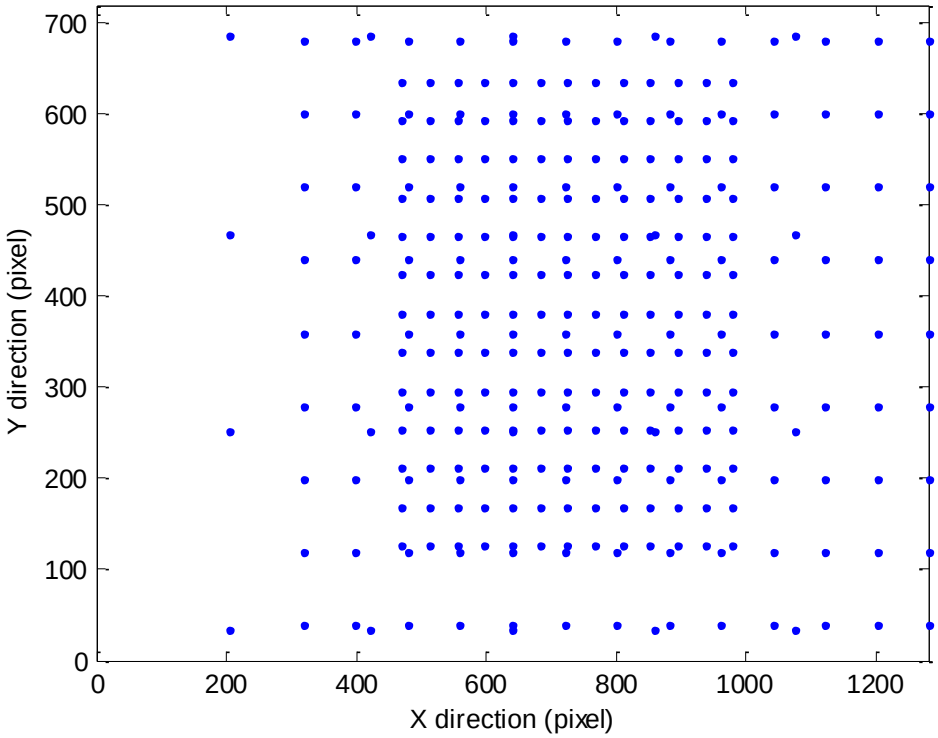


Figure 7.5 : 2D feature points on the first image

$$(7.14)\ \boldsymbol{A_t^w = \dot{V}_t^w = \begin{pmatrix} 4\pi^2\rho_x f_x^2 cos(2\pi f_x t + \varphi_x) \\ 4\pi^2\rho_y f_y^2 cos(2\pi f_y t + \varphi_y) \\ 4\pi^2\rho_z f_z^2 cos(2\pi f_z t + \varphi_z) \end{pmatrix}}$$

A_t^w represents the acceleration vector in the world frame, whereas the IMU measures the acceleration in its own frame of reference. Besides, A_t^w above does not include the effect of gravity. To take these two factors into account, gravity vector $G = (0\ 0\ g)$ was added to A_t^w and the result was transformed to

the IMU frame using the rotation matrix in order to simulate the ground truth acceleration as measured by the accelerometer:

(7.15) $\boldsymbol{A_t^c = R_t^{cw}(A_t^w + G)}$

An accelerometer has certain characteristics that affect the measurement of the acceleration. The main parameters defining an accelerometer's performance are ADC resolution, sensitivity, noise level, gain error, offset, drift, cross-axis sensitivity and temperature. The accelerometer measurement range can be initially set to give the most accurate value within the specified range. A typical accelerometer such as InvenSense MPU6500 can be set up to measure accelerations within $\pm 2g$, $\pm 4g$, $\pm 8g$ and $\pm 16g$. The main parameters of this accelerometer for the $\pm 2g$ range have been detailed in Table 7-1 (InvenSense, 2014).

Parameter	Value	Description/Comments
Sensitivity	16384 LSB/g	16-bit ADC, approx. $61 \mu g/LSB$
Non-linearity	$\pm 0.5\%$	In all three axes
Zero G offset tolerance	$\pm 60mg$	In all three axes
Zero G offset temperature drift	$\pm 1mg/°C$	In all three axes
Noise	$300\mu g/\sqrt{Hz}$	For a bandwidth of 100Hz, the total noise level is around $3mg$

Table 7-1 : Accelerometer Parameters

In order to take these parameters into account, noise, offset, gain and quantisation error have been introduced into the ground truth acceleration described in equation (7.14). Therefore the measured acceleration takes the following form:

(7.16) $\boldsymbol{\widetilde{A}_{t_n,d}^c = round(\eta[(1 + e_{g,d})A_{t_n,d}^c + e_{n,d} + e_{os}])/\eta, \quad d = x, y, z}$

$A_{t_n,d}^c$ and $\tilde{A}_{t_n,d}^c$ represent the ground truth acceleration and measured value in every direction. $e_{g,d}$ and e_{os} are the gain and offset error values. $e_{n,d}$ is a

random value representing noise. Noise was sampled from a normal distribution function with mean value of zero. The standard deviation depends on the accelerometer characteristics. η is the coefficient converting the digital ADC output to acceleration. $Round(x)$ is a function returning the integer value closest to x. η and $Round(x)$ are used to take into account the quantisation error.

In applications where a high degree of accuracy is required, calibration at the factory level is not adequate. In order to minimise the offset and gain error the accelerometer must be further calibrated before being used in the system. The calibration method has been described in detail in STMicroelectronics AN3182 application note (ST, 2010)

Suppose $(\tilde{A}^c_{t,x}, \tilde{A}^c_{t,y}, \tilde{A}^c_{t,z})^T$ represents the 3D acceleration vector measured by the accelerometer. Due to the offset, gain and axis misalignment errors the sensor output must be corrected using the following equation.

(7.17) $$\widetilde{\boldsymbol{A}}^{\boldsymbol{c}}_{\boldsymbol{t}} = \boldsymbol{M}\begin{pmatrix} \frac{\mathbf{1}}{\boldsymbol{S_x}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\mathbf{1}}{\boldsymbol{S_y}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\mathbf{1}}{\boldsymbol{S_z}} \end{pmatrix}\begin{pmatrix} \widetilde{\boldsymbol{A}}^{\boldsymbol{c}}_{\boldsymbol{t,x}} - \boldsymbol{e_{os,x}} \\ \widetilde{\boldsymbol{A}}^{\boldsymbol{c}}_{\boldsymbol{t,y}} - \boldsymbol{e_{os,y}} \\ \widetilde{\boldsymbol{A}}^{\boldsymbol{c}}_{\boldsymbol{t,z}} - \boldsymbol{e_{os,z}} \end{pmatrix}$$

M is the 3x3 misalignment matrix between the accelerometer sensing axes and the device body axes. S_x, S_y and S_z are the sensitivity factors. $e_{os,x}$, $e_{os,y}$ and $e_{os,z}$ are the offset error values in different directions. The simplified form of the above equation is:

(7.18) $$\widetilde{\boldsymbol{A}}^{\boldsymbol{c}}_{\boldsymbol{t}} = \begin{pmatrix} \boldsymbol{\alpha_{11}} & \boldsymbol{\alpha_{12}} & \boldsymbol{\alpha_{13}} \\ \boldsymbol{\alpha_{21}} & \boldsymbol{\alpha_{22}} & \boldsymbol{\alpha_{23}} \\ \boldsymbol{\alpha_{31}} & \boldsymbol{\alpha_{32}} & \boldsymbol{\alpha_{33}} \end{pmatrix}\begin{pmatrix} \widetilde{\boldsymbol{A}}^{\boldsymbol{c}}_{\boldsymbol{t,x}} \\ \widetilde{\boldsymbol{A}}^{\boldsymbol{c}}_{\boldsymbol{t,y}} \\ \widetilde{\boldsymbol{A}}^{\boldsymbol{c}}_{\boldsymbol{t,z}} \end{pmatrix} + \begin{pmatrix} \boldsymbol{\beta_1} \\ \boldsymbol{\beta_2} \\ \boldsymbol{\beta_3} \end{pmatrix}$$

The goal of accelerometer calibration is to determine α and β parameters using 6 known stationary orientations with known acceleration. In the example used in this chapter it was assumed that the accelerometer was calibrated before being used. The following values have been used for the parameters in equation (7.16).

$$(7.19)\quad \begin{pmatrix} \mathbf{e}_{\mathbf{g},d} \\ \mathbf{e}_{\mathbf{os},d} \\ \mathbf{e}_{\mathbf{n},d} \\ \boldsymbol{\eta} \end{pmatrix} = \begin{pmatrix} \mathbf{0.5\%} \\ \boldsymbol{6mg} \\ \boldsymbol{N(0,100mg)} \\ \frac{\mathbf{2^{15}}}{\mathbf{2}\boldsymbol{g}} \end{pmatrix} \qquad \boldsymbol{d = x, y, z}$$

Figure 7.6 and Figure **7.7** show the accelerometer output and error in all three directions, respectively.

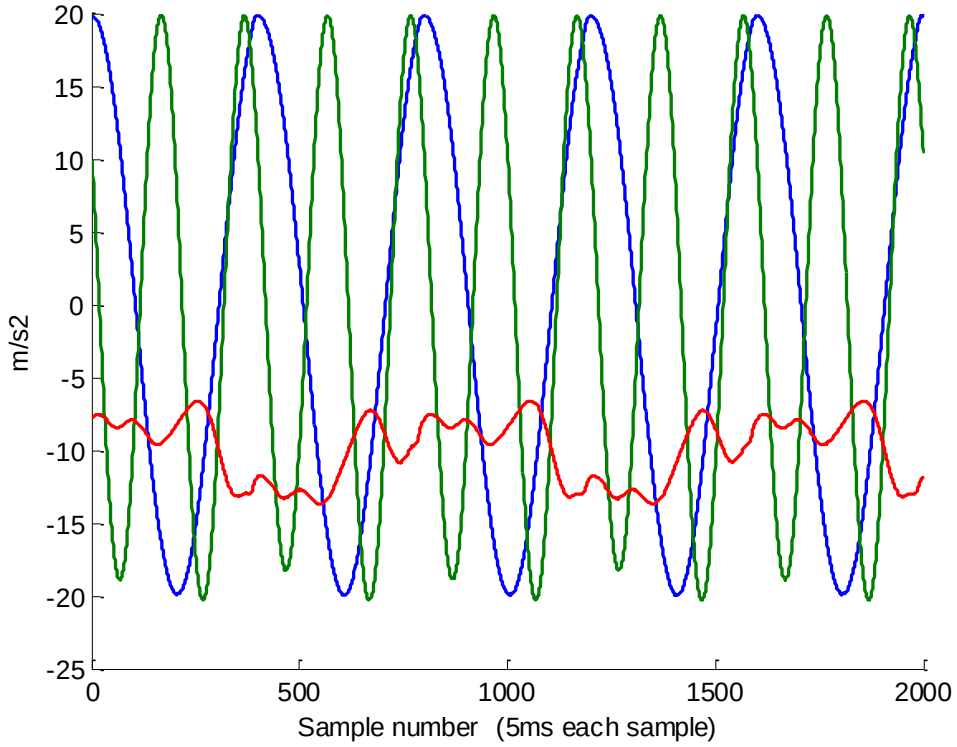


Figure 7.6 : Accelerometer output along x, y and z axes, shown by 'Blue', 'Green' and 'Red'

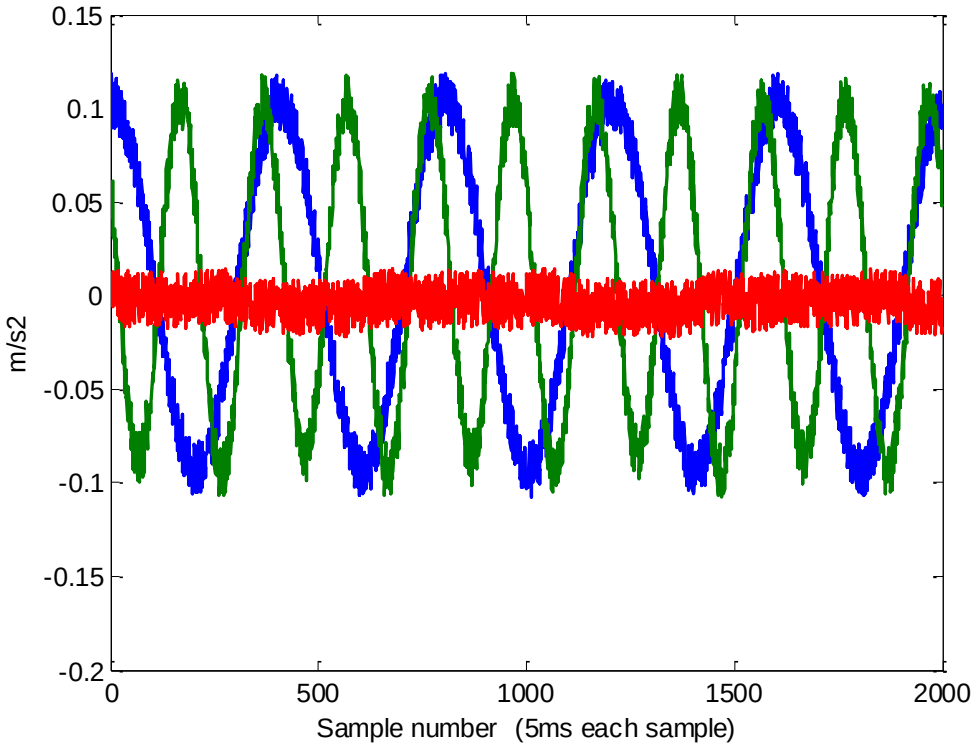


Figure 7.7 : Accelerometer error along x, y and z axes, illustrated by 'Blue', 'Green' and 'Red'

7.1.6 Gyroscope Data

The gyroscope data was generated by calculating the quaternion difference between two consecutive IMU samples at t_n and t_{n-1} with time difference δt.

(7.20) $\boldsymbol{\delta Q} = \boldsymbol{Q}^*_{\boldsymbol{t_{n-1}}} \boldsymbol{Q}_{\boldsymbol{t_n}} / \left\| \boldsymbol{Q}_{\boldsymbol{t_{n-1}}} \right\|$

In this equation $Q^*_{t_{n-1}}$ and $\left\| Q_{t_{n-1}} \right\|$ are the conjugate and norm of $Q_{t_{n-1}}$. δQ was then used to calculate the incremental rotation vector θ over δt. Angular velocity is expressed as in equation (7.21). Figure 7.8 shows the IMU angular velocity in three directions for the conditions stated in equations (7.5) to (7.8).

(7.21) $\begin{pmatrix} \boldsymbol{\omega_1} \\ \boldsymbol{\omega_2} \\ \boldsymbol{\omega_3} \end{pmatrix} = \frac{\mathbf{1}}{\boldsymbol{\delta t}} \begin{pmatrix} \boldsymbol{\theta_1} \\ \boldsymbol{\theta_2} \\ \boldsymbol{\theta_3} \end{pmatrix}$

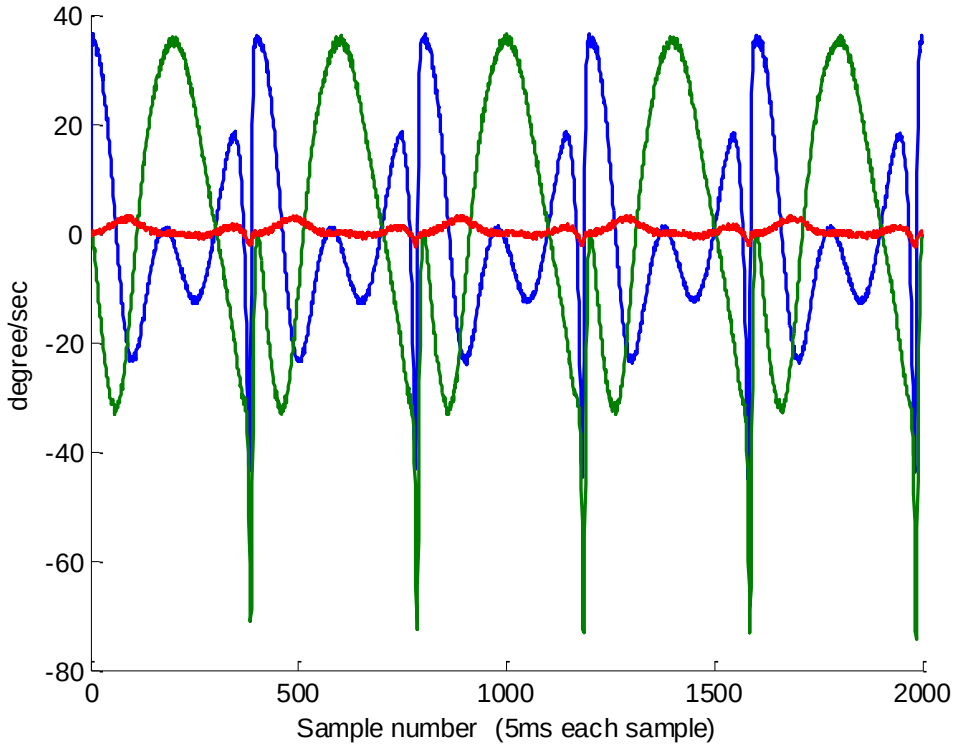


Figure 7.8 : Angular velocity shown by 'Blue', 'Green' and 'Red' around x, y and z axes

7.2 Tracking Results and Analysis

In this section the tracking output for the simulated system described in section 7.1 is presented and analysed. The system incorporated the parameters set out in equations (7.2) to (7.8). The image system sampling rate was set to 50Hz. Figure 7.9 shows the tracking trajectory, ground truth and 3D points. The tracked trajectory closely followed the ground truth, however in some regions, where the population of feature points was less dense, the tracked trajectory deviated from the ground truth.

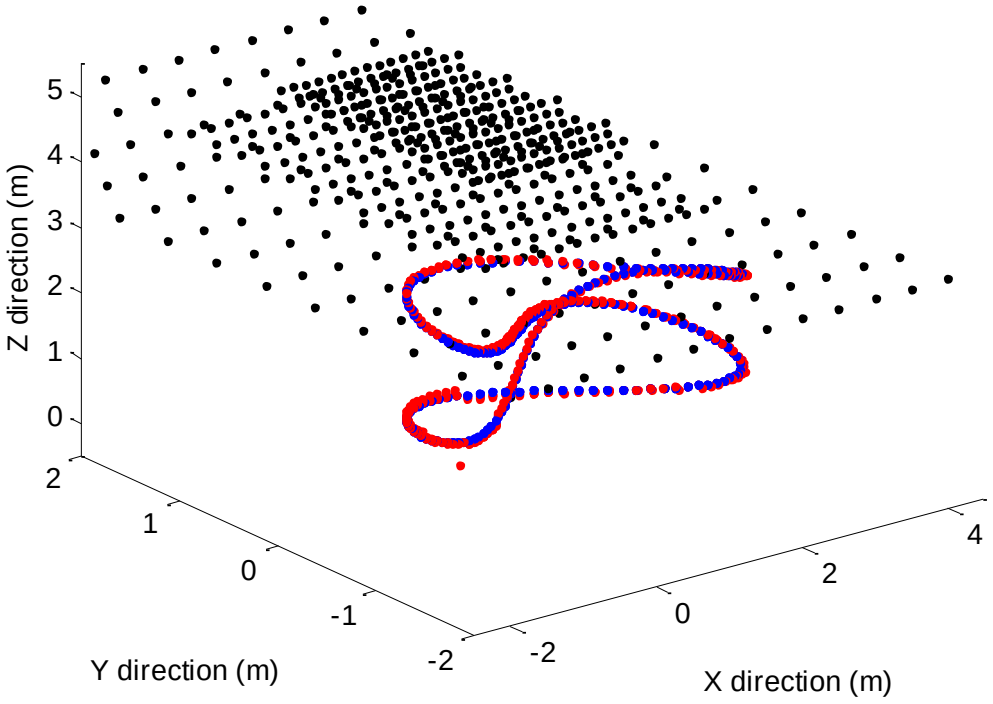


Figure 7.9 : Tracking trajectory. '3D points in space', 'Ground truth' and the 'Tracked trajectory' have been illustrated by 'Black', 'Blue' and 'Red' dots, respectively.

Figure 7.10 shows 3D camera position as the ground truth, hybrid system output and IMU-only tracking output. It can be seen that although the camera goes round the trajectory several times, the system returns back to low-error state and that there is no permanent tracking drift.

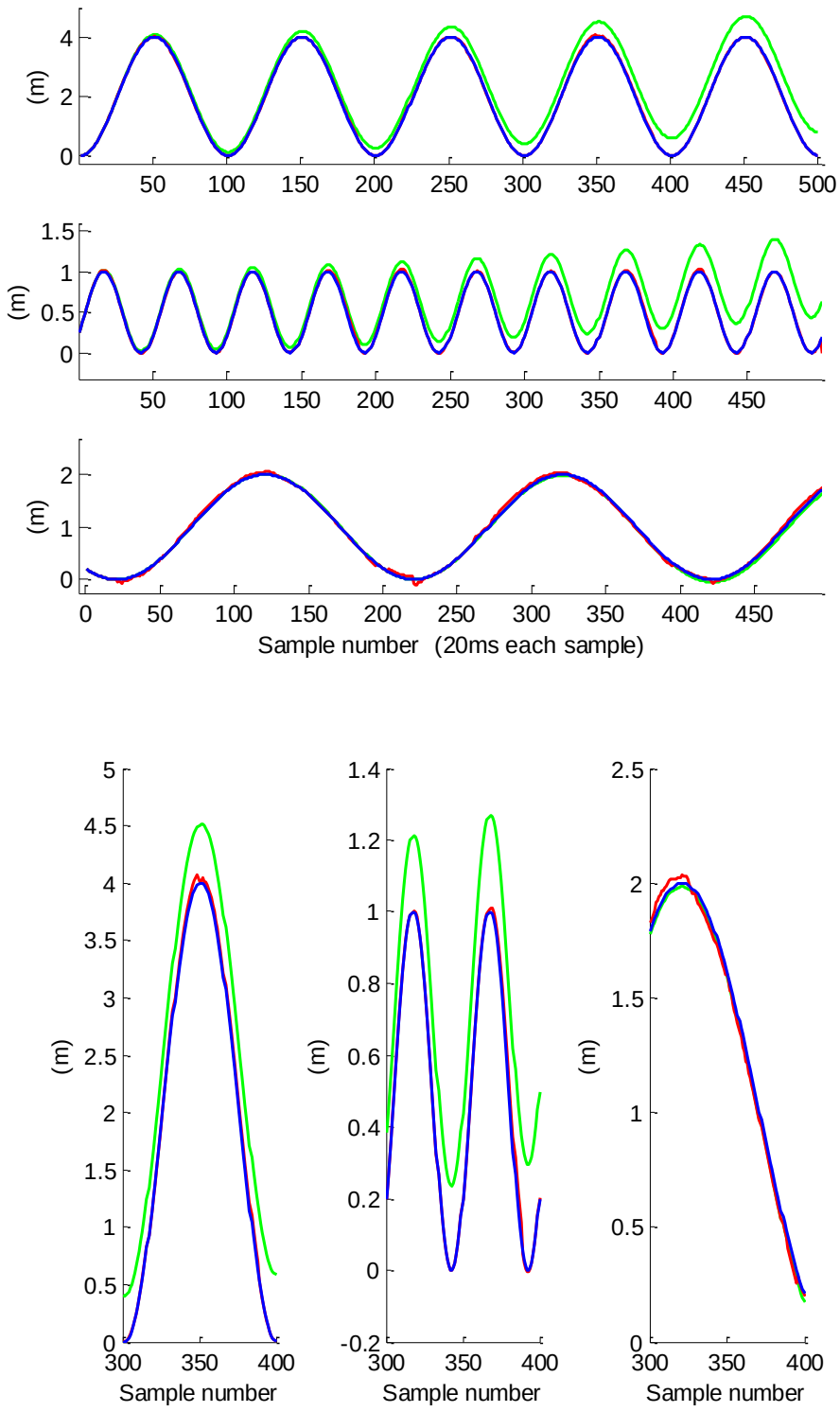


Figure 7.10 : Camera position along x,y and z axes shown in the first three images (top, middle, bottom) for the entire simulation, as well as the camera position shown in the bottom three images (left, middle, right) between samples 300 and 400. The ‘Blue’, ‘Red’ and ‘Green’ lines represent ‘Ground truth’, ‘Hybrid tracker’ and ‘IMU trajectories’.

7.2.1 Effect of FoE Quality on Tracking

Focus of expansion plays an important role in the proposed solution in particular the particle evaluation. During tracking there will be circumstances where a valid FoE does not exist, requiring the hybrid tracking to be bypassed and IMU-only take over. The hybrid system is capable of determining such cases. The red circles on the blue error line in Figure 7.11 indicate circumstances where the image-based system, due to the lack of a reliable FoE point, was not capable of providing additional information and therefore IMU-only tracking has taken over.

Figure 7.11 shows the tracking error over 500 images. The camera has travelled approximately 48m and the average tracking error in x, y and z directions are as per **Error! Reference source not found.**.

Mean error (cm) over total travelled distance of 48m				
3D pose	X axis	Y axis	Z axis	Error ratio
3.6177	1.605	0.7321	3.1584	0.07%

Table 7-2 : Tracking error

Figure 7.12 shows that the distance between FoE and the flow lines dramatically increased when the angle between camera speed vector and image plane approached zero. Looking again at Figure 7.11 these points correspond to where the IMU-based tracking took over and the hybrid system was bypassed.

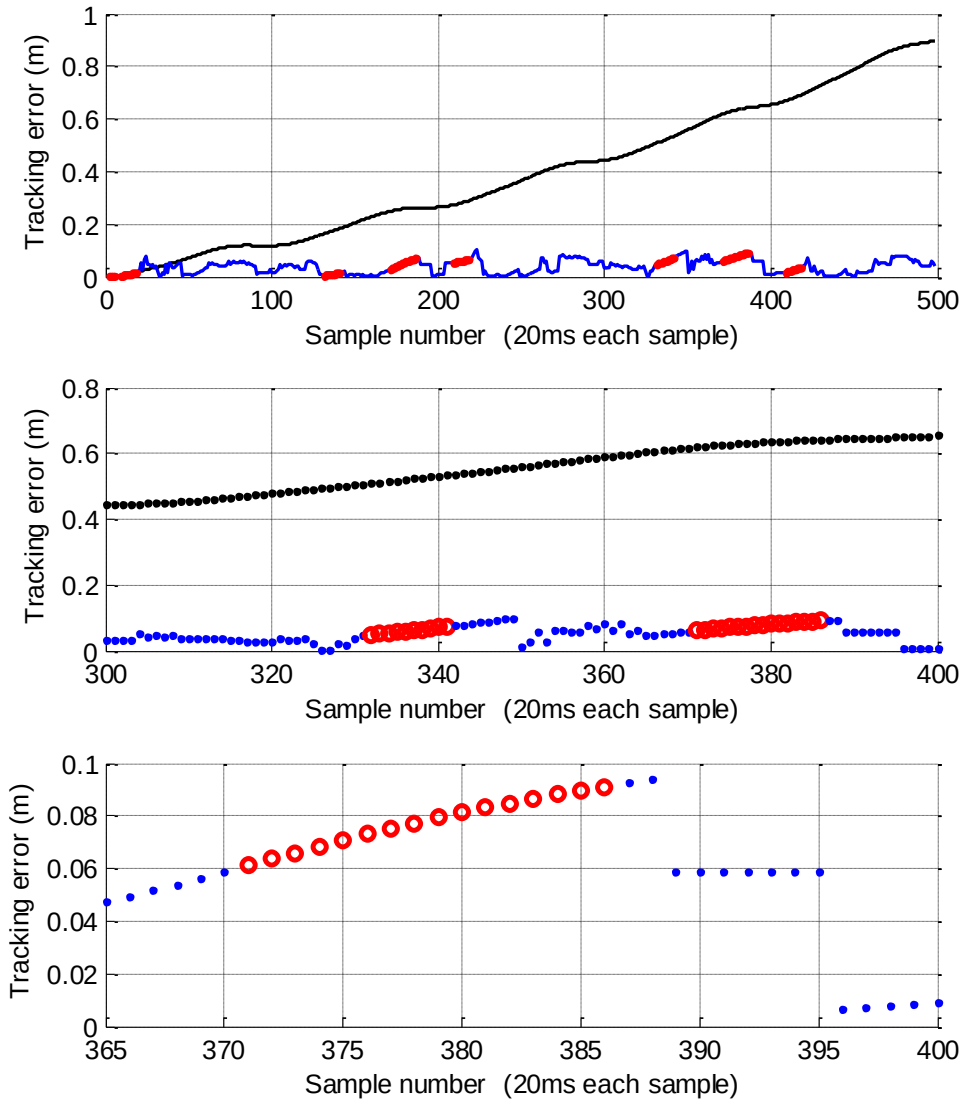


Figure 7.11 : Tracking error for IMU only system vs hybrid system. The top image shows the entire period of simulation, the middle one between samples 300 and 400, and the bottom one between 365 and 400, in order to provide more details. The 'Blue' and 'Black' dots illustrate the 'Hybrid' and 'IMU-only' tracking error. The 'Red' circles show where the hybrid tracking has been bypassed and IMU tracking taken over.

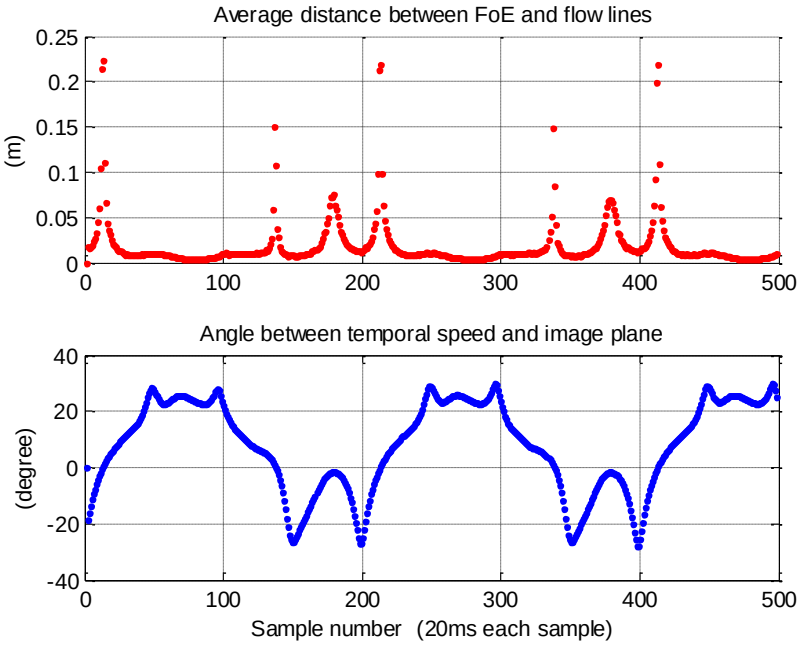


Figure 7.12 : FoE average distance to flow lines vs the angle between the speed vector and image plane

7.2.2 Effect of Image Sampling Time

In the first example, the image was sampled at 50fps. Here the image sampling rate was reduced to 25fps. Figure 7.13 shows the tracking error in this case.

The camera has travelled approximately 48m and the average tracking error in x, y and z directions was as per Table 7-63. The results indicate that the average tracking error compared to the 50fps case has increased considerably (see Figure 7.11 for comparison). This was due to the fact that the error associated with the IMU-based tracking between two images increased with the increased camera sampling interval, resulting in a bigger region for particle selection and consequently less accurate particles.

Mean error (cm) over total travelled distance of 48m				
3D pose	X axis	Y axis	Z axis	Error ratio
9.3781	4.8227	2.5221	7.6374	0.19%

Table 7-3 : Tracking error – Reduced sampling time

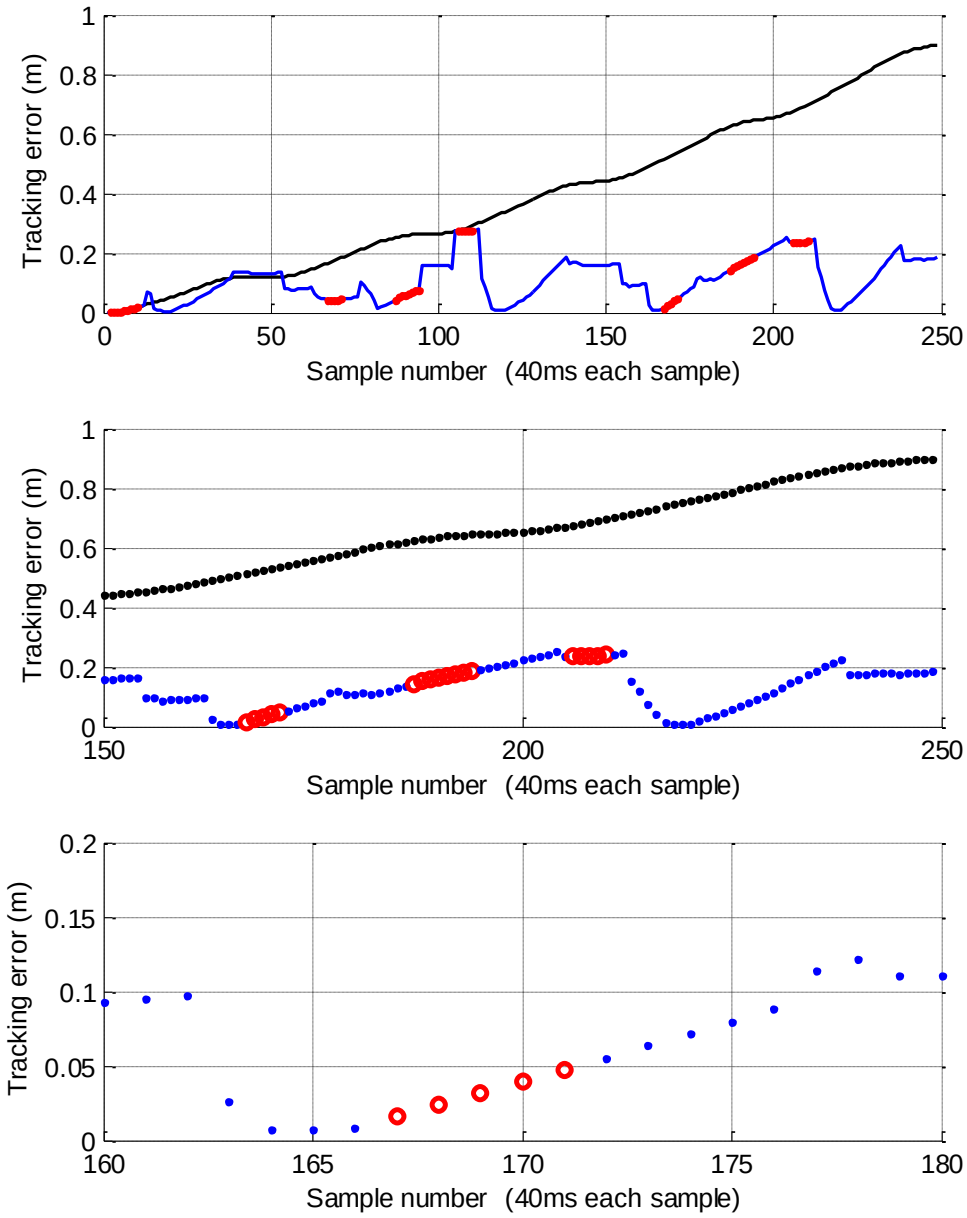


Figure 7.13 : Tracking error for 'IMU-only' system vs 'hybrid' system at 25fps. The top image shows the entire period of simulation, the middle one between samples 150 and 250, and the bottom one between 160 and 180, in order to provide more details. The 'blue' and 'black' dots illustrate the 'hybrid' and 'IMU-only' tracking error. The red circles show where the hybrid tracking has been bypassed and IMU tracking taken over.

7.2.3 Effect of Increased Acceleration

In order to simulate a different amount of acceleration, the frequency parameters in equation (7.3) were doubled (see (7.22)). The result was four times peak acceleration. Figure 7.14 shows the acceleration in 3 directions.

(7.22) $$\begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 0.5 \end{pmatrix}$$

Figure 7.15 shows the angular velocity in this case. Figure 7.16 shows the tracking error over 500 image samples (10sec). The average tracking error over nearly 97m travelled in 10sec was as per Table 7-4.

Mean error (cm) over total travelled distance of 97m				
3D pose	X axis	Y axis	Z axis	Error ratio
6.6375	2.9661	0.7831	5.8860	0.068%

Table 7-4 : Tracking error – Increased acceleration

The error in this case was higher than the error in the initial example of this chapter. The reason was the increased acceleration and consequently a higher proportional level of error in the acceleration data. This created a bigger search area for the particles, which adversely affected the accuracy of the selected particles. In addition, the FoE distance in Figure 7.17 shows that there were more instances of circumstances when a valid FoE did not exist, leading to the hybrid tracking being bypassed and taken over by the IMU-based tracking. This was another reason for experiencing higher value errors. Nevertheless, even though the error level was higher, the hybrid tracking result showed a stable performance, with no permanent drift taking place.

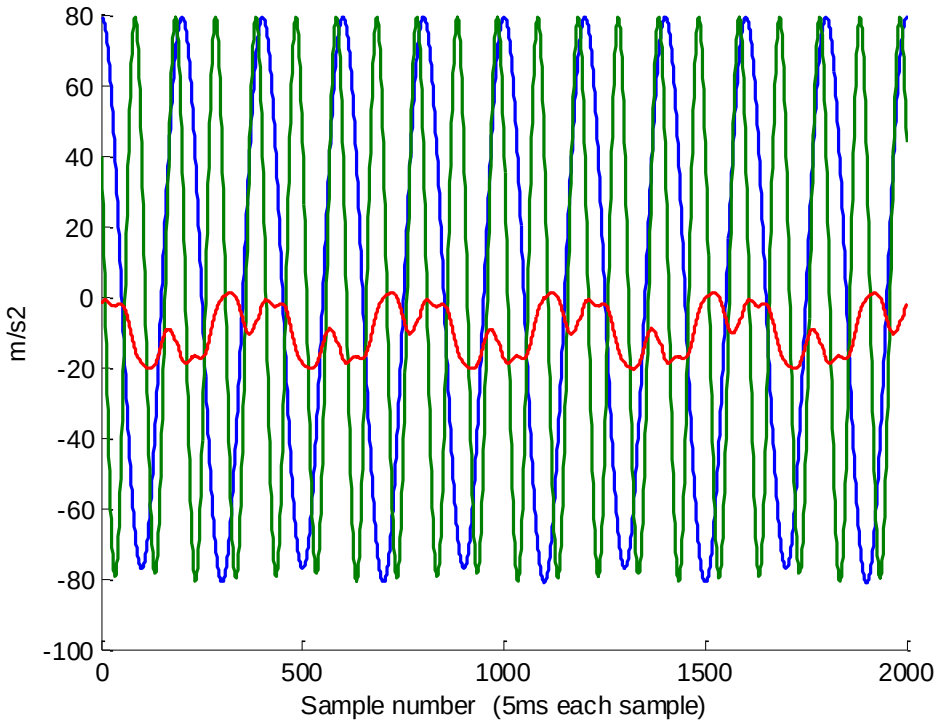


Figure 7.14 : Accelerometer output along x, y and z axes, shown by blue, green and red lines

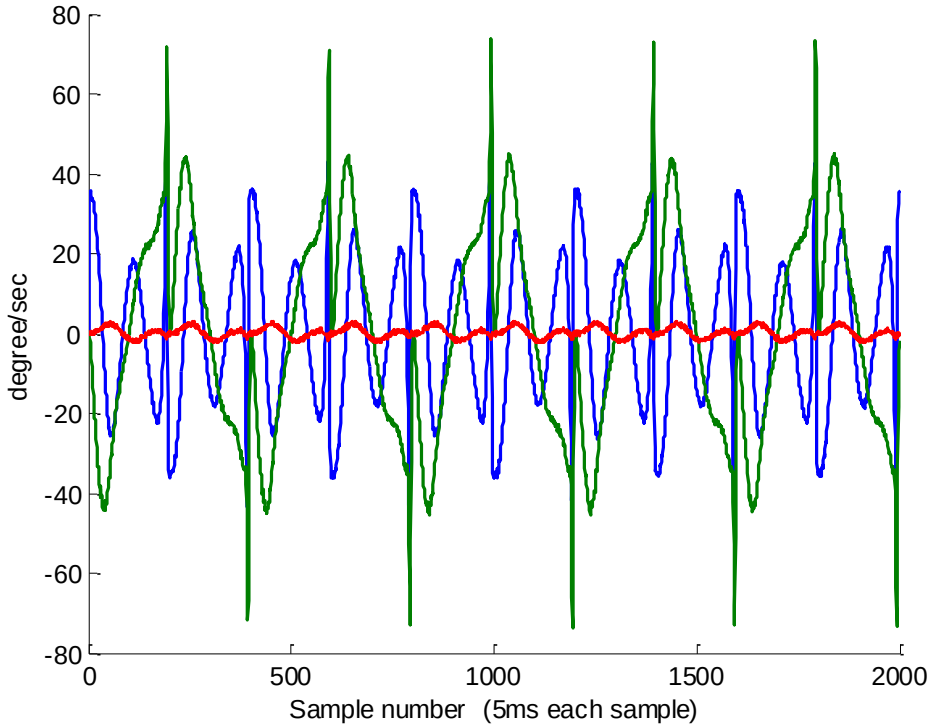


Figure 7.15 : Angular velocity. Blue, green and red lines represent angular velocity around x, y and z axes.

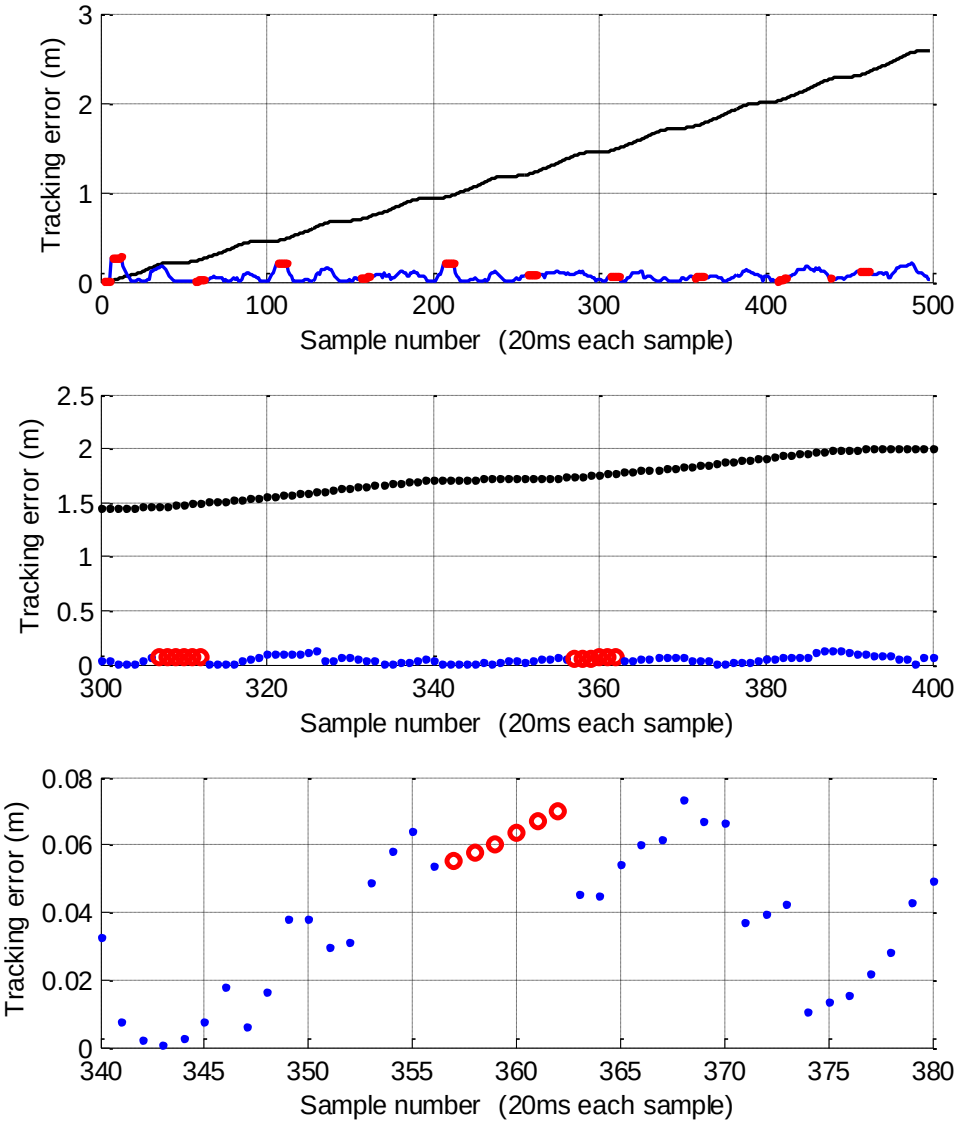


Figure 7.16 : Tracking error for 'IMU-only' system vs 'hybrid' system for the case of increased acceleration. The top image shows the entire period of simulation, the middle one between samples 300 and 400, and the bottom one between samples 340 and 380 for the purpose of clarification. The 'blue' and 'black' dots illustrate the 'hybrid' and 'IMU-only' tracking error. The red circles show where the hybrid tracking has been bypassed and IMU tracking taken over.

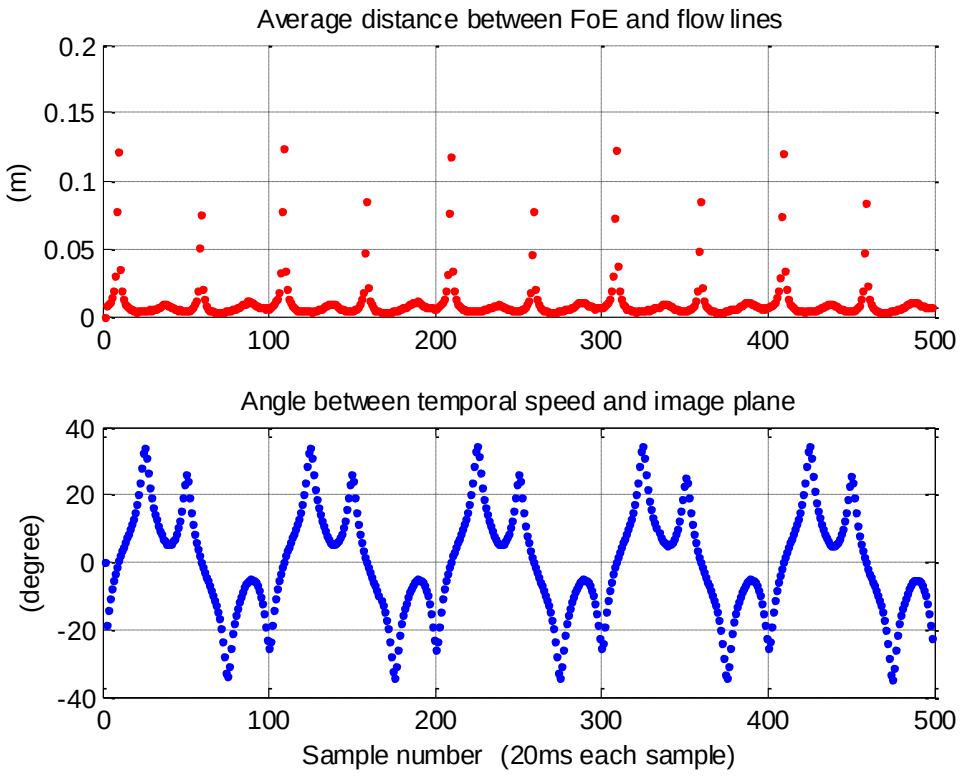


Figure 7.17 : FoE average distance to flow lines vs the angle between the speed and image plane

7.2.4 Effect of Increased Angular Velocity

In order to simulate angular velocity with a higher peak value and frequency content, new axis of rotation and angle were defined for the quaternions as follows:

(7.23) $\mathbf{V_q} = \frac{\mathbf{V_1} \times \mathbf{V_2}}{\|\mathbf{V_1} \times \mathbf{V_2}\|}, \qquad \mathbf{V_1} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \end{pmatrix}, \quad \mathbf{V_2} = \begin{pmatrix} \mathbf{0} \\ \mathbf{1} \\ \mathbf{2} \end{pmatrix} - \mathbf{X_c^w}, \quad \boldsymbol{A_\theta = \pi/8}, \quad \boldsymbol{f_\theta = 2Hz}$

Figure 7.18 shows the axes of rotation. Figure 7.19 shows the angular velocity for this case, which had a higher peak and changed at a significantly higher pace compared to the initial example given in this chapter. Figure 7.20 shows the tracking error over the travelled distance. The average error was as per Table 7-5.

Mean error (cm) over total travelled distance of 48m			
3D pose	X axis	Y axis	Z axis
5.0106	2.3052	0.5525	4.4145

Table 7-5 : Tracking error – Increased angular velocity

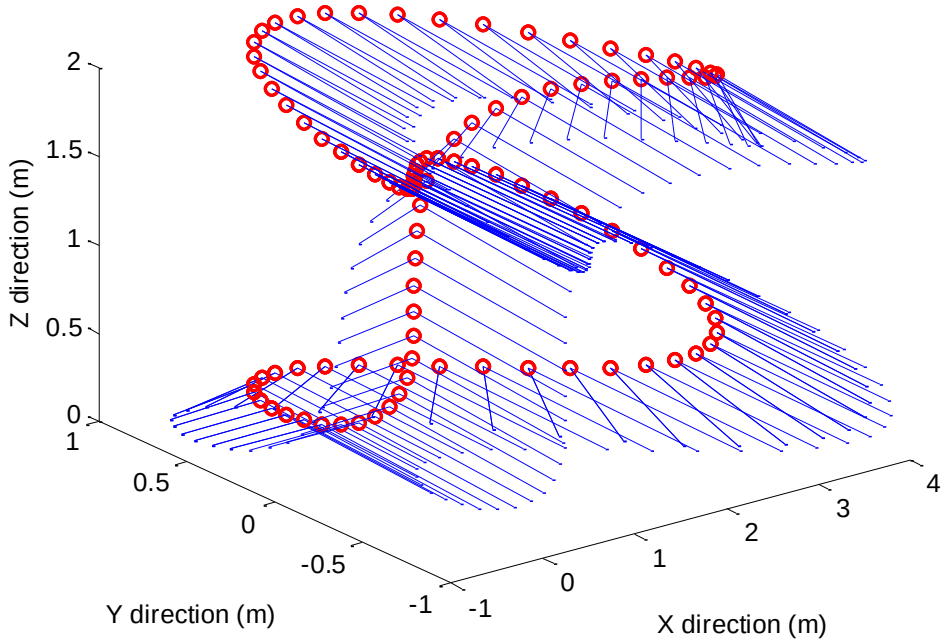


Figure 7.18 : Camera axis of rotation. Red circles and blue lines represent the camera motion trajectory and axis of rotation, respectively

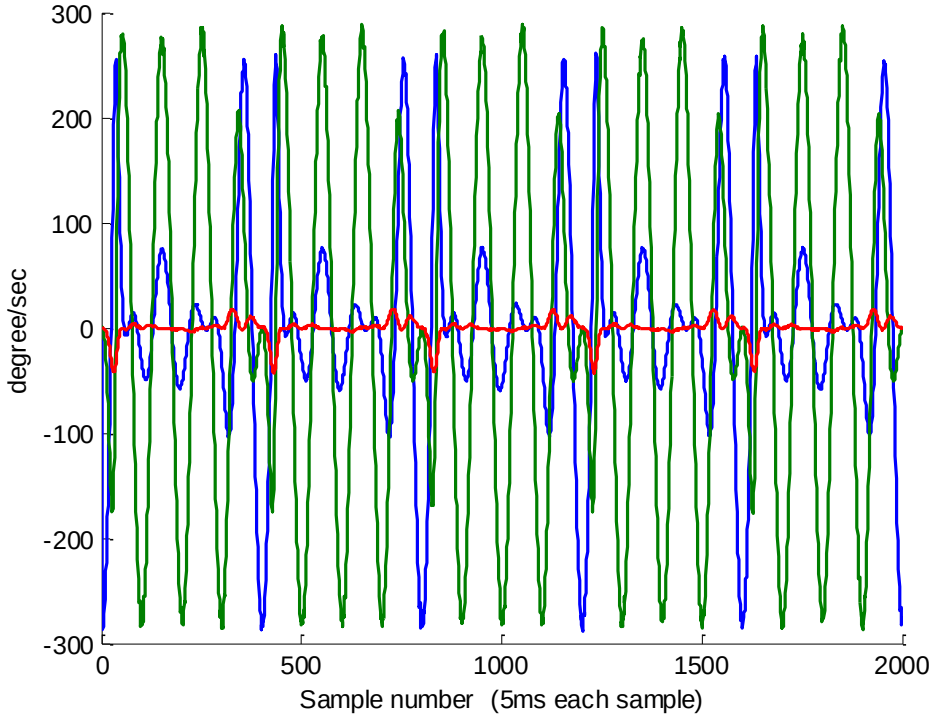


Figure 7.19 : Angular velocity shown by Blue, Green and Red around x, y and z axes

In this case the tracking error was higher than the initial example, due to the fact that invalid FoE points occurred more often. Also, as a result of the fast changing angular velocity, the rotation matrix changed rapidly as well, resulting in a higher acceleration measured by the IMU and consequently a higher accelerometer error. This led to less accurate particle generation, which in turn increased the error value.

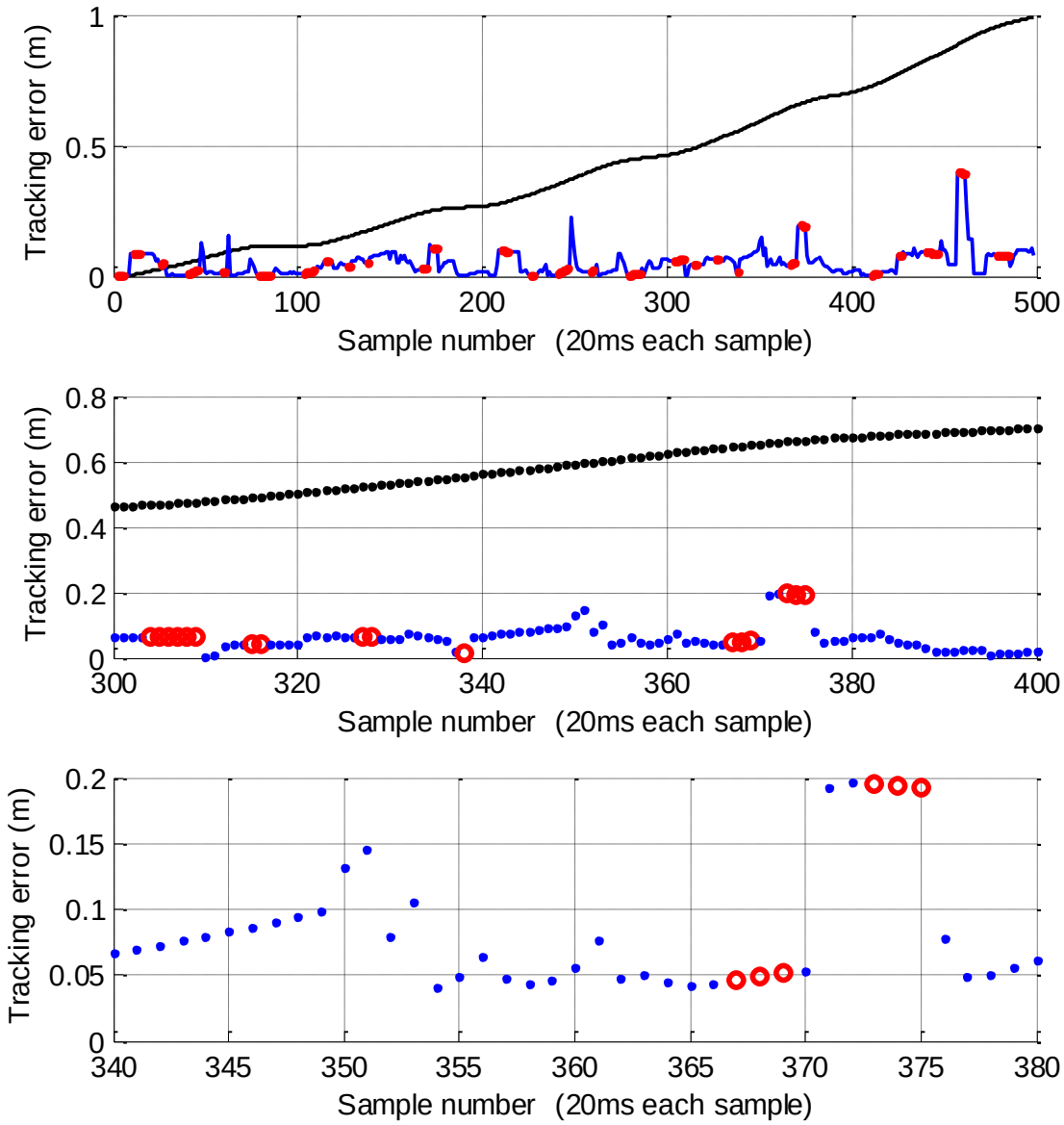


Figure 7.20 : Tracking error for IMU only system vs hybrid system for the case of increased acceleration. The top image shows the entire period of simulation and the middle one between samples 300 and 400 and the bottom image between samples 340 and 380, for the purpose of clarification. The blue and black dots illustrate the hybrid and IMU-only tracking error. The red asterisks show where the hybrid tracking has been bypassed and IMU tracking taken over.

7.3 Tracking Using Real Data

Camera tracking has been the subject of numerous studies in the last few years. One of the most well-known works carried out in this regard is the sFly project funded by European Union through the Seventh Framework Programme, FP7:2007-2013 (Zürich, 2014). The objective of the sFly project was to develop several small and safe helicopters, which could fly autonomously in city-like environments and be used to assist humans in tasks such as rescue and monitoring. The research team generated a number of datasets for evaluating their work. The main dataset produced during benchmarking (Lee, et al., 2010) is used in this section for evaluating the proposed hybrid tracking method.

Each helicopter, which is referred to as a Micro Aerial Vehicle (MAV), had been equipped with an IMU and two cameras. The MAV was a “Pelican” quad-rotor from ©Ascending Technologies. This has a built-in MEMS IMU equipped with 3 axis gyroscopes, 3 axis accelerometers and a magnetometer, which respectively give necessary measurements for the attitude rates, accelerations and the absolute heading. A filter within the IMU fuses the attitude rates, accelerations and heading at a rate of 1KHz to give absolute attitude measurements. The IMU provides acceleration, angular velocity and attitude data at 200Hz.

In addition to the IMU, the MAV also incorporates two cameras, one forward-looking and the other one downward-looking. The dataset selected for this evaluation only provides images seen by the downward looking camera. This camera has a fisheye lens with 150° field of view, running at a maximum frame rate of 30fps. Figure 7.21 shows the MAV with the integrated IMU and camera.

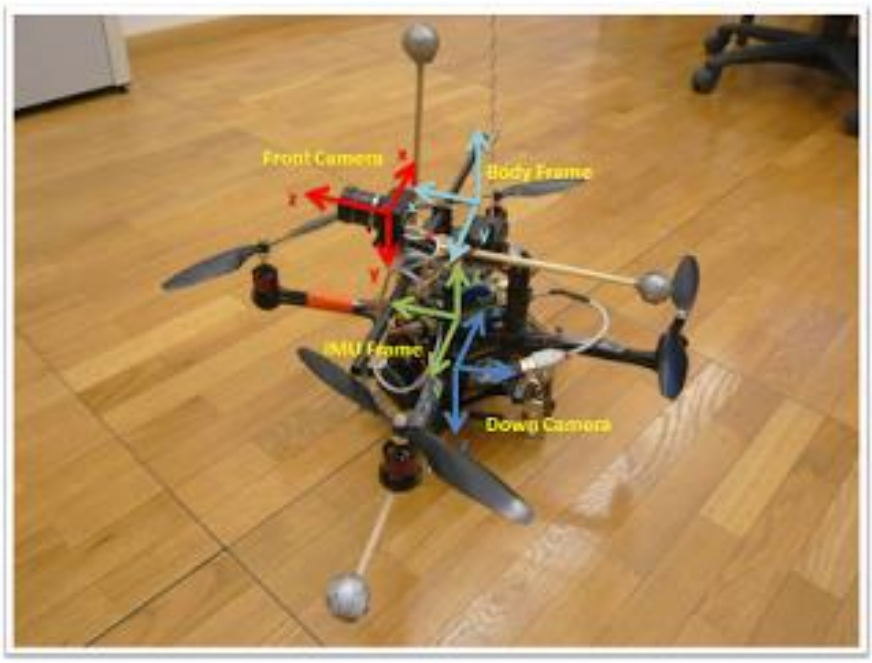


Figure 7.21 : Quad-rotor MAV with integrated IMU and camera

7.3.1 System Setup

In order to provide ground truth data, a Vicon tracking system with 8 cameras was setup by the sFly project team to track the MAV with three reflective markers in a 10m×10m×10m indoor environment. The three markers were the silver balls visible in the above picture. Pose estimation by the Vicon system was performed using a separate computer at a precise frequency of 200Hz. The Vicon system provides data concerning pose of the body frame with reference to the world frame. Figure 7.22 shows the world reference frame, also called the 'inertial frame'.

The IMU, camera and MAV body all have different reference frames (see Figure 7.21 for detail). In order to run the hybrid tracking algorithm, the IMU and Vicon data needed to be transformed to the camera reference frame. The IMU, camera and body frames are fixed to the MAV, therefore have fixed transformation matrices. The matrices transforming camera and IMU frames to

body frame are referred to as G^{bc} and G^{bi}, respectively. The dataset specifies these two matrices as follows:

(7.24) $$\boldsymbol{G^{bc}} = \begin{pmatrix} \mathbf{-0.0293} & \mathbf{-0.9989} & \mathbf{-0.0356} & \mathbf{0.0364} \\ \mathbf{-0.9996} & \mathbf{0.0294} & \mathbf{-0.0022} & \mathbf{-0.0122} \\ \mathbf{0.0033} & \mathbf{0.0356} & \mathbf{-0.9994} & \mathbf{-0.2354} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1.0000} \end{pmatrix}$$

(7.25) $$\boldsymbol{G^{bi}} = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{-0.04} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

These two principal transformation matrices can be used to derive the matrices for transforming body and IMU frames to camera frame, G^{cb} and G^{ci}, respectively:

(7.26) $$\boldsymbol{G^{cb}} = \boldsymbol{G^{cb^{-1}}}$$

(7.27) $$\boldsymbol{G^{ci}} = \boldsymbol{G^{cb}G^{bi}}$$

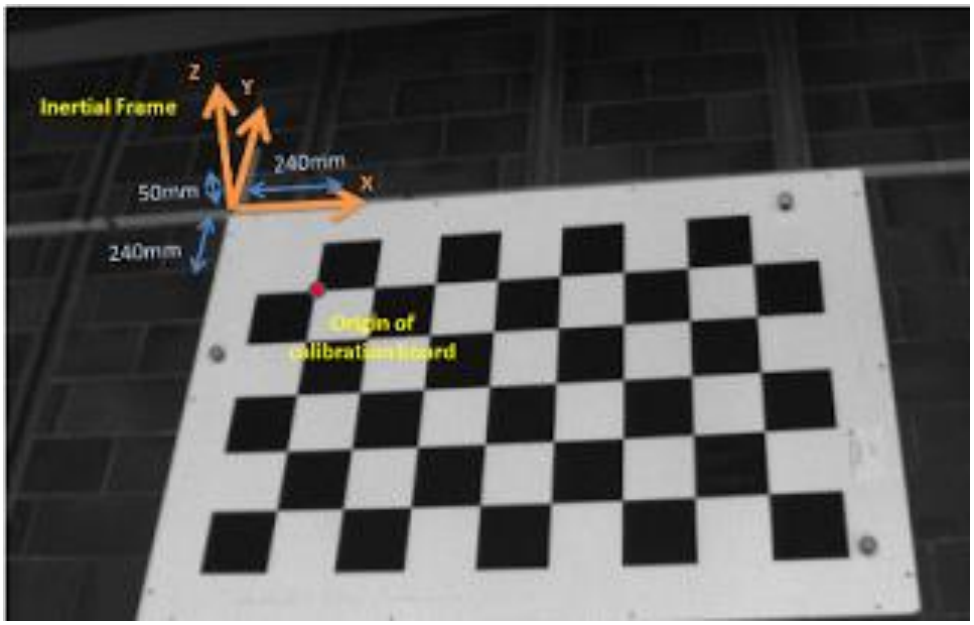


Figure 7.22 The world reference frame (inertial frame)

The MAV camera needs to detect and track feature points in the environment in order to contribute to the hybrid tracking. To do so, ARToolkit markers have been placed flat on the floor as shown in Figure 7.23. The hovering MAV captures images of these markers; some of them shown in Figure 7.24.

Figure 7.23 ARToolkit markers on the floor

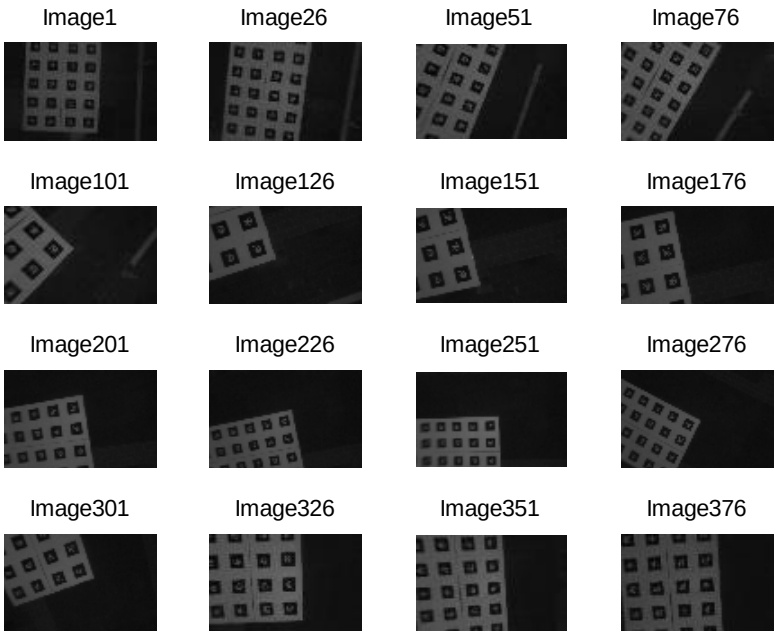


Figure 7.24 Some of the images captured by MAV

Tracking was carried out for over 400 images with the maximum image capture rate set to 30Hz, giving a period of approximately 33ms between two consecutive images. This was the maximum capture rate and in practice images were captured at an integer multiple of the basic sampling time of 33ms. In the chosen dataset nearly half of the images have been sampled at 33ms intervals, another half at 66ms and the remaining few at 99ms. The acceleration and angular velocity with respect to the IMU frame have been depicted in Figure 7.25 and 7.26.

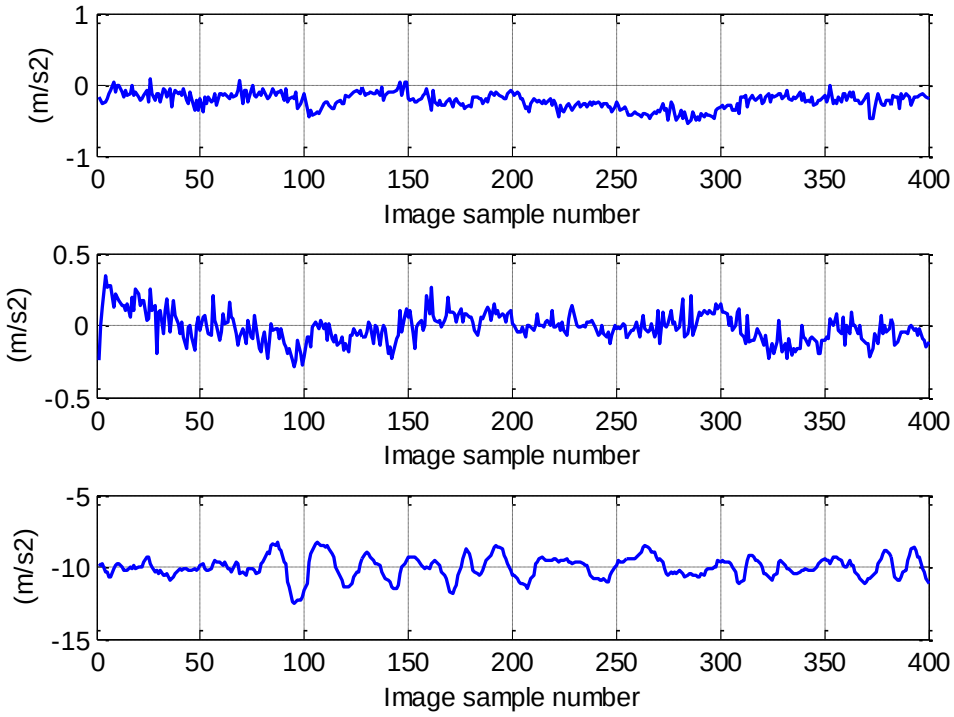


Figure 7.25 : Accelerometer data. The top, middle and bottom images illustrate acceleration along x, y and z axes, respectively

7.3.2 Tracking Results

The camera pose and orientation results using the proposed hybrid tracking system have been provided in Figure 7.27 and Figure 7.28. Figure 7.29 shows the tracked camera trajectory compared with the trajectory obtained from the

Vicon system. Table 7-6 shows the tracking RMS error over the travelled distance.

Mean error (cm) over total travelled distance of 11.2m			
3D pose	X axis	Y axis	Z axis
5.9842	4.9028	8.9836	1.6405

Table 7-6 : Tracking Error for SFLY data

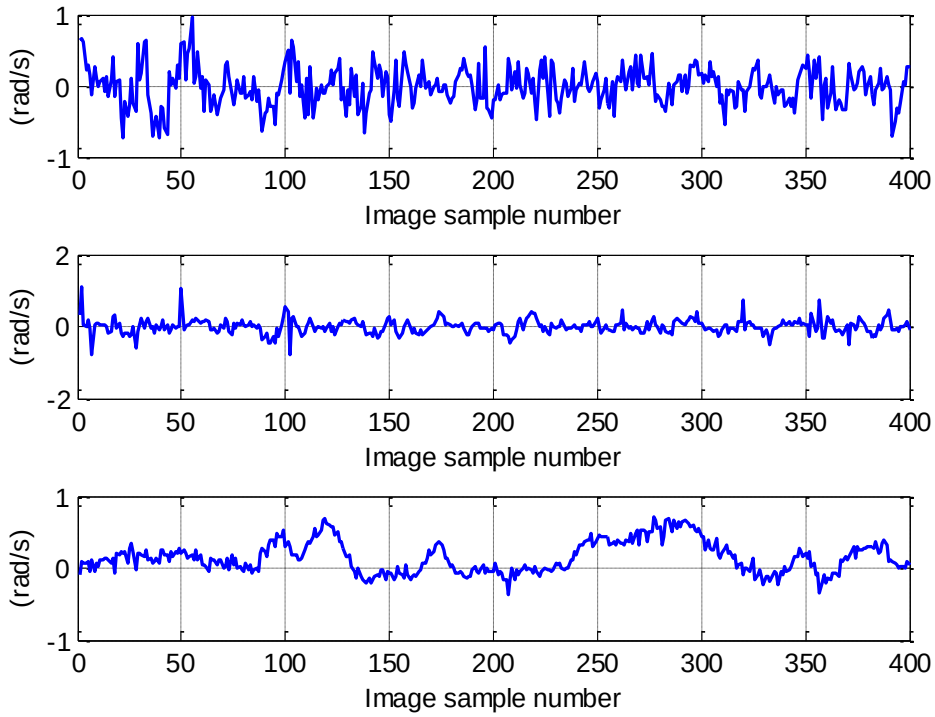


Figure 7.26 : Angular velocity. The top, middle and bottom images illustrate angular velocity round x, y and z axes, respectively

As can be seen in Figure 7.28, the hybrid tracking system closely tracked the ground truth, even when it moved away from ground truth in some instances, it moved back towards it and followed the path again.

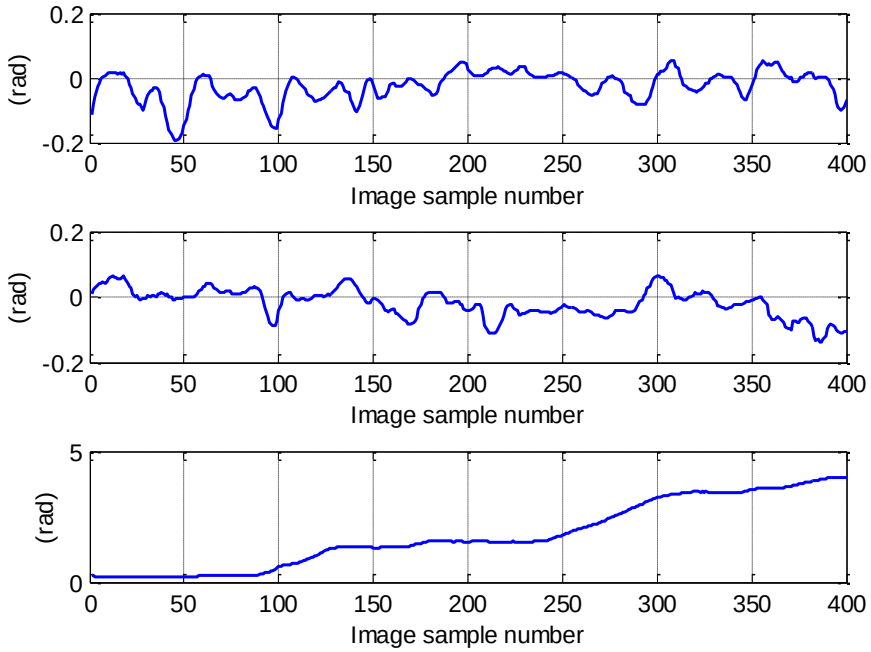


Figure 7.27 : Camera orientation. The top, middle and bottom images illustrate camera orientation, Roll, and Pitch and Yaw with respect to world frame

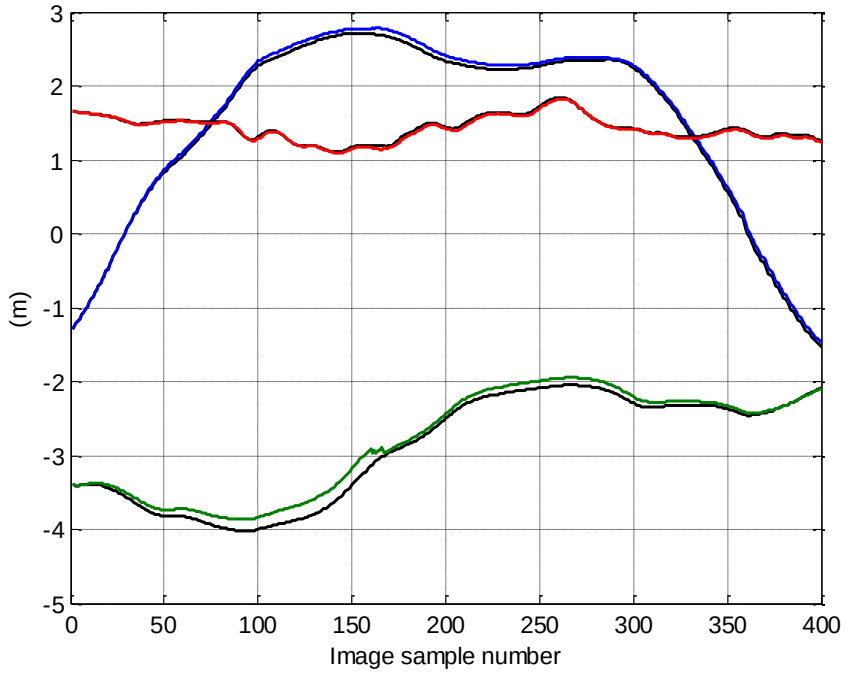


Figure 7.28 : Camera Pose. 'Blue', 'Green' and 'Red' graphs show camera pose along x,y, and z axes. The Black lines show the corresponding 'Vicon ground truth' data

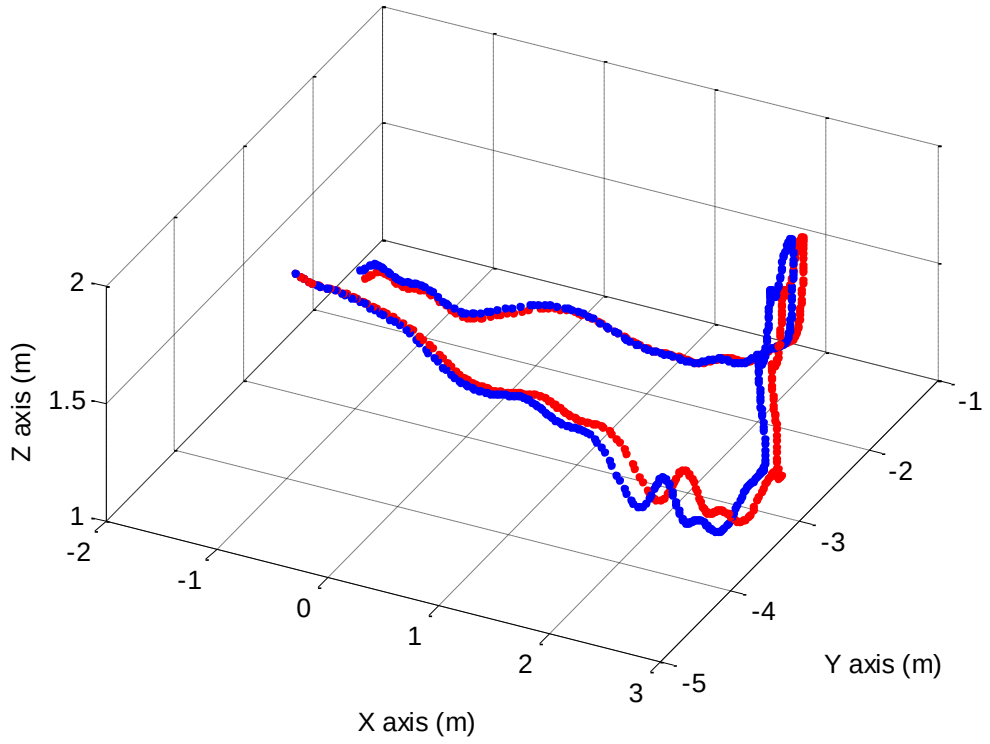


Figure 7.29 : Camera trajectory. The 'blue line' and 'red asterisks' are the 'Vicon ground truth' and 'hybrid tracking' trajectories.

7.3.3 Analysis

As discussed in Chapter 6, the proposed hybrid tracking method incorporates a decision making process which selects the most suitable tracking method at any particular time from IMU only, particle filtering-based with or without past state correction. The angle between the camera speed vector and the image plane (θ_{vn}) is the major factor in making this decision. Figure 7.30 shows a graph illustrating this angle, with black dots indicating instances where the hybrid tracking has been bypassed and IMU-only tracking has taken over. These are areas where the average distance between the FoE point and flow lines was significant either due to the small angle between the speed vector and image plane or due to the bad quality of one of the two consecutive images.

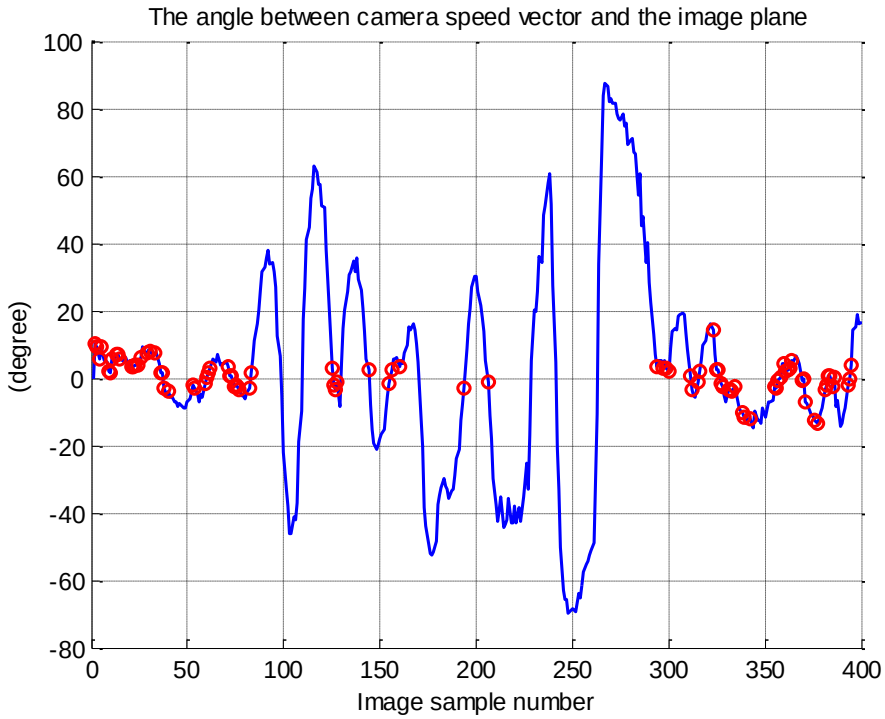


Figure 7.30 : The angle between camera speed vector and image plane. The red circles show where there has been an invalid FoE

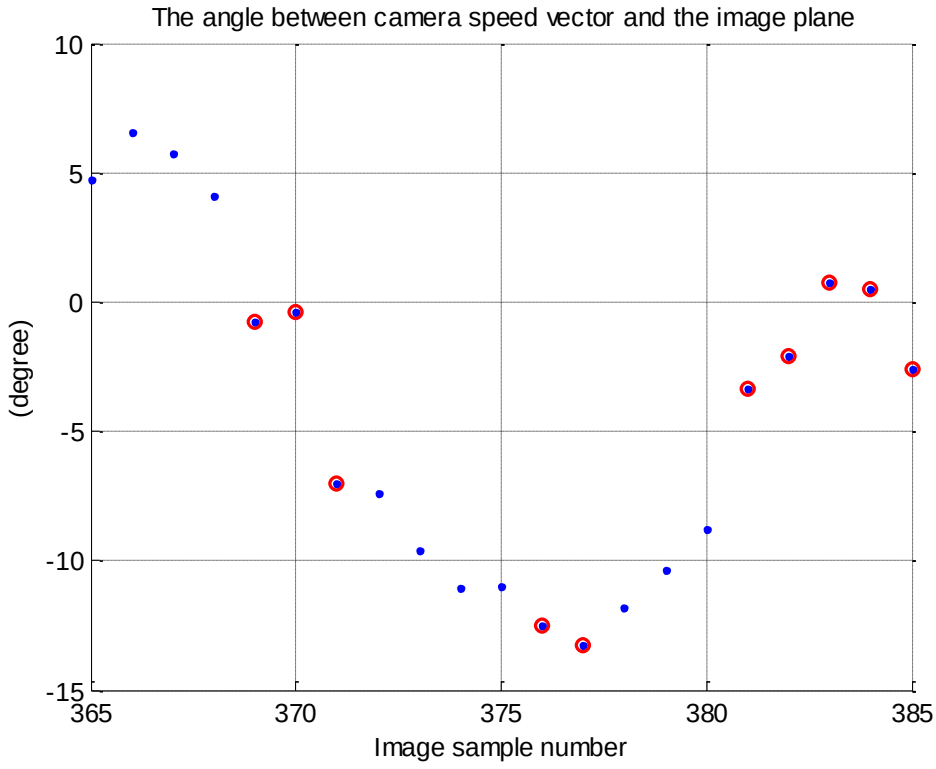


Figure 7.31 : The angle between camera speed and image plane (image samples 365 to 385). The red circles show there has been an invalid FoE

Figure 7.31 shows more clearly θ_{vn} angle from image 365 to 385. The circle symbols show the instances of IMU-only tracking. For example, image 370 had a small θ_{vn}, therefore a reliable FoE point could not be found and therefore IMU only tracking was applied.

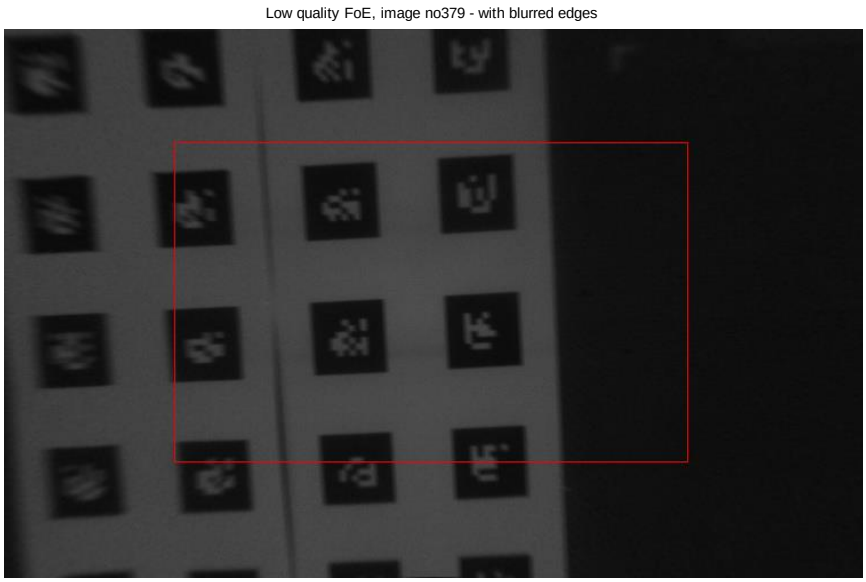


Figure 7.32 : Low quality FoE, image number 377 with blurred edges

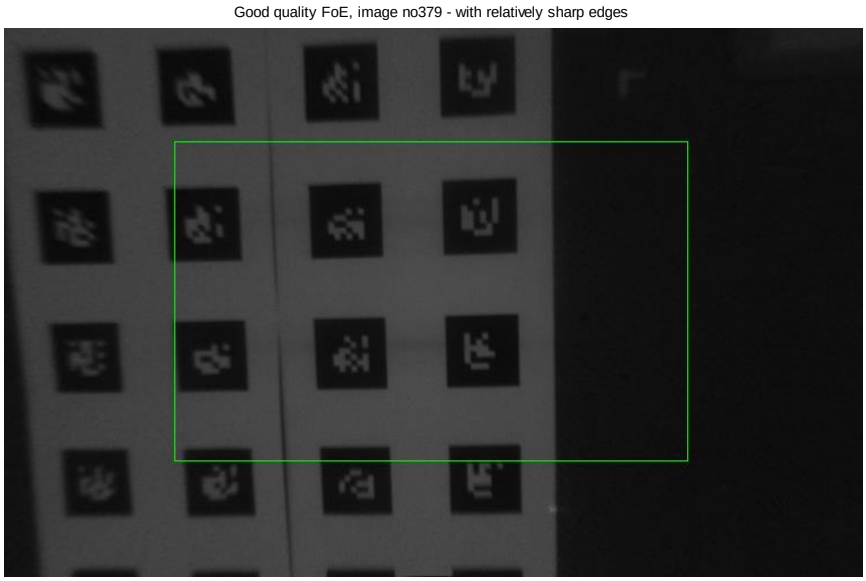


Figure 7.33 : Good quality FoE, image number 379 with relatively sharp edges

Figure 7.31 also shows that there were some instances, e.g. image 377, where despite having a large θ_{vn}, IMU only tracking was required. This was due to there being blurred edges in the image, which led to a less accurately tracked feature. Figures 7.32 and 7.33 show images with blurred and sharp edges, producing low and respectively good quality FoE points. Figure 7.33 shows that even though the image edges were not perfectly sharp, the algorithm still considered the quality of the produced FoE adequate and exhibited a good tracking performance (see Figure 7.28 between images 350 and 400).

In addition to the switching between IMU only and hybrid tracking, the system also occasionally applies past state correction. This has been shown in Figure 7.34. Black circles indicate where past state correction has applied. This figure shows that between images 160 to 170, past state correction has applied a number of times and on every occasion the error has reduced.

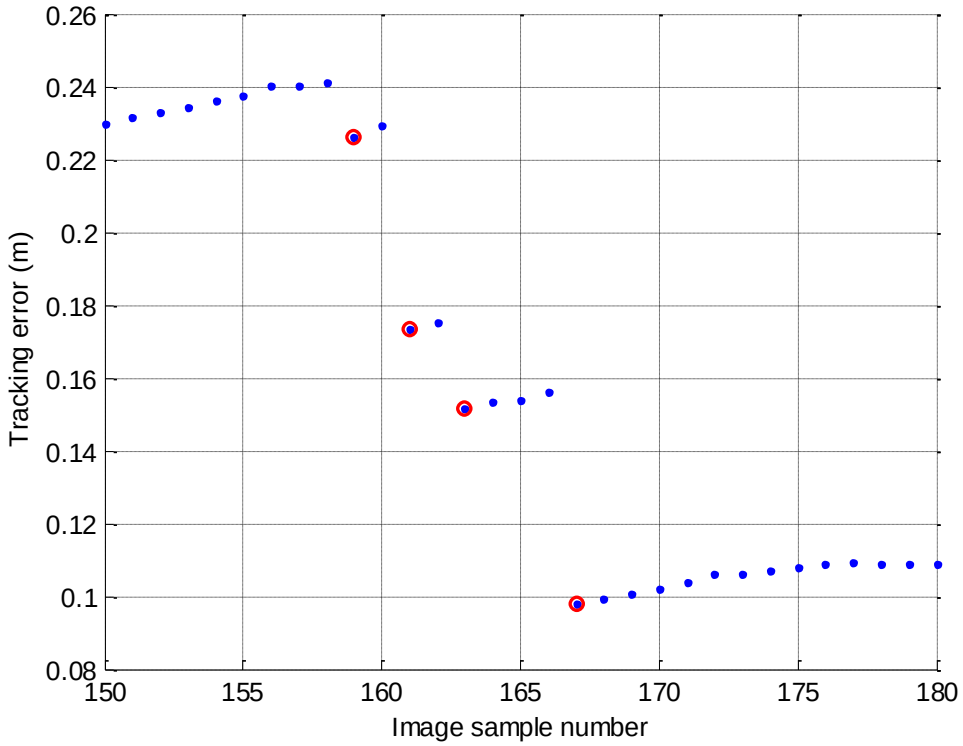


Figure 7.34 : Tracking error between samples 150 to 180. The red circles indicate where past state correction has applied.

7.4 Comparison with Alternative Solutions

Earlier in chapter 3, the Weiss VBN system (Weiss, 2012) was described as an advanced multi-sensor tracking technology. The system was developed as a part of the European sFly project and used for MAV trial flights. The Weiss VBN is based on fusing data from IMU and image-based tracking systems. Here the Weiss tracking system is compared with our system in the principles of operation and the output results.

The Weiss method treats the camera and IMU sensors independently and fuses the pose data from both sensors in an EKF framework. However like other vision-only systems, the estimated pose includes an un-known scaling factor. The independent image-based tracking is based on PTAM, which is computationally expensive, although faster than SLAM.

The EKF-based system relies on fusing two independent decisions. However our system has an integrated supervisory system, where the best candidates (particles) are identified and independently evaluated. A combination of the best candidates is chosen as the final outcome. This method is more likely to produce good results due to the fact that a wide area is searched for the correct result and more than two solutions are used for making the final decision.

The Weiss system implements a deterministic decision making process, whereas our system takes a stochastic approach. The downside of this method is the probability of particles being generated outside the region of optimum result. However this effect has been minimised by limiting the region of particle selection using IMU data and providing an effective supervision mechanism using the image-based system.

The Weiss system includes a method to detect where the image-based system is not reliable. Our system also incorporates a similar concept, however multiple criteria have been introduced to provide a more comprehensive approach.

In the Weiss system, in order to mitigate the effect of the occasional drift, the use of additional sensors like compass, magnetometer and GPS have been suggested. Our system however provides a novel method of past state correction using continuous epipolar geometry and optical flow, where no additional sensor is required.

The Weiss system considers both camera pose and inter-sensor states in the state vector. This is useful in some applications, however in most applications, in particular augmented reality, the camera and IMU are contained within one module (a mobile device for example). Therefore the inter-sensor parameters are known. Nevertheless such parameters can be calculated using accurate offline methods. As the Weiss system requires no off-line calibration, it has some benefits in applications where an MAV travels over a long time. Our system relies on off-line calibration, however due to the presence of accurate and effective off-line calibration methods the lack of on-line calibration is not a limiting factor. Such methods can also apply during the operation of the MAV but as a background process, without affecting the real-time operation of the system. The sFly dataset used in this chapter provides an accurate measure of inter-sensor parameters (refer to 7.24 to 7.27). The camera intrinsic parameters have been assumed known in both Weiss and our method.

The results presented in this chapter show a stable and accurate tracking performance, with an RMS error of less than 0.06% over 100m simulated travelled distance, which is an improved performance compared to the 0.1% error reported on PTAM or Visual SLAM. With regards to real data, Weiss applied the tracking method on an ellipsoid trajectory. The RMS error ratio over a travelled distance of 9.4m was approximately 0.6%. Our tracking method was also applied to the sFly trajectory, resulting in a tracking error ratio of approximately 0.54% over 11.2m travelled distance.

7.5 Summary

This chapter provided a framework for evaluating the performance of the proposed hybrid tracking solution. In order to examine various aspects of the system, a set of synthetic ground truth data was created, from which the IMU and vision data were generated whilst taking account of their noise and error models. The synthetic data was used to examine the effect of various system parameters such as sampling rate, FoE quality, increased acceleration and angular velocity. The proposed solution was then applied to a set of real data provided by the sFly project. This dataset provided ground truth gathered by a Vicon system whilst IMU and image information was provided by the IMU and camera on-board the MAV.

The results show a stable and accurate tracking performance, with the total mean error of approximately 3.6cm over 49m or 6.63cm over 97m distance, equivalent to around 0.07% of the travelled distance. Both cases exhibited a performance comparable with the reported error for PTAM (Klein & Murray, 2007) but suitable for wide area applications with an improved computational cost. Using simulated data, various aspects of the system, such as the effect of quality of FoE, image sampling rate, as well as increased acceleration and angular velocity were examined. It was shown that the system can detect instances where a reliable FoE does not exist and switch autonomously to IMU-only tracking. The result of experimentation using sFly real-world data also showed similar performance with error in the region of a few centimetres, comparable with the outcome of Weiss's VBN system (Weiss, 2012), but with an improved computational cost.

CHAPTER EIGHT

8 Conclusions, Discussion and Future Work

8.1 Conclusions

Pose tracking has a range of applications from augmented reality to navigation and control. Its widespread applications have been a driving factor and motivation for this PhD work. The thesis has presented a framework for hybrid marker-less inertial-visual camera pose tracking with integrated sensor fusion, addressing a fundamental problem in computer vision and robotics. The main contribution of the work has been to provide a scalable solution for real-time wide-area tracking.

In order to arrive at an improved pose tracking method, an in-depth investigation was conducted into current methods and the variety of sensors used for pose tracking. Alternative algorithms and methods were considered and analysed. Multisensory approaches were reviewed and a mobile GPS-Visual SLAM tracking system was developed and evaluated for its potential suitability in wide-area tracking. A set of experiments carried out with this hybrid GPS-Visual tracking system has demonstrated the potential for utilising visual images from the real-world for obtaining more accurate GPS location on a mobile device. The fusion of GPS data with the Visual SLAM algorithm was found to enhance the performance over a GPS-only system. The multisensory approach improved the typical low range accuracy of GPS, although was found to suffer from cumulative error problems, especially in environments with less

distinctive visual features since, under these conditions, when the original reference scene moved too far out of view, Visual SLAM was not able to accumulate and maintain a sufficiently accurate map of the environment. A further experiment was conducted using wireless location tracking beacons based on ©Apple's iBeacons as an alternative or supplement to GPS for achieving micro-location tracking in both indoor and outdoor environments. This experiment indicated a lower accuracy for micro-location estimation than would be hoped for, although somewhat better in accuracy than raw GPS and, of course, operable in GPS-denied environments.

While wireless systems such as GPS enable location tracking at a macro level, location accuracy can be improved and orientation tracking can be achieved using a hybrid visual-inertial approach. Multisensory systems comprising of both inertial devices (i.e. gyroscope and accelerometer) and camera(s) as motion sensors has proved suitable for use in pose recovery, despite the pros and cons of each of the individual components. An IMU can capture motion at a much faster rate than a typical camera. This makes an IMU a suitable sensor for detecting rapid motion. However, the pose estimate derived from an IMU exhibits significant drift over time, making it unsuitable as the sole sensor for tracking. On the other hand, vison-based tracking using camera as the motion sensor often provides robust and accurate pose information, although only up to determination of a scaling factor which arises due to the fact that image based systems lose a dimension during the 3D to 2D transformation.

In this thesis, particle filtering was adopted as the method of data fusion and a state-space model was selected as the backbone of an improved tracking system proposal. In order to develop this new framework for tracking, the mathematical and physical principles used for particle selection and evaluation, state estimation and formation of decision-making criteria were formulated and presented in a fashion suited to the intended purpose.

The thesis has provided the details of such models and formulations, including; the particle evaluation and formation mechanisms, image formation and manipulation, 3D transformation and epipolar geometry. It has also covered the defined state-space model and the mathematical principles behind the system, in particular the concepts of optical flow focus of expansion and associated properties necessary for particle evaluation. A hybrid tracking system has then been designed and implemented consisting of an inertial system, a vision-based system and stochastic data fusion as its three main components. The design of the proposed system has been presented in detail.

Finally, the work carried out to evaluate the performance of the system has been presented. This evaluation was conducted using both synthetic and real data. First a set of synthetic data was generated, reflecting real IMU and image sensor characteristics. The results show a stable and accurate tracking performance, with an RMS error of less than 0.06% over 100m travelled distance, which is an improved performance compared to the 0.1% error reported on PTAM or Visual SLAM. Using simulated data, various aspects of the system such as the effects of quality of the FoE, the image sampling rate, and increased acceleration and angular velocity were also examined. It was shown that the system performs as intended under a range of circumstances. The algorithm was also tested on real data using the dataset provided by the SFLY project, with the results showing a good performance compared with the error reported by the SFLY system developers (Scaramuzza, et al., 2014).

8.2 Discussion

The data fusion method utilised in this thesis was a stochastic data fusion technique based on recursive particle filtering. The proposed system had a non-linear measurement model, which could not be linearised, making EKF-based sensor fusion not applicable. The proposed system incorporated an effective particle selection method and an enhanced particle evaluation mechanism. The

particles were selected from a proposal distribution function based on the transitional prior. The observation model was defined based on the properties of optical flow Focus of Expansion (FoE). This model offered a simple and effective method for extracting pose information from image sequences and optical flow, without having to retain a map of feature points in the environment. The particle evaluation method required three images, namely; key, past and current images. The combination of the key and current images was used for evaluating the position element of the particles. The key image mostly remained unchanged, resulting in a stable outcome, with low spatial noise and drift. The velocity element of the particles was evaluated using the current and past images which, due to the high number of tracked feature points between two consecutive images, provided a reliable evaluation.

Measures were considered and applied in order to significantly reduce the computational cost of the sensor fusion and pose tracking. Firstly, the orientation of the camera was determined using an image-based 8-point algorithm independent from the state space equations. Secondly, the accelerometer characteristics such as offset and gain error were considered in the particle proposal distribution and, consequently, were removed from the state vector. The combined effect was a dramatically reduced state vector dimensionality from 28 to 6 (compared to the most recent hybrid system proposed by (Weiss, 2012)). In addition, the formation of FoE, in theory, only requires two feature points, although in practice, due to the image noise and feature detection error, outliers may develop and more than two points will be needed. Due to the use of 8-point algorithm, a minimum of 12 feature points were considered. This number of feature points, compared to the tracking methods such as SLAM and PTAM which rely on a high number of feature points, resulted in a lower computational cost at the same time retaining the effectiveness of the proposed algorithm.

In addition to the core hybrid tracking, a novel past-state correction mechanism based on continuous epipolar geometry and focus of expansion was also

proposed, becoming operative when new reliable image data became available. This additional mechanism was beneficial in correcting system drift. The system, thus, incorporated a set of criteria which enabled the application of past-state correction when it was suitable to do so.

The process of FoE determination also provided metrics for the quality and reliability of the FoE estimation, which were used to determine whether or not the vision-based system, at any particular instant, could provide sufficient information to influence the pose estimate. These measures were successfully utilised in order to develop a mechanism to automatically select the best tracking method from IMU-only, hybrid or hybrid with past-state correction at any given time.

The proposed system was implemented and first evaluated using synthetic data and then using the real dataset provided by the SFLY project. The simulation data provided valuable insight into the operation of the algorithm and could be used as a benchmarking tool in any future development of this or other algorithms. The application of the tracking system to the sensory and ground truth data provided by the SFLY project demonstrated the effectiveness and accuracy of the proposed system.

8.3 Future Work

Seamless wide-area pose tracking: This work has addressed the use of IMU and vision sensors for hybrid pose tracking. Although the combination of these sensors shows encouraging results, none of the sensors alone provides an absolute estimate of pose with reference to the world frame. As discussed earlier, tracking methods such as GPS or sensors such as wireless beacons and magnetometers can provide some form of absolute pose with reference to the world frame. However, none of these sensors can provide accurate pose estimate on their own. Nevertheless, due to their nature, they can correct drift

error over long distances or when a vision-based tracker is incapable of contributing to the hybrid system. As future work, it is recommended that a fusion system be designed to include GPS and magnetometer data in addition to that from an IMU and camera, providing a more ubiquitous solution for tracking over extra-long distances in unknown outdoor and indoor environments. In such a system, a decision making engine would need to be designed to automatically switch from one or multiple sources of tracking to other sources, enabling a seamless tracking solution.

Applications of pose tracking with significant impact: The applications of pose tracking are numerous, however the proposed hybrid tracking system, due to its effectiveness and low implementation cost, could be employed in applications that have significant social impact. Such applications include healthcare, keyhole surgery and location-tracking by emergency services in indoor places. However, the proposed algorithm, although scalable, still needs to be optimised for specific applications. As future work it is suggested that such applications be reviewed, their performance criteria identified, and relevant aspects of the hybrid tracking system's algorithm and decision-making criteria optimised to suit each specific application.

Active control for improved tracking performance: The performance of the proposed system relies on the quality and existence of an optical flow focus of expansion. For navigation and positioning applications, where the camera operates solely for the purpose of tracking, the camera could be motorised in order to enable the system to change the angle between the camera speed vector and image plane, hence ensuring the existence of a focus of expansion. This would require a closed-loop control system operating on the basis of a state-space model. The development of a combined tracking and control system could be a potential area for study with applications in robotics and un-manned vehicle design.

Implementing an adaptive system: The presented framework offers a generic scalable solution for use across a wide range of applications. However, performance requirements will vary depending on application. For a fully versatile tracking solution it is suggested that some of the system parameters be determined adaptively. For instance, in order to form a reliable point of focus of expansion, the distance between the corresponding feature points in the two images of interest must be significantly more than the feature detection error. For velocity-based focus of expansion, this is sometimes hard to achieve since the two consecutive images are often in close proximity. This could be solved by having an adaptive image sampling rate so that, when the travel speed is low, images are captured at a lower rate than when the speed is high. As future work it is suggested that all such aspects of the tracking system be identified and replaced by a suitable adaptive alternative.

8.4 Final Remarks

The aim of this PhD work was to propose a multisensory solution for pose tracking, which can potentially be used in wide-area and on mobile platforms. It is the author's belief that considering the proposed framework, its implementation and evaluation, this aim has been achieved within the defined scope of the PhD program. However, as outlined above, some further work is still recommended in order create a completely robust and adaptive system, which can be used in unknown circumstances and commercial applications. Nevertheless, the author presents this work as a way forward and as a platform on which to base future research towards a fully comprehensive pose tracking technology.

Bibliography

A

Ahamed, S., Talukder, N. & Monjur, M. (2008). WiFi Radar: Design and Implementation of an Infrastructure-less Location Tracking System for Pervasive Environment. *2nd Annual IEEE International Computer Software and Applications, COMPSAC '08. 3,* pp. 335 - 338 .

The Amazon (2013), Amazon_PrimeAir, (2014). [Online]
Available at: http://www.amazon.com/b?node=8037720011 [Accessed: 2 July 2014]

Angelaki, D. & Cullen, K., (2008) Vestibular system: the many facets of a multimodal sense.. Annual Review of Neurosciences, Volume 31, p. 125–150.

Argon, *(*2011-2014) *Georgia Institute of Technology, Atlanta, Georgia 30332* [Online]
Available at: http://ael.gatech.edu/argon/ [Accessed: 2 July 2014]

Armstrong, M. & Zisserman, A., (1995). Robust object tracking. *Asian Conference on ComputerVision (ACCV), Singapore,* pp. 58-62.

ARToolKit, 1999. *ArToolKit.* [Online] Available at:
http://www.hitl.washington.edu/artoolkit/ [Accessed 29 June 2014].

Arulampalam, M. S., Maskell, S., Gordon, N. & Clapp, T., (2002). A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing,* 50(2), pp. 174-188.

ARVIKA Project (2009). *ARVIKA .* [Online] Available at:
http://www.arvika.de/www/index.htm [Accessed 1 July 2014].

Ascension (1986). *Ascension Technology Corporation.* [Online] Available at:http://www.ascension-tech.com/ [Accessed 20 July 2014]

Azarbayejani, A. & Pentland, A. (1995) Recursive estimation of motion structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI),* 17(6), pp. 562 - 575.

Azuma, R. (1995) *Predictive tracking for augmented reality - Phd Thesis.* UNCCS, TR-95-007: UNCCS.

Azuma, R. (1997). A Survey of Augmented Reality and Virtual Environment. *Presence-Teleoperators and Virtual Environments,* Volume 6, p. 355–385.

Azuma, R.; Hoff, B.; Neely, H.; Sarfaty, R. (1999) A Motion-Stabilized Outdoor Augmented Reality System, *In IEEE Proceedings of Virtual Realitypp, 252-259*

Azuma, R., (2001). Recent Advances in Augmented Reality. *IEEE Computer Graphics and Applications,* 21(6), pp. 34-47.

Azuma, R. et al. (2010) Tracking in unprepared environments for augmented reality systems. *Computers & Graphics,* 23(6), p. 787–793.

B

Bahl,P. and Padmanabhan, V. N. (2000) RADAR: An In-Building RF-Based User Location and Tracking System. *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies.,* p. 775– 784.

Bailey, T. & Durrant-Whyte, H. (2006). Simultaneous Localization and Mapping: Part I. *IEEE Robotics & Automation Magazine,* 13(2), pp. 99-110.

Bay, H., Ess, A., Tuytelaars, T. & Gool, L. (2008). Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding,* 110(3), p. 346–359.

BBC News, (2013). *BBC Technology News - Amazon testing drones for deliveries.*

[Online] Available at: http://www.bbc.co.uk/news/technology-25180906 [Accessed 1 July 2014]

Beder, C. and Steffen, R. (2006), Determining an initial image pair for fixing the scale of a 3d reconstruction from an image sequence, *In Proceedings of the German Pattern Recognition Association Symposium (DAGM)*,Springer, vol (4174) pp. 657-666

Berrabah, SA. , Sahli, H. , Baudoin, Y. (2011) Visual-based simultaneous localization and mapping and global-positioning system correction for geo-localization of a mobile robot. *Measurement Science and Technology ,* 22(12)

Besl, P. and McKay, N. (1992). A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Bishop, G. (1984) *The self-tracker: A smart optical sensor on silicon,* University of North Carolina at Chapel Hill, Chapel Hill, USA: Unpublished doctoral Dissertation,University of North Carolina at Chapel Hill, Chapel Hill, USA.

Blake, A. and Isard, M. (1998). Active Contours. *Springer.*

Bleser, G. (2009). *Towards Visual-Inertial SLAM for Mobile Augmented Reality, PdH Thesis.* Darmstadt: University Koblenz.

Bleser, G. and Stricker, D. (2008) , *Advanced tracking through efficient image processing and visual-inertial sensor fusion, IEEE Virtual Reality Conference, VR08 - pp.137 – 144*

Bleser, G., Wuest, H. and Stricker, D. (2006), Online camera pose estimation in partially known and dynamic scenes. *Proc. 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR06), pp.* 56 – 65

Born, M. and Wolf, E. (1999) *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light.* Cambridge : Cambridge University Press

Bowring, B. (1985) , The accuracy of geodetic latitude and height equations , *Survey Review,* 28(218), pp. 202-206

Breitmeyer, B. (1977) ,Temporal studies with flashed gratings: Inferences about human transient and sustained channels. *Vision Research,* 17(7), p. 861–865

Broida, T., Chandrashekhar, S. and Chellappa, R. (1990) Recursive 3-d motion estimation from a monocular image sesequence. *IEEE Transactions on Aerospace and Electronic Systems,* p. 639–656

Burger, W. and Bhanu, B., (1990) Estimating 3-D Egomotion from Perspective Image Sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 12(11), pp. 1040-1058

Burton, R. (1973) *Real-time measurement of multiplethree-dimensional positions,* University of Utah, Salt Lake City: Unpublished doctoral dissertation

Burton, R. P. and Sutherland, I. E. (1974) Twinkle Box: Three-dimensional computer-input devices. *AFIPS Conference Proceedings, 1974 National Computer Conference ,* Volume 43, p. 513–520.

C

Calonder, M. , Lepetit, V. , Özuysal, M. ,Trzcinski, T. , Strecha, C. , Fua, P. (2012) BRIEF: Computing a Local Binary Descriptor Very Fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence ,* 34(7), pp. 1281-1298 .

Carmigniani, J. , Furht, B. , Anisetti, M. , Ceravolo, P. , Damiani, E. and Ivkovic, M (2011) Augmented reality technologies, systems and applications. *Multimed Tools Applications, Springer,* Volume 51, p. 341–377

Caudell, T. and Mizell, D. (1992) Augmented reality: an application of heads-up display technology to manual manufacturing processes. *Proceedings of the Twenty-Fifth*

Hawaii Conference, System Sciences, Volume 2, pp. 659 - 669.

Chandaria, J., Thomas, G. and Stricker, D. (2007) The MATRIS project: real-time markerless camera tracking for Augmented Reality and broadcast applications. *Journal of Real-Time Image Processing,* 2(2-3), pp. 69-79.

Chekhlov, D., Gee, A. P., Calway, A. and Mayol-Cuevas, W.(2007) Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. *IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 1-4*

Chen, Y. and Medioni, G. (1991) Object modelling by registration of multiple range images. *IEEE International Conference on Robotics and Automation.* 3 pp 2724 - 2729
Chiuso, A., Favaro., P., Jin., H. and Soatto, S. (2002) Structure from motion causally integrated over time. *In IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI),* 24(4), p. 523–535.

Civera, J. , Grasa, O. ,Davison, A. J. and Montiel, J. M. M., 2010. 1-Point RANSAC for EKF Filtering. Application to. Real-Time Structure from Motion and Visual. Odometry. *Journal of Field Robotics,* 27(5), p. 609–631.

CodaMotion (1970) *CodaMotion Solutions.* [Online] Available at: http://www.codamotion.com/index.php [Accessed 20 July2014]

Comport, A., Kragic, D., Marchand, E. and Chaumette, F. (2005) Robust real-time visual tracking: comparison,theoretical analysis and performance evaluation. *In Proc. IEEE International Conference In Proc. IEEE International Conference,* p. 2852–2857

Comport, A., Marchand, E., Pressigout, M. & Chaumette, F. (2006) Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Transactions on Visualization and Computer Graphics,* 12(4).

Cook, B. , Buckberry, G. , Scowcroft, I. , Mitchell, J. , Allen, T. (2006) Indoor Location Using Trilateration Characteristics. *Proceedings London Communications Symposium.*

Corke, P., Lobo, J. and Dias, J. (2007) An Introduction to Inertial and Visual Sensing.

The International Journal of Robotics Research, 26(6), pp. 519-535 .

Corke, P., Lobo, J. and Dias, J. (2010) An Introduction to Inertial and Visual Sensing. *The International Journal of Robotics Research,* 29(2-3), pp. 231-244.

Cornelis, K. (2004) *From Uncalibrated Video to Augmented Reality. PhD thesis,* Naamsestraat 22, 3000 Leuven, Belgien: Catholic University Leuven.

Cullen, K. (2012) The vestibular system: multimodal integration and encoding of self-motion for motor control. *Trends in Neurosciences,* 35(3), pp. 185-196.

D

Davison, A. J. (2003) Real-time simultaneous localisation and mapping with a single camera. *Proceedings of the International Conference on Com- puter Vision (ICCPV)*

Davison, A. J., Mayol, W. and Murray, D. (2003) Real-time localization and mapping with wearable active vision. *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality,* pp. 18 – 27

Desouza, G. and Kak, A. (2002) Vision for Mobile Robot Navigation: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 237-267.

Dissanayake, G. , Newman, P. , Durrant-Whyte, H.F. ,Clark, S. , Csobra, M. (2001) A solution to the simultaneous localisation and mapping (SLAM) problem. *In IEEE Transactions on Robotics and Automation,* 17(3), p. 229–241.

Dong, Z., Zhang, G. F., Jia, J. Y. & Bao, H., 2009. Keyframe-based Realtime Camera Tracking. *IEEE 12th International Conference on Computer Vision,* pp.1538 - 1545 .

Droeschel, D., May, S. and Behnke, S. (2011) Fusing Time-of-Flight Cameras and Inertial Measurement Units. *Automatika – Journal for Control, Measurement, Electronics, Computing and Communications,* 52(3), p. 189–198.

Drummond, T. and Cipolla, R. (2002) Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 24(7), p. 932–946.

Durrant-Whyte, H. and Bailey, T. (2006) Simultaneous Localization and Mapping: Part II. *IEEE IEEE Robotics & Automation Magazine,* 13(3), pp. 108-117.

E

Eade, E. and Drummond, T. (2007) Monocular SLAM as a Graph of Coalesced Observations. *IEEE 11th International Conference on Computer Vision, ICCV ,* pp. 1-8.

Eade, E., Reitmayr, G. and Drummond, T., 2007. Semi-automatic annotations in unknown environments. *International Symposium on Mixed and Augmented Reality, pp. 67-70*

Earnshaw, R. , Jones, H. and Gigante, M. (1993)*"Virtual Reality Systems".* Academic Press Inc.

Elliot, D. (1996) *Understanding GPS: principles and applications.* E Kaplan, Artec House Publishing.

Endres, C., Butz, A. and MacWilliams, A. (2005) A survey of software infrastructures and frameworks for ubiquitous computing. *Mobile Information Journal,* vol. 1, pp. 41-80

ExactEdition (2013) , *Promotion by iBeacons.* [Online] [Accessed July 2014].Available at: http://blog.exacteditions.com/2013/11/29/promotion-by-ibeacons/

F

Farrell, J. (1999) *The Global Positioning System & Inertial Navigation ,* McGrawHill

Faugeras, O. and Luong, Q.-T. (2001) *The Geometry of Multiple Images.* Cambridge, Massachusetts: The MIT Press, Massachusets Institute of Technology.

Feiner, S., MacIntyre, B., Höllerer, T. and Webster, A. (1997) A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. *Personal and Ubiquitous Computing, Springer,* 1(4), pp. 208-217.

Fiala, M. (2005) ARTag, a fiducial marker system using digital techniques. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR,* Volume 2, pp. 590 - 596.

Fiala, M. (2010) Designing Highly Reliable Fiducial Markers. *IEEE Transactions On Pattern Analysis And Machine Intelligence,* 32(7), pp. 1317 - 1324.

Forsyth, D. and Ponce, J. (2003) *Computer Vision: A Modern Approach :* Pearson Fraundorfer, F. and Scaramuzza, D. (2012) Visual Odometry Turotial - Part II: Matching, Robustness, Optimization, and Applications. *IEEE Robotics & Automation Magazine,* 19(2), pp. 78-90

Fukuju, Y., Minami, M., Morikawa, H. and Aoyama, T. (2003) DOLPHIN: An Autonomous Indoor Positioning System in Ubiquitous Computing Environment. *Proc. IEEE Workshop on Software Technologies for Future Embedded Systems,* pp. 53-56.

G

Gee, A. P., May 2010. *Incorporating Higher Level Structure in Visual SLAM. PhD thesis.,* Bristol, UK: University of Bristol.

Gennery, D. B. (1992) Visual tracking of known three-dimensional objects. *International Journal of Computer Vision,* 7(3), pp. 243-270.

H

Hager, G. and Belhumeur, P. (1998) Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 20(10), p. 1–15.

Hamming, R. W. (1950) Error Detecting and Error Correcting Codes. *Bell System*

Technical Journal, 29(2), p. 147–160.

Hanek, R. and Beetz, M. (2004) The contracting curve density algorithm: Fitting parametric curve models to images using local self-adapting separation criteria. *International Journal of Computer Vision,* p. 233–258

Haralick, R. M., LeeC., Ottenberg, K. and Nolle, M. (1994) Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision,* 13(3), pp. 331-356.
Harris, C. (1992) Tracking with rigid models. *Active Vision,* p. 59–73.

Hartley, R. and Zisserman, A. (2004) *Multiple View Geometry in Computer Vision.* Cambridge: Cambridge University Press.

Hochdorfer, S. and Schlegel, C. (2010) 6 DoF SLAM using a ToF Camera:The challenge of a continuously growing number of landmarks. *The 2010 IEEE/RSJ International Conference on Intelligent Robots and System(IROS), pp.* 3981-3986

Hockney, D. (2001) *Secret Knowledge (New and Expanded Edition): Rediscovering the Lost Techniques of the Old Masters ,* Viking Press

Holm, S. (2005) Airborne ultrasound data communications: the core of an indoor positioning system. *IEEE Ultrasonics Symposium,* Volume 3, pp. 1801-1804 .

Holm, S., Hovind, O., Rostad, S. & Holm, R. (2005) Indoors data communications using airborne ultrasound. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05).,* Volume 3, pp. 957 - 960.

Hughes, H. (1999), *Sensory Exotica: A World Beyond Human Experience.* Cambridge, MA: MIT Press.

I InvenSense (2003) *InvenSense Inc.* [Online] [Accessed Octobere 2013] Available at: http://www.invensense.com/index.html [Accessed 29 June 2014]

Irschara, A. (2012) *Scalable Scene Reconstruction and Image based Localization. PhD thesis.* Graz: Institute for Computer Graphics and Vision , Graz University of Technology

Irschara, A., Zach, C., Frahm, J.-M. and Bischof, H. (2009) From structure from motion point clouds to fast location recognition. *IEEE Conference on Computer Vision and Pattern Recognition,CVPR,* pp. 2599 - 2606

Irschara, A., Zach, C., Frahm, J.-M. and Bischof, H. (2009) From structure-from-motion point clouds to fast location recognition. *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 2599 – 2606

J

Jeroen D. Ho, J. et al., 2007. Robust real-time tracking by fusing measurements from inertial and vision sensors. Journal of Real-Time Image Processing, 2(2-3), pp. 149-160.

Jiang, B. and Neumann, U. (2001) Extendible tracking by line autocalibration. *Proc. IEEE and ACMInternational Symposium on Augmented (ISAR'01),* pp. 97-103

K

Kato, H. and Billinghurst, M. (1999) Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System. *Proc. Second IEEE and ACM International Workshop Augmented Reality,* Volume 85, pp. 20-21.

Kjrergaard, M. B. and Munk, C. V. (2008) Hyperbolic location fingerprinting: A calibration-free solution for handling differences in signal strength. *Sixth Annual IEEE International Conference on Pervasive Computing and Communications,* Volume 08, pp. 110-116

Klein, G. (2006) *Visual Tracking for Augmented Reality, Phd Thesis,* Cambridge, UK: University of Cambridge, Department of Engineering.

Klein, G. & Murray, D. (2007) Parallel tracking and mapping for small AR workspaces. *IEEE and ACM, Symposium on International Mixed and Augmented Reality,* pp. 225-234

Kruijff, E. and Veas, E. (2007) R-Transforming Handheld Augmented Reality. *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality,* pp. 1-2.

L

Layar (2014)*Layar,Augmeneted Reality Platform.* [Online] Available at: https://www.layar.com/ [Accessed June 2014]

Lee, G. H., Achtelik, M. , Fraundorfer, F. , Pollefeys, M. and Siegwart, R. (2010) A benchmarking tool for MAV visual pose estimation, *In IEEE International Conference on Control, Automation, Robotics and Vision (ICARCV10)*

Lim, H., Sinha, S., Cohen, M. and Uyttendaele, M. (2012) Real-time image-based 6-DOF localization in large-scale environments. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Lobo, J. and Dias, J. (2003) Vision and inertial sensor cooperation using gravity as a vertical reference. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 25(12), p. 1597–1608

Lowe, D. (1992) Robust Model-based Motion Tracking Through the Integration of Search and Estimation. *International Journal of Computer Vision,* 8(2), pp. 113-122

Lowe, D. (2004) Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision,* 60(2), pp. 91-110.

M

Macii, D., Trenti, F. and Pivato, P. (2011) A robust wireless proximity detection technique based on RSS and ToF measurements. *IEEE International Workshop on Measurements and Networking Proceedings (M&N), ,* pp. 31 - 36 .

Maidi, M., Didier, J.-Y., F., A. & Mallem, M., (2010) A performance study for camera pose estimation using visual marker based tracking. *Machine Vision and Applications, IAPR International Journal, Springer,* 21(3), p. pp. 365–376.

Maidi, M., Preda, M. and Van Hung Le (2011) *Markerless tracking for mobile augmented reality.* , *IEEE International Conference on Signal and Image Processing Applications (ICSIPA),* pp. 301 – 306

Mautz, R. and Tilch, S. (2011) Survey of Optical Indoor Positioning Systems. *International Conference on Indoor Positioning and Indoor Navigation (IPIN),* pp. 1-7.

Mayol, W. and Murray, D. W. (2008) Tracking with General Regression. *Machine Vision and Applications,* 19(1), pp. 65-72

Ma, Y., Soatto, S., Kosecka, S. and Sastry, S. (2004) An invitation to 3D Vision – From Images to Geometric Models,. New York: Springer-Verlag.

Ma, Y., Soatto, S. and Kosecka, J. (2004) An Invitation to 3D Vision , Publisher: Springer-Verlag, pp. 211-214.

MESA (2006) *MESA Imaging.* [Online] Available at: http://www.mesa-imaging.ch/home/ [Accessed 26 July 2014]

Mingyang, L. and Mourikis, A. (2012) Vision-aided inertial navigation for resource-constrained systems. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) ,* pp. 1057 - 1063

Misra, P. and Enge, P. (2001) *Global Positioning System Signals, Measurements, and Performance.* Lincoln, Massachusetts: Ganga-Jamuna Press.

Moemeni, A. and Tatham, E. (2014) Inertial-Visual Pose Tracking using Optical Flow-aided Particle Filtering. *IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2014).*

Moemeni, A. & Tatham, E. (2010) A framework for camera pose tracking using stochastic data fusion. In *Games Innovations Conference International IEEE Consumer Electronics Society's , (ICE-GIC10),pp. 1-7*

Montemerlo, M., Thrun, S., Koller, D. & Wegbreit, B., 2003. Fast-SLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. *in Proc. Int. Joint Conf. Artif. Intell.,* p. 1151–1156.

N

Noe, P. and Zabaneh, K. (1994) Relative GPS. *IEEE Position Location and Navigation Symposium,* pp. 586-590.

O

Ohno, K., K., T. and Nomura, S.(2006) Real-Time Robot Trajectory Estimation and 3D Map Construction using 3D Camera. *IEEE International Conference on Intelligent Robots and Systems (IROS).*

P

Papandrea, M. and Giordano, S. (2012) Enhanced Localization Solution. *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops),* pp. 241 - 246 .

Park, J., You, S. and Neumann, U. (1998) Natural feature tracking for extendible robust augmented realities. *Proc* IWAR '98 Proceedings of the International Workshop on Augmented Reality, pp. 209-217

Pavani, T., Costa, G., Mazzotti, M. and Conti, A. (2006) Experimental Results on Indoor Localization Techniques through Wireless Sensors Network. *IEEE Vehicular Technology Conference,* Volume 2, pp. 663 – 667

Pivato, P., Fontana, L., Palopoli, L. and Petri, D. (2010) Experimental Assessment of a RSS-based localization algorithm in indoor environment. *Instrumentation and Measurement Technology Conference (I2MTC),* pp. 416 – 421
PointCloud (2013) *PointCloud SDK for iOS.* [Online]
Available at: http://developer.pointcloud.io/sdk [Accessed January 2014].

Priyantha, N. B., Chakraborty, A. and Balakrishnan, H. (2000) The Cricket Location-Support system. *Proceedings of the 6th annual international conference on Mobile computing and networking ,* pp. 32-43.

R

Ratshidaho, T., Tapamo, J., Claassens, J. and Govender, N. (2012) ToF Camera Ego-Motion Estimation. *Robotics and Mechatronics Conference of South Africa (ROBOMECH),* pp. 1-6.

Raudies, F. and Neumann, H. (2012) A review and evaluation of methods estimating ego-motion. *Computer Vision and Image Understanding,* 116(5), pp. 606-633

Reitmayr, G. and Drummond, T. W. (2006) Going out: robust model-based tracking for outdoor augmented reality.. *IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR 2006.,* p. 109–118.

Rekimoto, J. (1998) A Realtime Object Identification and Registration Method for Augmented Reality. *Proceedings Proceeding Third Asia Pacific Computer,* pp. 63 - 68.

Ristic, B., Arulampalam, S. and Gordon, N. (2004) Beyond the Kalman filter: particle filters for tracking applications. In: Boston, London: Artech House Publishers, pp.48-49.

Rolland,J.P , Baillot,Y. and Goon, A. (2001) A Survey of Tracking Technology For Virtual Environments. *Fundamentals of Wearable Computers and Augmented Reality,* New Jersey: CRC Press, pp. 67-112.

S

Sattler, T., Leibe, B. and Kobbelt, L. (2011) Fast Image-Based Localization using Direct 2D-to-3D Matching. *3th International Conference on Computer Vision, (ICCV)*, pp. 667 - 674

Scaramuzza, D., Achtelik, M.C. , Doitsidis, L. , Fraundorfer, F. , Kosmatopoulos, E. , Martinelli, BA; Achtelik, V. , Chli, M. , Chatzichristofis, S.A. , Kneip, L. , Gurdan, D., Heng, L. , Lee, G.H., Lynen, S., Meie, L. , Pollefeys, M. , Renzaglia, A. and Siegwart, J.C (2014) Vision-Controlled Micro Flying Robots: from System Design to Autonomous Navigation and Mapping in GPS-denied Environments. *IEEE Robotics and Automation Magazine,* 3(21), pp. 26 – 40

Scaramuzza, D. and Fraundorfer, F. (2011) Visual Odometry [Tutorial] - Part I: The First 30 Years and Fundamentals. *IEEE Robotics & Automation Magazine,* 18(4), pp. 80 - 92 .

Schall, G. (2011) *Mobile Augmented Reality for Human Scale Interaction with Geospatial Models, PhD Thesis,* Austria: Graz University of Technology, Institute for Computer Graphics and Vision.

Schall, G. , Mendez, E. , Kruijff,E. , Veas, E., JungHanns, S.,Reitinger, B. and Schmalstieg, D., (2009) Handheld augmented reality for underground infrastructure

visualization. *Personal and Ubiquitous Computing,* 13(4), pp. 281-291.

Schleicher, D., Bergasa, L., Ocana, M. and Barea, R. (2009) Real-time hierarchical outdoor SLAM based on stereovision and GPS fusion. *IEEE Transactions on Intelligent Transportation Systems,* 10(3), pp. 440 - 452 .

Simon, G. (2006) Automatic online walls detection for immediate use in ar tasks. *In IEEE ACM International Symposium on Mixed and Augmented Reality (ISMAR06) pp.* 39 – 42

Steidle, F.,Tobergte, A. and Hirzinger, G.(2012), Optical-Inertial Tracking with Active Markers and Changing Visibility. *International Conference on Intelligent Robots and Systems (IEEE/RSJ),* pp. 3978 - 3984.

Strasdat, H., Montiel, J. M. M. & Davison, A., February (2012) . Visual SLAM: Why filter?. *Image and Vision Computing,* 30(2), pp. 65-77

Strasdat, H., Montiel, J. M. M. and J., D. (2010) Real-time monocular slam: Why filter?. *IEEE International Conference on Robotics and Automation (ICRA)*

StryKer, 1894. *StryKer.* [Online] Available at: http://www.stryker.com/en-us/corporate/AboutUs/index.htm [Accessed 20 July 2014]

Stüber, G. and Caffrey, J. (1999) Radiolocation Techniques, Chapter 24. In: J. Gibson, ed. *The mobile Communications Handbook.* 2 ed. s.l.:CRC Press, pp. 1-12

Subramanian, A., Deshpande, P., Gaojgao, J. and Das, S. (2008) Drive-By Localization of Roadside WiFi Networks. *The 27th Conference on Computer Communications, IEEE NFOCOM,* pp. 718-725.

Sutherland, I. (1968) A head-mounted three dimensional display. *ACM, fall joint computer conference, part I ,* p. 757–764

T

Thomas, G. A. (2007) Mixed Reality Techniques for TV and their Application for On-Set and Pre-Visualization in Film Production. *BRITISH BROADCASTING CORPORATION (BBC R&D Publications)*

Thomas, G.A., Jin, J., Niblett, T. and Urquhart, C. (1997) A versatile camera position measurement system for virtual reality TV production. *Proceedings of IBC'97,* p. 284-289

Thrun, S., Burgard, W. and Fox, D. (2005) *Probabilistic Robotics, ser. Intelligent Robotics and Autonomous Agents.* MA, USA: Cambridge : The MIT Press

Titterton, D. & Weston, J. (2004) *Strapdown Inertial Navigation Technology.* 2 ed., *The American Institute of Aeronautics and Astronautics*

Tola, E., Lepetit, V. & Fua, P.(2010) DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 32(5), pp. 815-830.

Triggs, B., McLauchlan, P. F., Hartley, R. I. & Fitzgibbon, A. W. (2000) Bundle Adjustment — A Modern Synthesis. *In International Workshop on Vision Algorithms: Theory and Practice, volume 1883, pages 298–372 - Springer-Verlag.*

Trucco, E. and Plakas, K., 2006. Video Tracking: A Concise Survey. *IEEE Journal of Oceanic Engineering ,* 31(2), pp. 520 - 529 .

V

Vacchetti, L., Lepetit, V. V. and Fua, P. (2004) Stable real-time 3d tracking using online and offline information. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 26(10), pp. 1385-1391.

Van Krevelen, D. and Poelman, R. (2010) A Survey of Augmented Reality Technologies, Applications and Limitations. *The International Journal of Virtual Reality,* 9(2), pp. 1-20.

Vicon, 1984. *Vicon Motion Systems Ltd.* [Online] Available at: http://www.vicon.com/ [Accessed 29 June 2014]

Viéville, T. & Faugeras, O., 1990. Cooperation of the inertial and visual systems. *In Traditional and NonTraditional Robotic Sensors (ed. T. C. Henderson), SpringerVerlag, Berlin, Heidelberg,* Volume F63, p. 339–350

W

Wagner, D. and Schmalstieg, D. , ARToolKitPlus for Pose Tracking on Mobile Devices. *Proceedings of 12th Computer Vision Winter Workshop (CVWW'07)*

Wagner, D., Pintaric, T., Ledermann, F. & Schmalstieg, D. (2005) Towards massively multi-user augmented reality on handheld devices. *Pervasive Computing,Springer,* p. 208–219.

Wagner, D. , Reitmayr, G., Mulloni, V, Drummond, T. and Schmalstieg, D. (2010) Real-time detection and tracking for augmented reality on mobile phones. *IEEE Transactions on Visualization and Computer Graphics,* 16(3), pp. 355-368.

Wang, J., Hu, G., Huang, S. and Dissanayake, G. (2009) 3D Landmarks Extraction from Range Imager Data for SLAM. *Australasian Conference on Robotics adn Automation (ACRA).*

Ward, A., Jones, A. & Hopper, A., 1997. A New Location Technique for the Active Office. *IEEE Personal Communications,* 4(5), pp. 42-47.

Ward, M., Azuma, R. T. , Bennett, R. , Gottschalk, S. and Fuchs, H. (1992) *A demonstrated optical tracker with scalable work area for head-mounted display*

systems., Cambridge, MA: Symposium on Interactive 3D Graphics.

Weiser, M. (1999).The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communications,* Volume 3 (3), pp. 3-11

Weiss, S. M. (2012) *Vision Based Navigation for Micro Helicopters.* Phd Thesis, The ETH: Zurich

Welch, G. , Bishop, G , Vicci, L. , Brumback, S. , Keller, K. and Colucci, D. (2001) High-Performance Wide-Area Optical Tracking, The HiBall Tracking System. *Presence: Teleoperators and Virtual Environments,* 10(1), pp. 1-21

Welch, G. and Foxlin, E. (2002) Motion tracking: no silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications,* 22(6), pp. 24 – 38

Wikitude (2014) *Wikitude.* [Online] Available at: http://www.wikitude.com/

Woltring, H. J. (1974) New possibilities for human motion studies by real-time light spot position measurement. *Biotelemetry,* Volume 1, pp. 132-146

Woodman, O. (2007). *An introduction to inertial navigation,* Cambridge, UK: University of Cambridge, UCAM-CL-TR-696 ISSN / ISSN 1476-2986.

Z

Zhang, X., Fronz, S. and Navab, N. (2002) Visual marker detection and decoding in AR systems: a comparative study. *ISMAR,* Volume 02, pp. 97-106

Zhang, X., Genc, Y. and Navab, N. (2001) *Mobile computing and industrial augmented reality for real-time data access,* Princeton, NJ 08540, USA: Technical report, Siemens Corporate Research, Imaging and Visualization Department.

Appendix A : Publications

A.1

2014 IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2014)

Inertial-Visual Pose Tracking using Optical Flow-aided Particle Filtering

Armaghan Moemeni[1] and Eric Tatham[2]
Centre for Computational Intelligence (CCI)
Faculty of Technology, De Montfort University
Leicester, United Kingdom
[1] armaghan@dmu.ac.uk , [2] etatham@dmu.ac.uk

***Abstract*—This paper proposes an algorithm for visual-inertial camera pose tracking, using adaptive recursive particle filtering. The method benefits from the agility of inertial-based and robustness of vision-based tracking. A proposal distribution has been developed for the selection of the particles, which takes into account the characteristics of the Inertial Measurement Unit (IMU) and the motion kinematics of the moving camera. A set of state-space equations are formulated, particles are selected and then evaluated using the corresponding features tracked by optical flow. The system state is estimated using the weighted particles through an iterative sequential importance resampling algorithm. For the particle assessment, epipolar geometry, and the characteristics of focus of expansion (FoE) are considered. In the proposed system the computational cost is reduced by excluding the rotation matrix from the process of recursive state estimations. This system implements an intelligent decision making process, which decides on the best source of tracking whether IMU only, hybrid only or hybrid with past state correction. The results show a stable tracking performance with an average location error of a few centimeters in 3D space.**

Keywords— motion tracking, camera pose tracking, 6DOF, Inertial, IMU, particle filtering, optical flow, focus of expansion, SLAM, PTAM,

INTRODUCTION

A. Motivation

Camera pose tracking is an assisting technology in enabling the accurate and continuous recovery of the six degree of freedom (6DOF) position and orientation of a moving camera, with the most prominent challenges in real-time systems. The potential applications of accurate 6DoF pose tracking are numerous, including entertainment and immersive games, augmented reality, industrial maintenance and engineering, architecture, medicine, assisted living for the elderly, security, education, prototyping and autonomous navigation systems.

Proposals and solutions for pose recovery have been so many in the past few decades. Among all, visual SLAM and PTAM based solutions provided reasonable accuracy especially for the Augmented Reality (AR) applications; however they were mostly reported to be limited in wide area tracking measurements and uncontrolled real-time localization due to the expensive computational cost involved [1], [2].

Moreover, in recent years the hybrid systems consisting of low cost inertial measurement units (IMUs) and the robust and high-dimensional computer vision-aided algorithms have enhanced the performance and agility of the tracking systems [3], [4], [5]. Such solutions have also tackled the hurdles of real-time sensing and localization especially in GPS-denied environments [6], [7].

Advances in MEMS-based inertial sensors have enabled pose estimation in systems such as mobile robots or unmanned micro aerial vehicles (MAVs), often operating in so called 'urban canyon' environments where GPS signals are either unavailable or unreliable. Recently, there has been substantive research and progress in autonomous MAVs such as the EU-funded SFLY (Swarm of Micro Flying Robots) project, which consists of a micro flying robot using only one single on-board camera and an inertial measurement unit (IMU). This vision-controlled MAV is capable of autonomous navigation in the GPS-denied environments [7].

In this paper, we propose a novel algorithm for single camera-pose tracking, addressing the integration of an IMU sensor to compensate for the deficiencies of the vision-based tracking system, while the inherent drift and accumulation errors are in turn rectified by the vision-aided. A set of state space equations are formulated considering the kinematics motion equations. With the aid of recursive particle filtering [8], [9] the state parameters (camera pose) are estimated and evaluated taking into account the geometry of two views (epipolar geometry modelled for a single moving camera at two locations). In assessing the estimated states, the 3D geometric constraint derived from the vision-aided system e.g. the optical flow lines and their focus of expansion (FoE) are carefully considered and incorporated in the tracking system. The overview of the algorithm and system architecture is illustrated in

Figure 1 and detailed in sections 0 , 0 and 0. The experimental setup and results are described in section 0. Section 0 concludes by evaluating the final results based on a series of tests carried out as described in section 0.

B. *Related Work*

Traditionally, in computer vision systems, camera has been used as the only motion sensor for tracking. In the past decades, there have been several advances in computer vision-based tracking techniques in order to recover the 2D/3D correspondences between successive images where two main approaches namely marker-based and marker-less were considered. The former is tracking fixed fiducial markers, which implicitly solves the tracking and localization because the markers and their relative 3D positions are known. Zhang et al [10] have carried out a comprehensive study on the approaches to marker-based tracking methods using fiducial markers. Examples include ARToolKit [11] and ARTag [12] planar fiducial markers, which are mainly used for camera tracking and solving the problem of image registration for AR.

The marker-less approaches however use naturally distinctive features such as points, lines, edges, or textures whose 3D positions are not known. These systems use the naturally driven features for both motion estimation and localization. Comprehensive surveys on monocular camera pose tracking using only vision-based approaches have been carried out in [13], [14] and [15].

Among all techniques, the developments of Visual SLAM (Simultaneous Localization and Mapping) or visual odometry and the PTAM (Parallel Tracking and Mapping) of Klein and Murray [2] can be referred to as the most relevant methods for localization with a single camera in 6DOF. In general the standard Davison's SLAM method [1] is based on tracking and localization of the 'robot' or 'camera' in an unknown environment while a map of the environment is constructed alongside tracking.

The standard SLAM is sometimes referred to filter-based SLAM where Bayesian filters such as Kalman or Extended Kalman Filter (EKF) are used to infer the current state (pose) based on the current observation and past state of the system. The state estimation process generates uncertainty for both the features and the camera pose, which adds to the complexity of the system, making the process computationally expensive. For that reason such filter-based methods in the original format, are not applicable in real-time or wider area tracking applications. However, PTAM, as an enhancement to the filter-based SLAM, splits the simultaneous localization and mapping tasks into two separate threads. The PTAM sometimes referred to as key-frame SLAM where the mapping thread uses a subset of all camera frames i.e. key-frames to build a 3D-point map of the surroundings. [16] demonstrates how key-frame SLAM outperforms the EKF-filter based SLAM. However, taking into account the bundle adjustment and online batch optimization approach, this process is still considered to be computationally expensive and therefore more applicable in smaller workspaces [2].

The system proposed in this paper benefits from the advantages of both filter-based and key-frame based SLAM which will be described in the next section.

OUR APPROACH

The aim of the proposed system here is to estimate the 6DOF pose of the camera, consisting of 3D position and 3D orientation, with reference to a fixed coordinate system referred to as the world frame. The system uses inertial and visual sensors, where the inertial sensor, IMU, is a combined accelerometer and gyroscope and the vision sensor is a monocular camera. The system collects sensory data and by fusing them provides the camera pose estimate. However, both visual and inertial sensors are influenced by measurement noise and error which affect the measurement accuracy.

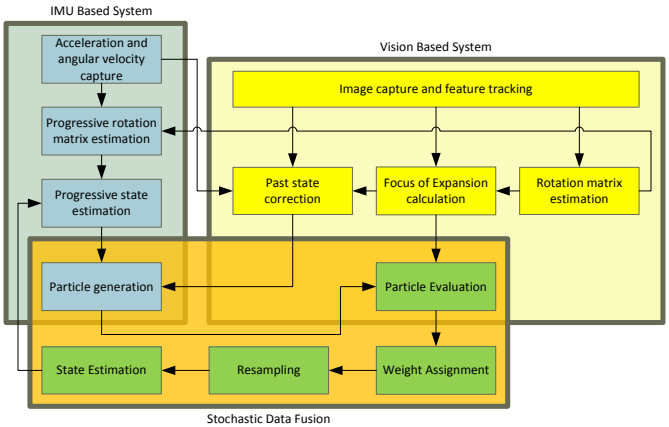


Figure 1 : Sensor Fusion System Architecture

The noise and error cause the IMU-only tracking system to drift significantly over time, making it an unsuitable sensor to be solely used for camera tracking. On the other hand camera-only tracking systems not only suffer from noise and measurement error, but also exhibit an inherent deficiency, which is the inability to estimate all 6 degrees of freedom. The camera-only systems provide 5 degrees of freedom, meaning that the estimate of the camera position can be provided up to a scaling factor.

By combining the information gathered through the two sensors the shortcomings of either system can be addressed and a more accurate estimate of the camera pose estimated.

Traditionally, Extended Kalman Filter (EKF) based methods have been used in inertial visual hybrid tracking systems [4]. However such methods require a high number of states to be estimated at every step, with significant computational cost. These methods also suffer from inaccuracy due to linearization, in particular during fast camera movements. The alternative approach PTAM also has the problem of image accumulation.

In our approach we avoid the image accumulation by using only three images; namely, reference, past and current. The reference image remains constant until its feature points can no longer be tracked. The three-image approach improves the stability of the tracking process.

The proposed method benefits from the concept of focus of expansion (FoE), which due to the way it has been calculated directly provides the measurement error. It also provides a means of determining whether the current image information is accurate enough to be used for performing data fusion at a particular time (see section E). The observation function however is a non-linear complex function, which cannot be effectively linearized through EKF. Therefore the EKF cannot be applied and instead an adapted particle-filtering based method is utilized for the sensor fusion.

A state correction mechanism has been implemented to correct the past state of the system, where new information becomes available. In order to reduce the complexity of the algorithm and number of states, the orientation of the camera is estimated by the vision based system only, where a new image is available, and by IMU only between two consecutive images, where no new image is available. Therefore orientation estimation is removed from the state estimation, thus reducing the number of states to be estimated. The system incorporates an intelligent adaptive mechanism to make a decision whether the tracking must be performed using IMU only, inertial-visual fused data or fused data with past state correction (see section 0).

C. Problem Formulation and Definitions

In this system the position and orientation are estimated separately. The position of the camera (X_t^w) is estimated using the state space equations (see section 0) and the orientation (θ_t^w) via the image based system (see section 0). The superscript in the notation represents the frame of reference, i.e. world frame in this instance. The subscript represents the time. These definitions apply throughout this paper. The camera pose at time t_n, which is the time of the current image is therefore represented by $(X_{t_n}^w, \theta_{t_n}^w)$. Once the camera pose is known the 3D coordinates of a fixed point in space seen at P_t^c in the camera frame can be translated to the 3D coordinates of the same point, but with reference to world frame (P^w) (see Figure 2).

$$(1)\quad \boldsymbol{P^w = X_t^w + R_t^{wc}(\theta_t^w)P_t^c}$$

R_t^{wc} is the rotation matrix which is formed by 3 individual rotations around x, y and z axis each represented by an element of 3D θ_t^w vector.

ORIENTATION ESTIMATION

In order to determine the camera orientation the rotation matrix between the camera frame and world frame is determined (R_t^{wc}). The rotation matrix can be driven directly from the camera orientation vector and vice versa. This correspondence is not unique and depends on the order of individual rotations. In this work it is assumed that the order of individual rotations is known therefore the orientation can be uniquely derived from the rotation matrix.

The rotation matrix is calculated using the Essential Matrix and Singular Value Decomposition (SVD) [23]. In the context of multi-view epipolar geometry as illustrated in Figure 3, the Essential Matrix (E) defines the relationship between corresponding feature points (x^{c1} and x^{c2}) in two calibrated camera views (see equation (2)). The Essential Matrix is related to the Fundamental Matrix (F) by the camera Intrinsic Matrix (K) as per equation (3). The Fundamental Matrix defines the relationship between corresponding feature points in two un-calibrated camera views and is computed using the 8-point algorithm [17].

Considering the Singular Value Decomposition, the Essential Matrix can be expressed as in (4) [18]. The rotation matrix is then calculated using equation (5). This method provides two possible solutions for the rotation matrix. The solution matrix closest to the rotation matrix $\widetilde{R}_t$ (see equation (6)), is considered to be the correct rotation matrix. In this equation, Ω is the IMU angular velocity vector, δt is the time difference between two consecutive images, R_{t-1} is the rotation matrix at time $t-1$ and $DCM(.)$ is the direction cosine matrix of Euler angles.

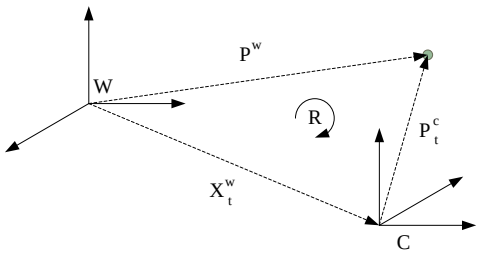


Figure 2 : The Transformations

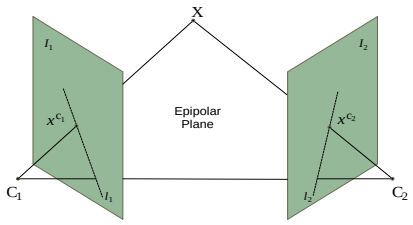


Figure 3 : Epipolar Geometry

(2) $\boldsymbol{x^{c1^T} E\, x^{c2} = 0}$
(3) $\boldsymbol{E = K^T F K}$
(4) $\boldsymbol{E = U\, diag(1,1,0) V^T}$
(5) $\boldsymbol{R = UWV^T \quad where \quad W = ((0,1,0)^T, (-1,0,0)^T, (0,0,1)^T)^T}$
(6) $\boldsymbol{\tilde{R}_t = DCM(\delta t \Omega)\, R_{t-1}}$

POSITION ESTIMATION

The position estimator operates based on kinematics motion equations which are used to formulate the state-space model. The state-space equations are accordingly solved applying recursive filtering techniques. Particle filter consists of a particle selection stage, followed by particle evaluation, weight assignment and state estimation. The particle filtering method employed here is based on Sampling Importance Resampling (SIR) outlined in [9] and [19]. In order to apply this method, a proposal distribution is formed and then particles are randomly selected from this distribution. Subsection D details how particles are generated.

The particles are then evaluated using a likelihood function (see Subsection E for details). An appropriate weight is then assigned to each particle, resampling is applied if necessary and finally current state of the system is estimated (see subsection F). The system state is estimated using weighted summation of the particles.

D. Particle Selection

The tracking system provides an estimate for the state vector of the system. The state vector here consists of two elements, namely 3D position and 3D linear velocity in the world frame (see (7)). The particle selection process operates based on the state-space model, in which the current state of the system is modelled using the past state and control input as well as noise and modelling error as outlined in equations (8) and (9). As described earlier the state space equations are formed using the principles of motion kinematics. A_t^c is the 3D vector of the camera linear acceleration in the camera frame, δt is the time difference between two consecutive images, G is the 3D gravity vector in the world frame, and R_t is the rotation matrix from the camera frame to world frame at time t.

(7) $\mathbf{S_t^w = (X_t^w \quad V_t^w)^T}$, $\mathbf{V_t^w = (v_{x,t}^w \quad v_{y,t}^w \quad v_{z,t}^w)}$
$\mathbf{X_t^w = (x_t^w \quad y_t^w \quad z_t^w)}$

(8) $\mathbf{X_t = X_{t-1} + \delta t V_{t-1} + 0.5 \delta t^2 A_t^w}$, $\mathbf{A_t^w = R_t A_t^c + G}$
(9) $\mathbf{V_t = V_{t-1} + \delta t A_t^w}$

Due to noise and error in angular velocity and acceleration, A_t^w contains noise and measurement error in a non-linear fashion. The IMU is sampled at a high sampling rate (100Hz or above), whereas sampling rate of the image-based system is much lower (typically 50Hz or less). Between the image samples, the IMU is the only source of motion information. The above state-space model is used for estimating the position between two image samples. However the focus of this paper is to estimate the position of the camera at the time of a newly captured image. When a new image is captured the position estimated using the IMU data is as per equation (10). N_i is the number of IMU data sets between two consecutive images and l is the dataset number from 1 to N_i. A_l^c is the acceleration vector and R_l is the rotation matrix at the time of lth packet of IMU data. The particle filtering method employed here is based on SIR filter [9]. In order to select particles, the estimated advance in the system state is calculated using equations (11) and (12). The particles are selected from a normal proposal distribution as per (13) and (14). N_p random values are drawn from the proposal distribution and used to form the particles.

(10) $\boldsymbol{\tilde{X}_t = X_{t-1} + \Delta X_t, \quad \tilde{V}_t = V_{t-1} + \Delta V_t}$
(11) $\boldsymbol{\Delta X_t = \delta t V_{t-1} + 0.5 \delta t^2 \sum_{l=1}^{N_i} (2(N_i - l) + 1)(R_l A_l^c + G)} \quad \boldsymbol{l = 1 \dots N_i}$
(12) $\boldsymbol{\Delta V_t = \delta t \sum_{l=1}^{N_i} (R_l A_l^c + G)} \quad \boldsymbol{l = 1 \dots N_i}$
(13) $\boldsymbol{P_{p,t}^X = P_{p,t-1}^X + \Delta P_{p,t}^X}$, $\boldsymbol{\Delta P_{p,t}^X \sim N(\mu_X, \sigma_X)}$, $\boldsymbol{p = 1 \dots N_p}$
(14) $\boldsymbol{P_{p,t}^V = P_{p,t-1}^V + \Delta P_{p,t}^V}$, $\boldsymbol{\Delta P_{p,t}^V \sim N(\mu_V, \sigma_V)}$, $\boldsymbol{p = 1 \dots N_p}$
(15) $\boldsymbol{(\mu_X, \mu_V) = (\Delta X_t, \Delta V_t)}$
(16) $\boldsymbol{(\sigma_X, \sigma_V) = (\Delta X_{t,max} - \Delta X_{t,min}, \quad \Delta V_{t,max} - \Delta V_{t,min})}$

The accelerometer has measurement noise and error characterize by $e_{n_i}^c$, $e_{os_i}^c$ and $e_{g_i}^c$ which are the noise, offset and the gain error values. These parameters are normally specified in the IMU datasheet. The probable error value for A_l^c is therefore estimated as follows, where x, y and z are three axes of the IMU.

(17) $e_{A_i}^c = e_{n_i}^c + e_{os_i}^c + e_{g_i}^c A_l^c, \quad i = x, y, z$

Variance σ is calculated by inserting the probable maximum and minimum values of A_l^c due to noise and error into equations (11) and (12). The sensor characteristics of InvenSense MPU-6000 are used for an estimate of these values. The differences between the maximum and minimum values are used as the variance (see (16)).

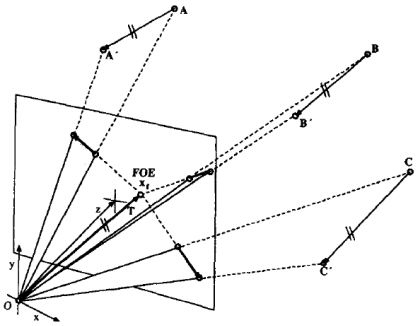


Figure 4 : Optical Flow Lines and the Focus of Expansion, adapted from [20]

E. Particle Evaluation

Camera converts a 3D point in space to a 2D feature point on the image plane, following the pinhole camera model as described by equation (18). In this equation, x^c is the 2D feature point in the calibrated camera view, f is the camera focal length and Z is the distance of the 3D point along the Z axis of the camera. In this work optical flow is used for feature detection and tracking.

Particle evaluation is carried out using the properties of Focus of Expansion (FoE). The FoE point is where all flow lines meet on the image plane. A flow line is the line connecting corresponding feature points on two images. It can be shown that if two images are related to each other only by a translation vector, the translation vector is parallel to the line connecting the camera center to the FoE point [20]. Also for two adjacent images, with a sufficiently small time difference between them (δt), the linear velocity vector is parallel to the above FoE vector. In our system this is the main criteria for evaluating the particles. See Figure 4, which illustrates the concept of the FoE in relation to optical flow lines. Equations (19) and (20) describe these relationships, where $(T_x,\ T_y,\ T_z)^T$ and $(v_x,\ v_y,\ v_z)^T$ are the translation and linear velocity vectors with reference to the camera frame at the time when the first of the two images was captured.

(18) $\boldsymbol{x^c = \frac{f}{z} X^c}$

(19) $\boldsymbol{(x_{FOE},\ y_{FOE},\ f)^T \parallel \left(T_x,\ T_y,\ T_z\right)^T \Rightarrow x_{FOE} = T_x/T_z,\ y_{FOE} = T_y/T_z}$

(20) $\boldsymbol{(T_x,\ T_y,\ T_z)^T \parallel \delta t\,(v_x,\ v_y,\ v_z)^T \Rightarrow x_{FOE} = v_x/v_z,\ y_{FOE} = v_y/v_z}$

In order to evaluate the speed and position particles three images are required. These images are the current image (taken at time t), the past image (taken at time $t-1$) and the reference (or key-frame) image (taken at time t_r). When the tracking starts, the first image is considered as the reference image. The reference image remains unchanged until the number of common feature points with the current image falls below a threshold, in which case the reference image is replaced by the current image. The threshold is defined through heuristic tests.

In the proposed system two types of FoE points are required. The first one is the FoE between the current and past images, with reference to the camera frame at time $t-1$. This is referred to as $\mathrm{FoE_{cp}}$. The second one is the FoE between the current and reference images, with reference to the camera frame at the time that the reference image was taken (t_r). This FoE is referred to as $\mathrm{FoE_{cr}}$.

In order to calculate $\mathrm{FoE_{cp}}$ or $\mathrm{FoE_{cr}}$, first the current image is rotated to have the same orientation as the past or reference camera frames. Then the FoE point is calculated by minimizing the mean square error. The FoE is a point on the image plane where its distance to all optical flow lines is minimized.

Suppose the flow line for each feature point m is characterised as in equation (21). If $(x_{FOE}, y_{FOE})^T$ is considered as the coordinates of the FoE on the image plane, equation (22) gives the average distance of the FoE point to the flow lines. The FoE point is defined as a point where the average distance is minimized; therefore the coordinates of the FoE are where the partial derivatives become zero as per equations (23). By taking particle derivatives a set of two linear equations characterized by equations (24), (25) and (26). Parameters A and B can then be used to derive coordinates of FoE as per equation (27). This method is used for deriving both $\mathrm{FoE_{cp}}$ or $\mathrm{FoE_{cr}}$ using the current and past images and then current and reference images, respectively. Once the $\mathrm{FoE_{cp}}$and $\mathrm{FoE_{cr}}$ points are calculated, the velocity and translation vectors associated with each particle are compared with the corresponding FoE point to produce a score for that particular particle. To do so for each speed or position particle, the corresponding FoE is formed using equations (28) and (29). The distance between the FoE based on particle and the FoE based on image is used as the particle score (see equations (30) and (31).

(21) $\boldsymbol{a_m x + b_m y + c_m = 0}$

(22) $\boldsymbol{d^2 = \frac{1}{m}\sum_{m=1}^{N_f}(a_m x + b_m y + c_m)^2/(a_m^2 + b_m^2)}$

(23) $\boldsymbol{\partial d/\partial x|_{x=x_{FOE}} = 0,\ \ \partial d/\partial y|_{y=y_{FOE}} = 0}$

(24) $\boldsymbol{A\begin{pmatrix} x_{FOE} \\ y_{FOE} \end{pmatrix} + B = \begin{pmatrix} 0 \\ 0 \end{pmatrix}}$

(25) $\boldsymbol{A = \begin{pmatrix} \sum_{m=1}^{N_f} \frac{2a_m^2}{b_m^2+c_m^2} & \sum_{m=1}^{N_f} \frac{2a_m b_m}{b_m^2+c_m^2} \\ \sum_{m=1}^{N_f} \frac{2a_m b_m}{b_m^2+c_m^2} & \sum_{m=1}^{N_f} \frac{2b_m^2}{b_m^2+c_m^2} \end{pmatrix}}$

(26) $\boldsymbol{B = \begin{pmatrix} \sum_{m=1}^{N_f} \frac{2a_m c_m}{b_m^2+c_m^2} \\ \sum_{m=1}^{N_f} \frac{2b_m c_m}{b_m^2+c_m^2} \end{pmatrix}}$

(27) $\boldsymbol{\begin{pmatrix} x_{FOE} \\ y_{FOE} \end{pmatrix} = -A^{-1}B}$

(28) $\boldsymbol{FoE_{x,p} = \begin{pmatrix} T_{x,p}/T_{z,p} \\ T_{y,p}/T_{z,p} \end{pmatrix}}$

(29) $\boldsymbol{FoE_{v,p} = \begin{pmatrix} v_{x,p}/v_{z,p} \\ v_{y,p}/v_{z,p} \end{pmatrix}}$

(30) $\boldsymbol{\left(S_p^x\right)^2 = \left\| FoE_{x,p} - FOE_{cr} \right\|} \qquad \boldsymbol{p = 1 \ldots N_p}$

(31) $\boldsymbol{\left(S_p^v\right)^2 = \left\| FoE_{v,p} - FOE_{cp} \right\|} \qquad \boldsymbol{p = 1 \ldots N_p}$

F. Weight Assignment and State Estimation

A low score represents a good particle match for the FoE point. Therefore any particle with low score should receive a high importance weight. The particle weight is defined as the inverse of the particle score as shown in equations (32). The weights for the velocity and position particles are then separately normalized to have a total sum of one. Once the weight for each particle is calculated, the number of effective particles is estimated using equation (34). If this number is less

than a pre-defined threshold, the particles are resampled following the SIR method described in [19]. This process may be repeated until the number of effective particles exceeds a pre-set limit or the resampling has run for more than a set number of iterations. Finally the current state of the system is estimated using a weighted summation of all particles as shown in equations (35).

(32) $\mathbf{w}_{\mathrm{p}}^{\mathrm{v}} = \mathbf{1}/\mathbf{S}_{\mathrm{p}}^{\mathrm{v}}$ $\quad \mathbf{w}_{\mathrm{p}}^{\mathrm{X}} = \mathbf{1}/\mathbf{S}_{\mathrm{p}}^{\mathrm{X}}$ $\quad \boldsymbol{p} = \mathbf{1} \dots \boldsymbol{N_p}$

(33) $\widehat{\mathbf{w}}_{\mathrm{p}}^{\mathrm{v}} = \mathbf{w}_{\mathrm{p}}^{\mathrm{v}} / \sum_{p=1}^{N_p} \mathbf{w}_{\mathrm{p}}^{\mathrm{v}}$ $\quad \widehat{\mathbf{w}}_{\mathrm{p}}^{\mathrm{X}} = \mathbf{w}_{\mathrm{p}}^{X} / \sum_{p=1}^{N_p} \mathbf{w}_{\mathrm{p}}^{\mathrm{X}}$ $\quad \boldsymbol{p} = \mathbf{1} \dots \boldsymbol{N_p}$

(34) $\boldsymbol{N_{eff}^{v}} = \mathbf{1}/\sum_{p=1}^{N_p} \widehat{\mathbf{w}}_{\mathrm{p}}^{\mathrm{v}}$ $\quad \boldsymbol{N_{eff}^{X}} = \mathbf{1}/\sum_{p=1}^{N_p} \widehat{\mathbf{w}}_{\mathrm{p}}^{\mathrm{X}}$ $\quad \boldsymbol{p} = \mathbf{1} \dots \boldsymbol{N_p}$

(35) $\widehat{\boldsymbol{X}}_{\boldsymbol{t}} = \sum_{p=1}^{N_p} \widehat{\mathbf{w}}_{\mathrm{p}}^{\mathrm{x}} \boldsymbol{P_{p,t}^{X}}$ $\quad \widehat{\boldsymbol{V}}_{\boldsymbol{t}} = \sum_{p=1}^{N_p} \widehat{\mathbf{w}}_{\mathrm{p}}^{\mathrm{v}} \boldsymbol{P_{p,t}^{V}}$ $\quad \boldsymbol{p} = \mathbf{1} \dots \boldsymbol{N_p}$

G. Past State Correction

At time t a new image becomes available and therefore the image velocity vector $(\mathrm{u}, \mathrm{v})^{\mathrm{T}}$ for each pixel can be calculated. The image velocity vector can be written as a summation of translational $(\mathrm{u_T}, \mathrm{v_T})^{\mathrm{T}}$ and rotational $(\mathrm{u_R}, \mathrm{v_R})^{\mathrm{T}}$ components as stated in equation (36) [21]. x and y are the pixel coordinates of a feature point in a calibrated camera view. By using optical flow tracking the displacement of a feature point between two images $(\delta\mathrm{x}, \delta\mathrm{y})^{\mathrm{T}}$ can be calculated. Equation (37) shows how the image velocity vector $(\mathrm{u}, \mathrm{v})^{\mathrm{T}}$ and its rotational component $(\mathrm{u_R}, \mathrm{v_R})^{\mathrm{T}}$ can be determined. This is done by knowing a feature point coordinates (x, y) and the IMU angular velocity vector $\Omega = (\omega_{\mathrm{x}}, \omega_{\mathrm{y}}, \omega_{\mathrm{z}})$. The translational element $(\mathrm{u_T}, \mathrm{v_T})$ is then derived from equation (36). On the other hand $(\mathrm{u_T}, \mathrm{v_T})$ is related to the linear velocity, pixel coordinates and the feature point depth as per (39). This will result in an estimate for the depth value $(\mathrm{z_p})$ of a feature point at time $\mathrm{t} - 1$. In the same way depth value $(\mathrm{z_r})$ of the feature point at the time of the reference image is calculated. Using equation (18) the 3D coordinates of the corresponding 3D point in space with respect to the past and reference images are found. Equation (41) is then used to calculate the corrected translation vector at time $\mathrm{t} - 1$. This value is used to correct the past state of the system, which in turn updates the state estimation at time t.

(36) $(\boldsymbol{u}, \boldsymbol{v})^{T} = (\boldsymbol{u_T}, \boldsymbol{v_T})^{T} + (\boldsymbol{u_R}, \boldsymbol{v_R})^{T}$

(37) $(\boldsymbol{u}, \boldsymbol{v}) = \left(\frac{\delta x}{\delta t}, \frac{\delta y}{\delta t}\right)$,

(38) $(\boldsymbol{u_R}, \boldsymbol{v_R}) = (\boldsymbol{\omega_y} - \boldsymbol{\omega_z y} - \boldsymbol{\omega_x xy} + \boldsymbol{\omega_y x^2},\ -\boldsymbol{\omega_x} + \boldsymbol{\omega_z x} + \boldsymbol{\omega_y xy} - \boldsymbol{\omega_x y^2})$

(39) $(u_T, v_T) = \left(\frac{v_x - v_z x}{Z},\ \frac{v_y - v_z y}{Z}\right)$

(40) $Z = 0.5\left(\frac{v_x - v_z x}{u_T} + \frac{v_y - v_z y}{v_T}\right)$

(41) $X^{w} = RX^{c} + T$

INTELLIGENT TRACKING METHOD SELECTOR

The proposed hybrid tracking requires the vision system to provide reliable information. Therefore it is important to make sure the FoE exists and is of a good quality. In our system we considered this by introducing intelligent criteria to identify the existence of the FoE. In the cases where the FoE is not possible to determine due to physical constraints as described in section I.H and I.I, the system bypasses the vision system and continues with the IMU until the arrival of suitable features to formulate the FoE.

H. Qulaity of FoE

The FoE point is where all optical flow lines meet on the image plane. In order to accurately locate the FoE the flow lines must be accurately parameterized. A flow line is specified by two corresponding feature points. Any error in the location of the feature points leads to error in the estimation of the flow line parameters. A good FoE point is the one with a very short average distance to the flow lines. On the other hand a FoE point which has a significant average distance to the flow lines cannot be trusted. This analysis leads to the definition of the quality of FoE based on the average distance between the FoE point and the flow lines.

(42) $\boldsymbol{d} = \sqrt{\frac{1}{m} \sum_{m=1}^{N_f} (\boldsymbol{a_m x} + \boldsymbol{b_m y} + \boldsymbol{c_m})^2 / (\boldsymbol{a_m^2} + \boldsymbol{b_m^2})}$

I. Existance of FoE

When the camera motion between two consecutive images is parallel to the image plane, the flow lines are parallel and do not meet at a point on the image plane. In such circumstances a FoE point does not exist. This is determined by measuring the angle between the image plane's normal vector $(\mathrm{i_z^w})$ and the camera velocity vector (V) as per equation (43). $\mathrm{i_z^w}$ is obtained by rotating the unit vector in camera frame $(\mathrm{i_z^c})$ by $\mathrm{R^{wc}}$. When this angle is near $\frac{\pi}{2}$ the camera is moving approximately parallel to the image plane.

(43) $\boldsymbol{\theta} = \boldsymbol{V^T i_z^w} / \|\boldsymbol{V}\|$, $\quad \boldsymbol{i_z^w} = \boldsymbol{R^{wc} i_z^c} = \boldsymbol{R^{wc}} \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \end{pmatrix}$

J. Decision on the source of tracking

The system incorporates a decision making core, which decides on the source of tracking. When a FoE point does not exist (subsection I) or it has poor quality (subsection H), the hybrid tracking is bypassed and IMU only tracking is used. When the FoE has a very good quality meaning that the average distance as per equation (42) is very small, past state correction is applied prior to hybrid tracking using particle filtering method. In any other conditions the normal hybrid tracking method applies. This system implements an intelligent decision making process, which decides on the best source of tracking whether IMU only, hybrid only or hybrid with past state correction.

EXPERIMENTAL SET-UP AND RESULTS

In order to evaluate the performance of the proposed tracking method, a number of synthetic sequences for the camera motion have been simulated. Initially the camera motion was expressed in three directions using sinusoid waveforms as per equation (44), with various frequency, amplitude and phase values as detailed in Table 1. For the orientation a quaternion with a changing direction and angle has been considered (see Figure 7). The surrounding 3D space

is also filled with 507 feature points in three layers (see Figure 5).

To derive acceleration and angular velocity, the second and first derivatives of the ground truth position and orientation are calculated. For the measurement noise and error factors, the values listed in the InvenSense MPU-6000 datasheet are used. In the image-based system a random feature tracking error of up to 0.5pixel is added to represent the error margin of the feature detection and tracking algorithms.

Figures 5 to 7 illustrate the camera trajectory, 3D camera position and camera orientation over 500 image frames. The dotted lines represent the estimated values, while the solid lines indicate the ground truth. In Figure 5 the green, blue and red dots represent the feature points in 3D space, ground truth and estimated trajectory, respectively. Table 2 presents the mean error at two different sampling rates (50Hz and 25Hz) over the travelled distance of nearly 50m.

The algorithm was then applied on a more complex motion as described by equation (45) with parameters detailed in Table 3. Figure 8 to Figure 10 show the camera trajectory, angular velocity and acceleration.

$$(44)\quad \boldsymbol{path_i = a(1 - \cos(2\pi f t + \varphi))}$$

$$(45)\quad \boldsymbol{path_i = a + b(t - t_0)^n}\\ \boldsymbol{(1 + sin(2\pi sin(2\pi f_1 t)t) sin(2\pi f_2 t) cos(2\pi f_3 t) e^{-ct}}$$

TABLE 1 PARAMETRS FOR THE FIRST TRAJECTORY

i (axis)	a	f	φ
x	2m	0.5Hz	0
y	0.5m	1Hz	$\pi/3$
z	1m	0.25Hz	$-\pi/5$

TABLE 2 TRACKING ERROR

Trajectory	Mean error	Distance	Frequency	Error
1	2.66cm	48.54m	50Hz	0.054%
1	3.55cm	48.54m	50Hz	0.073%
1	5.44cm	48.54m	25Hz	0.11%
1	5.35cm	48.54m	25Hz	0.11%
2	11.68cm	137.27m	50Hz	0.085%
2	14.68cm	137.27m	50Hz	0.103%

TABLE 3 PARAMETRS FOR THE SECOND TRAJECTORY

i (axis)	a	b	c	n	t_0	f_1	f_2	f_3
x	1	1	-0.4	2	0	0.1	0.3	0.7
y	1	1	-0.5	2	0.5	0.05	0.4	0
z	0	1	-0.6	2	0	0.2	0.2	0.25

The results illustrate that despite tracking over a long distance of 48 meters in the first simulation and 137 meters in the second, the average tracking error remains in the region of few centimeters with the percentage error below 0.1% (see Table 2). Referring to the simulation results from [2] the EKF-SLAM and PTAM have achieved 0.75% and 0.03% error over 18.2 meters, respectively. These results have been gathered from a small and controlled workspace, on a simple trajectory, at a considerably slower motion with a dense feature point population (nearly 10 times the number of feature points used in this simulation). Nevertheless our proposed system outweighs the EKF-SLAM performance and is comparable with the PTAM based method, but with less computational cost and the ability to track in a wide area.

The system was tried several times at 50Hz video frame rate, with similar outcome (see Table 2). The system was also tried at 25Hz, where an increase in the error level was observed. This is due to the fact that the evaluation of the velocity particles depends on having a small time difference between two consecutive images. By reducing the sampling rate, sampling period is increased leading to less accurate results.

It must be noted that the system is based on a stochastic method; therefore minor performance changes from one run to another and occasional outliers are to be expected. It is also obvious that the performance degrades and error increases as the trajectory enters areas with less feature point density.

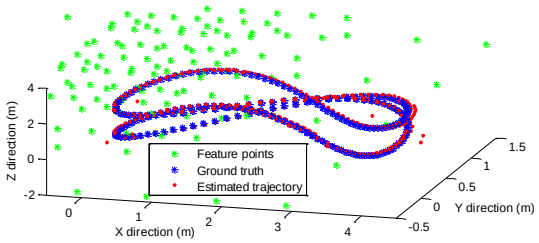


Figure 5 Camera Trajectory 1

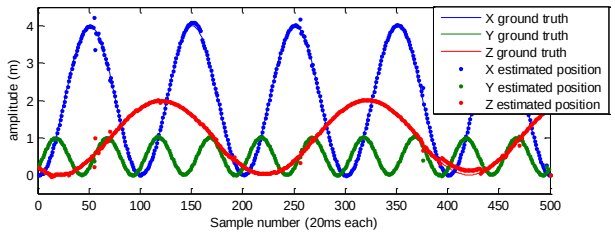


Figure 6 Camera 3D Position

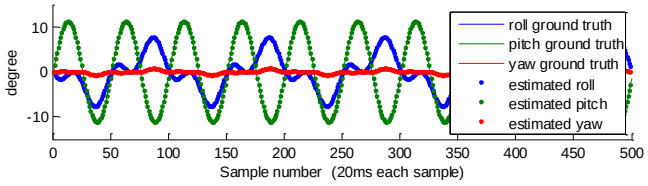


Figure 7 Camera Orientation

CONCLUSION AND FUTURE WORK

This paper proposes an algorithm for visual-inertial camera pose tracking, using adaptive recursive particle filtering. The paper presents an alternative approach to the EKF-SLAM and PTAM for wider area tracking. The technique benefits from a novel particle evaluation method based on the concept of focus of expansion and epipolar geometry constraints. This had

enabled an online automatic tracking method selection so that the robustness and accuracy of the tracking can be improved.

In order to enhance the tracking response time, the camera rotation matrix and consequently the camera orientation is separated from the successive state estimation process.

The performance of any particle filtering approach is very much dependent on the selection of the particles. The particle selection process in this work is mainly influenced by the IMU. As future work, it is suggested that the behavior of the system is constantly monitored in order to provide a more comprehensive insight into the particle selection. This can be achieved by the application of learning algorithms.

As another future work it is also suggested that an adaptive video sampling rate is considered to provide the best rate depending on the temporal speed.

The performance of the proposed system relies on the quality and existence of the focus of expansion. The formation of the focus of expansion can be influenced by the orientation of the camera. It is suggested that the camera is motorized and an intelligent agent is designed to take the sensory information and together with the decision making core (as described in section 0) effectively control the camera orientation.

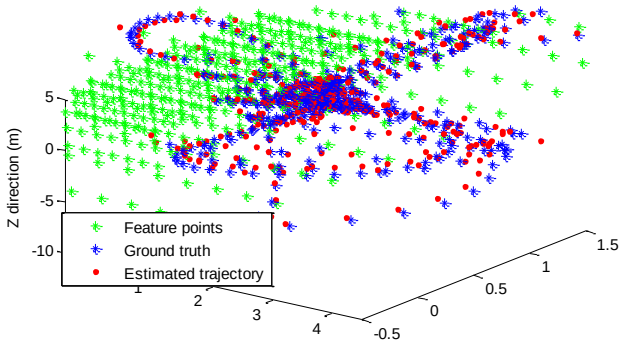


Figure 8 Camera Trajectory 2

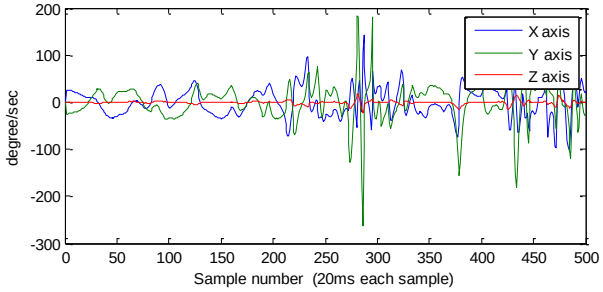


Figure 9 Angular Velocity

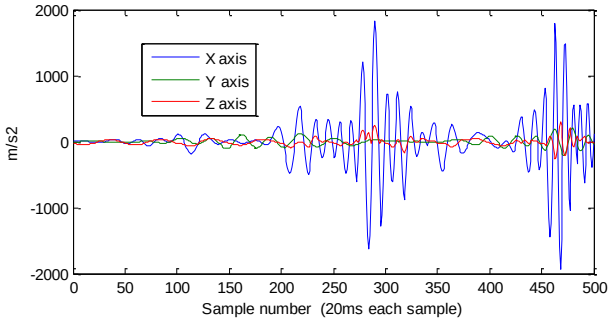


Figure 10 Acceleration

REFERNCES

[1] A. J. Davison, W. Mayol and D. Murray, "Real-time localization and mapping with wearable active vision," The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 18 - 27, 2003.

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

[3] J. Lobo and J. Dias, "Vision and inertial sensor cooperation using gravity as a vertical reference," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 12, p. 1597–1608, 2003.

[4] G. Bleser and D. Stricker, "Advanced tracking through efficient image processing and visual-inertial sensor fusion.," in IEEE Virtual Reality Conference (VR), Reno, Nevada, 2008.

[5] P. Corke, J. Lobo and J. Dias, "An Introduction to Inertial and Visual Sensing," The International Journal of Robotics Research, vol. 29, no. 2-3, pp. 231-244, 2010 .

[6] L. Mingyang and A. Mourikis, "Vision-aided inertial navigation for resource-constrained systems," 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) , pp. 1057 - 1063 , 2012.

[7] D. Scaramuzza, M. Achtelik, L. Doitsidis, F. Fraundorfer, E. Kosmatopoulos, B. Martinelli, V. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. Lee, S. Lynen, L. Meie, M. Pollefeys, A. Renzaglia and J. Siegwart, "Vision-Controlled Micro Flying Robots: from System Design to Autonomous Navigation and Mapping in GPS-denied Environments.," IEEE Robotics and Automation Magazine, 2014.

[8] M. S. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," IEEE Transactions on Signal Processing, vol. 50, no. 2, 2002

[9] B. Ristic, S. Arulampalam and N. Gordon, "Beyond the Kalman filter: particle filters for tracking applications," Boston, London, Artech House Publishers, 2004, pp. 48-49.

[10] X. Zhang, Y. Genc and N. Navab, "Mobile computing and industrial augmented reality for real-time data access," Technical report, Siemens Corporate Research, Imaging and Visualization Department, 755 College Road East, Princeton, NJ 08540, USA, 2001.

[11] H. Kato and M. Billinghurst, "Marker Tracking and HMD Calibration

for a Video-Based Augmented Reality Conferencing System," Proc. Second IEEE and ACM International Workshop Augmented Reality, vol. 85, pp. 20-21, 1999.

[12] M. Fiala, "Designing Highly Reliable Fiducial Markers," IEEE Transactions om Pattern Analysis and Machine Intelligence , vol. 32, no. 7, pp. 1317 - 1324, 2010.

[13] G. Desouza and A. Kak, "Vision for Mobile Robot Navigation: A Survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 237 - 267, 2002.

[14] E. Trucco and K. Plakas, "Video Tracking: A Concise Survey," IEEE Journal of Oceanic Engineering , vol. 31, no. 2, pp. 520 - 529 , 2006.

[15] R. Mautz and S. Tilch, "Survey of Optical Indoor Positioning Systems," International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1-7, 2011.

[16] H. Strasdat, J. M. M. Montiel and D. J., "Real-time monocular slam: Why filter?," IEEE International Conference on Robotics and Automation (ICRA), 2010.

[17] Y. Mia, S. Soatto and J. Kosecka, "“An Invitation to 3D Vision”," Springer-Verlag, 2004, pp. 211-214.

[18] R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision," Cambridge, Cambridge University Press, 2004, pp. 257-258.

[19] A. Moemeni and E. Tatham, "A framework for camera pose tracking using stochastic data fusion," In Games Innovations Conference (ICE-GIC), International IEEE Consumer Electronics Society's, 2010

[20] W. Burger and B. Bhanu, "Estimating 3-D Egomotion from Perspective Image Sequences," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 11, pp. 1040-1058, 1990.

[21] F. Raudies and H. Neumann, "A review and evaluation of methods estimating ego-motion," Computer Vision and Image Understanding, vol. 116, no. 5, pp. 606-633, 2012.

A.2

2010 2nd International IEEE Consumer Electronics Society's Games Innovations Conference

A Framework for Camera Pose Tracking Using Stochastic Data Fusion

Armaghan Moemeni
Eric Tatham
Department of Media Technology
Faculty of Technology
De Montfort University, Leicester, UK
Emails: armaghan@dmu.ac.uk ; etatham@dmu.ac.uk

ABSTRACT

A novel camera pose tracking system using a stochastic inertial-visual sensor fusion has been proposed. A method based on the Particle Filtering concept has been adapted for inertial and vision data fusion, which benefits from the agility of inertial-based tracking and robustness of vision-based camera tracking.

KEYWORDS - CAMERA POSE TRACKING, PARTICLE FILTER, INERTIAL-VISUAL SENSOR FUSION, AUGMENTED REALITY

1 INTRODUCTION

User tracking is an assisting technology for augmented reality (AR) and is also crucial for immersive virtual reality (VR). Its potential applications are numerous and include TV and film production, industrial maintenance, medicine, education, prototyping, entertainment and immersive games.

There are several aspects to consider when creating an augmented reality application. One of the most challenging is to precisely calculate the user's viewpoint in real-time so that the virtual elements are exactly registered with the real-world objects. The registration process usually requires an accurate six degrees of freedom (6DoF) tracking of the position and orientation of a head-mounted camera (HMC) generally referred to as camera pose tracking.

One of the major applications of camera position and orientation tracking in space is augmented reality games played outside over a wide area. Although video games have traditionally pulled players out of the real world and into a virtual one, augmented-reality games have the potential to engage people in the real world. For example, Novarama Technology [1] has developed a game called 'Invizimals' that makes it appear as if the world is populated by formerly invisible creatures that can interact with one another.

Several tracking and sensing methods have been researched or are commercially available. Miller [2] carried out a research survey to identify techniques and sensors that may be useful navigation and positioning methods for indoor applications. In this survey RFID, GPS and inertial and non-inertial sensors were studied to identify the optimal technique for position tracking. However, no existing methods fully satisfy the requirements for full 6DoF tracking over a wide area.

Placing fiducial markers in the real environment is currently the most common technique for recovering camera pose. However, the placement of artificial markers is not always convenient or possible especially for mobile AR applications. Therefore the desirability of developing a markerless wide-area camera tracking system is evident. In principle, besides using fiducial markers, camera pose tracking can be determined from naturally occurring features, such as points, lines, edges, or textures, and these strategies are referred to as vision-based tracking techniques. Zhang et al. [3] have made a fairly comprehensive study of the leading approaches to marker-based tracking, with the ARToolKit library [4] being a well-known and popular example.

Vision-based tracking techniques utilise image processing methods to calculate the camera pose relative to real world objects and so are analogous to closed-loop systems, which correct errors dynamically.

Purely vision-based tracking systems are known to have low jitter and no drift [5], however a drastic motion often leads to tracking failure due to latency caused by the high processing requirements and consequent low frame rate.

Inertial sensors consist of gyroscope and accelerometer devices for angular velocity and linear acceleration measurements. Despite their accuracy in fast motion tracking,

they suffer from accumulation errors over time, which can adversely affect registration stability.

Considering the limitations of existing tracking systems, in this paper we are proposing a hybrid inertial-visual pose estimation system which fuses both inertial sensor and vision systems data to provide the optimum accuracy for camera pose tracking and image registration. The aim of this work is to develop a novel framework for wide-area pose estimation of a head-mounted camera using a customised Particle Filter for sensor fusion.

In order to solve the problem of camera pose estimation and tracking, recursive filtering techniques such as Kalman Filter (KM), Extended Kalman Filter (EKF) and Particle Filter (PF) have been a continuing topic in the robotics and computer vision community [6],[7].

This paper addresses the question of how to use measurements from low-cost inertial sensors (gyroscopes and accelerometers) to compensate for the missing control information derived from a markerless vision-based system. For estimation and sensor fusion an adaptive recursive filtering technique is simulated.

2 PRINCIPLE OF OPERATION

The aim of the proposed system is to accurately determine the pose of the camera at any time. The camera pose is defined as follows:

$$X = (x, y, z, \theta_x, \theta_y, \theta_z)^T \quad (1)$$

x, y, z: 3D position with reference to initial camera position.
$\theta_x, \theta_y, \theta_z$: 3D orientation with reference to the initial camera orientation.

The proposed camera tracking system comprises of three main components, namely; IMU (Inertial Measurement Unit), vision-based tracker and particle filter. The data gathered from the IMU is fused with the data from the vision-based tracker using a novel fusion method based on particle filtering. This paper presents the details of the fusion algorithm and a simulated trajectory to demonstrate the performance of the system.

2.1 Vision-based Pose Estimation

Vision-based tracking techniques utilise image processing methods to calculate the camera pose relative to real world objects and thus are analogous to closed loop systems which correct errors dynamically.

After initially calculating camera pose from known visual features, the system dynamically obtains additional natural features and uses them to continuously update the pose calculation. The rationale underlying all feature-based methods is to find a correspondence between 2D image features and their 3D world frame coordinates. The camera pose can then be found by projecting the 3D coordinates of the feature into the observed 2D image coordinates and minimizing the distance to their corresponding 2D features.

There are prominent techniques proposed in the literature for solving the vision-based feature registration process. One approach uses the block matching method, see for instance Klein and Murray [8]. Another uses advanced techniques developed from the well-known Kanade-Lucas tracker (KLT) [9],[10].

In this work, the proposed vision-based tracking system will be based on optical flow measurements [11], in which after calibrating the camera using a flat chessboard pattern, the camera pose is calculated at every sampling interval. In each step the image features in the previous and current captured frames are identified using Shi and Tomasi's method [12]. The corresponding feature points in the two consecutive frames are then found using the Pyramid Lucas-Kanade method [9][10]. Finally, the rotation and transition matrices are determined using the corresponding feature points in the two consecutive frames, and accordingly, the 6DoF is estimated.

2.2 Inertial Measurement Unit

The processing latency of vision-based tracking systems can cause loss of real-time tracking capability. In addition, the tracked features can easily be lost due to occlusion or changing lighting conditions. In contrast, inertial sensor-based tracking techniques are proved to be more suitable for rapid and drastic movements/changes and offer attractive complementary features [4].

IMU sensors consist of gyroscope and accelerometer devices for angular velocity and linear acceleration measurements. Despite their accuracy in fast motion tracking, they suffer from accumulation errors over time that will adversely affect registration stability. In particular, the integration of accelerometer data is known to introduce instabilities when used for position estimation. This is due to the fact that accelerometers measure not only free acceleration but also acceleration due to gravity and centripetal forces, which have to be allowed for in estimating orientation. Whereas inertial-based methods are more suited for fast movements, they introduce considerable noise into the tracking system. On the other hand, vision-based tracking systems are robust and accurate with respect to slow movements and can, thus, be used to reset the accumulated tracking errors produced in inertial sensors.

The IMU provides the linear acceleration and angular velocity in a vector defined as follows:

$$I = (a_x, a_y, a_z, \omega_x, \omega_y, \omega_z) \quad (2)$$

a_x, a_y, a_z : *Linear acceleration*

$\omega_x, \omega_y, \omega_z$: *Angular velocity*

In an ideal situation the IMU pose can be calculated by twice integrating the Accelerometer data and once integrating the Gyroscope data. However in reality there are some issues with this approach which will introduce some error in the calculations:

- The data generated by IMU is noisy due to the electronic and ambient noise.
- The ADC resolution will also introduce some error to the output.
- The integration must be done in discrete time steps and therefore the acceleration and angular speed between the time steps are unknown, which will itself introduce additional error.
- Rotations are non-commutative, making order of rotation significant. When taking gyroscope readings at discrete time intervals order cannot be determined, thus introducing further potential error.
- Due to integration any noise introduced in each step will be accumulated, which will cause drift over time.

Although an IMU responds well to rapid changes in velocity, the above mentioned issues make the sole use of an IMU somewhat impractical. The IMU speed vector is defined as follows:

$$S^{IMU} = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z) \quad (3)$$

v_x, v_y, v_z : *Linear speed*

$\omega_x, \omega_y, \omega_z$: *Angular velocity*

The linear speed is calculated by integrating the linear acceleration and angular velocity is directly measured by the IMU.

In many augmented reality applications, a real-time video of the scene is available. Therefore, by combining the image data and IMU data, a more robust and accurate system can be made.

2.3 Inertial and Visual Sensor Fusion using Particle Filtering

In addition to application in wide-area gaming, integration of visual and inertial sensors opens new application directions in robotics, computer vision and numerous other fields [13],[14].

The Sequential Importance Sampling (SIS) algorithm is a Monte Carlo (MC) method that forms the basis for most sequential MC filters developed over the past decades. This sequential MC (SMC) approach is often referred to as bootstrap filtering, the condensation algorithm, particle filtering, interacting particle approximations, and survival of the fittest [15],[16].

Particle filtering can be used for estimating the internal states of a system when a series of observation data is available. The key idea is to represent the required posterior density function by a set of random samples with associated weights and to compute estimates based on these. It is a technique for implementing a recursive Bayesian filter by MC simulations. As the number of samples becomes very large, this MC characterization becomes an equivalent representation to the usual functional description of the posterior probability distribution function (PDF), and the SIS filter approaches the optimal Bayesian estimate.

In this work, the SIR method (Sampling Importance Resampling) is used for fusing IMU and vision-based data. To do so, particles are drawn from an importance function and then a weight is assigned to each of them using a likelihood function. The weights are then normalised and, if the number of effective particles is less than a threshold, the re-sampling algorithm is utilised to remove particles with small weights. The SIR algorithm is summarised as follows [15][16].

2.3.1 *Sampling*

I. FOR $n = 1{:}N$
   - Draw particles from a probability distribution function using Inverse Transform Sampling: $X_k^i \sim p(X_k \mid X_{k-1}^i)$
   - Calculate weights using likelihood function: $\widehat{w}_k^i = p(Z_k \mid X_k^i)$

II. Calculate total weight: $\Sigma\widehat{w} = \sum_{i=1}^{N} \widehat{w}_k^i$

III. FOR $i = 1{:}N$

Normalise weights: $w_k^i = \widehat{w}_k^i / \Sigma\widehat{w}$

IV. Calculate number of effective particles:

$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} (w_k^i)^2}$$

V. If $\widehat{N}_{eff} < N_{th}$ then do Re-Sampling

2.3.2 *Re-Sampling*

I. Initialise the *CSW* (Cumulative Sum of Weights): $c_1 = w_k^i$

II. FOR $i = 2{:}N$

Construct *CSW*: $c_i = c_{i-1} + w_k^i$

III. Start at the bottom of the *CSW*: $i = 1$

IV. Draw a starting point: $u_1 \sim U[0, \frac{1}{N}]$

V. FOR $j = 1{:}N$

- $u_j = u_1 + \frac{j-1}{N}$
- While $u_j > c_i$ then $i = i+1$
- Assign sample: $\hat{X}_k^j = X_k^i$
- Assign weight: $w_k^j = \frac{1}{N}$

The variables used in the algorithm are defined as follows:

N: The number of particles
i: Particle index
k: Time index
X_k^i *: i^{th} particle at time* t_k
X_k *: System state at time* t_k

2.4 Data Fusion

The proposed camera pose tracking system combines IMU data with additional information from a vision-based system. Taking into account the inherent latency of the vision-based system (due to use of standard video frame rates and the volume of computation required in tracking optical flow), the IMU and vision-based systems are sampled at two different sampling rates. The IMU must be sampled at a faster rate in order to minimise integration error. An integer ratio between the two sampling rates is selected to maintain synchronisation.

Our proposed fusion method benefits from the principles of particle filtering described earlier in section 2.3. To do so, a probability distribution function in the form of a Gaussian distribution is defined, from which particles are drawn. The PDF is defined as follows:

$$f_k^l = N(x, \mu_k^l, \sigma_k^l) \qquad (4)$$
$$\mu_k^l = X_{k-1},\ \sigma_k^l = S_{Max}^{IMU} \Delta t$$

where l, μ_k^l, σ_k^l, Δt, S_{Max}^{IMU} *are:*

l: The l^{th} element of the pose vector
μ_k^l *: Mean value*
σ_k^l *: Variance*
Δt *: Sampling time interval*
S_{Max}^{IMU} *: A vector consisting of the maximum of each individual element in the IMU speed vector.*

At each time step t_k the importance function is determined and *N* particles are drawn from this function. At this stage particles are divided into two parts. The first three elements form a vector representing the 3D camera position and the remaining three elements form the 3D camera orientation vector. Both position and orientation vectors are then compared against the respective vectors of the pose vector determined from the vision-based tracking system and, following that, two weights are assigned to each particle as follows:

$$\hat{w}_k^{i,POS} = 1/\sqrt{\sum_{i=1}^{3}(X_{k,l}^C - X_{k,l}^i)^2} \qquad (5)$$

$$\hat{w}_k^{i,ORN} = 1/\sqrt{\sum_{i=4}^{6}(X_{k,l}^C - X_{k,l}^i)^2} \qquad (6)$$

$\hat{w}_k^{i,POS}, \hat{w}_k^{i,ORN}$ *: Weights for position and orientation vectors of each particle, respectively.*
$X_{k,l}^C$ *: l^{th} element of the vision-based pose*

If the effective number of particles is less than a threshold level, the re-sampling algorithm is used to draw a new set of particles.

3 SIMULATION

The primary use of the proposed system is in head-mounted camera pose tracking. As per gait analysis literature, the walking process of a person is very close to a periodic movement, which can be estimated by a series of sine and cosine functions [17]. Therefore to evaluate the performance of the tracking system, the IMU outputs (acceleration, angular velocity) have been simulated using sine functions. This will be particularly beneficial as the frequency response of the proposed system can also be assessed. Gaussian noise is then added to the sine functions to represent noise as per the IMU datasheet [18]. The IMU outputs 10-bit acceleration and angular velocity data in three directions. These are simulated by combination of sine functions and noise as follows:

$$a_l = A_l^a \sin(2\pi f_l^a t) + \nu_l^a,\ l = 1,2,3 \qquad (7)$$
$$\omega_l = A_l^\omega \sin(2\pi f_l^\omega t) + \nu_l^\omega,\ l = 4,5,6 \qquad (8)$$

where a_l, ω_l, f_l^a, f_l^ω, ν_l^a, ν_l^ω are :

a_l *: Linear acceleration*
ω_l *: Angular velocity*
f_l^a, f_l^ω *: Simulation frequency*
ν_l^a, ν_l^ω *: Noise*

Each particle consists of 6 elements. The first three elements form the position vector and the remaining three the orientation vector. The position vector in the reference IMU pose is calculated by twice integrating noise-free acceleration data and the orientation vector by once integrating the noise-free angular velocity.

Similarly to reference IMU pose, the position vector in the real IMU pose is calculated by twice integrating noisy acceleration data and the orientation vector by once integrating the noisy angular velocity. Sampling rate of 100Hz was chosen for IMU data sampling.

The vision-based system has also been simulated by adding noise to the reference IMU pose. Sampling rate of 10Hz was chosen for the vision-based system. This will be 10 times slower than the IMU sampling time. This is due to the fact the vision-based analysis requires more processing power and tends to run at lower sampling rates.

In this work, elements of position and orientation vectors have been considered independent of each other. Therefore to show the performance of the algorithm, the position and orientation in the X direction are discussed in this section.

Figures 1 and 2 show position/orientation trajectories in X direction. Figure 3 shows the error for position in X direction. The reference in calculating the error is the reference IMU pose.

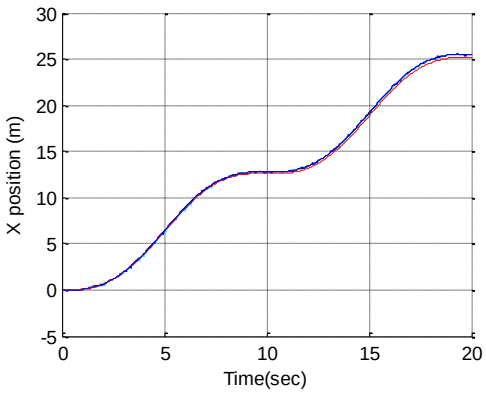
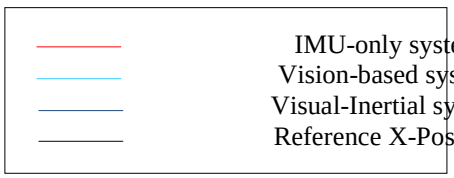


Fig.1a : Camera X-position trajectory



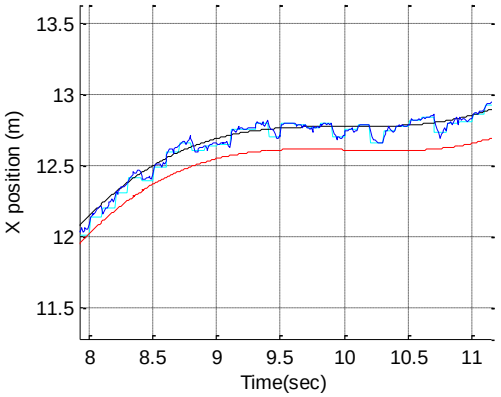


Fig.1b : 3-sec. snapshot of Fig.1a

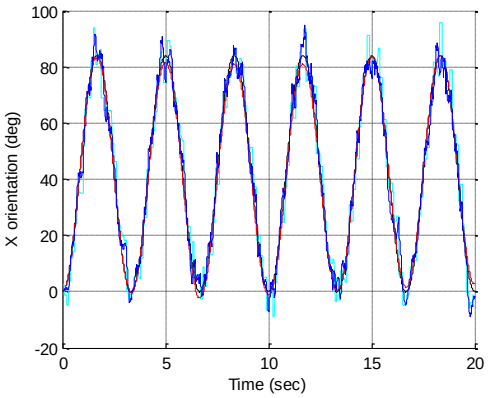


Fig.2a Camera X-orientation trajectory

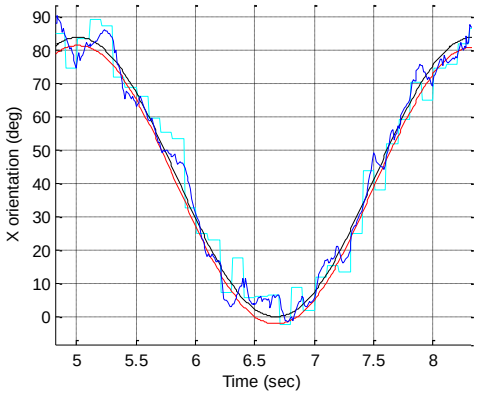


Fig.2b : 3-sec. snapshot of Fig.2a

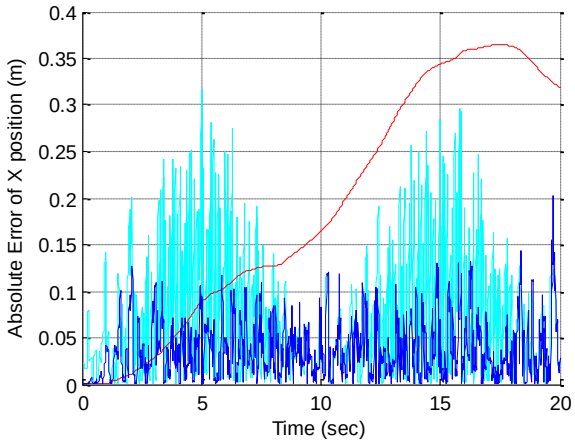


Fig.3 X-Position Absolute Error

Fig.3 shows that, initially, simple integration of IMU acceleration gives less error and more accurate results. However as the time goes by, due to the accumulation error, the integration error increases and gradually exceeds the pose error resulting from either the vision-based system or the inertial-visual fusion system. The mean values for different types of error are as follows:

X-Position	Mean Value of the Error
IMU	0. 1907
Inertial-Visual (Particle Filter)	0. 0369
Vision Based	0. 0732

Table 1: Position Error

X-Orientation	Mean Value of the Error
IMU	1.2755
Inertial-Visual (Particle Filter)	3.1533
Vision Based	4.7909

Table 2: Orientation Error

The mean values for position error show that, in general, the Particle Filtering method produces better results than vision-based. The Particle Filtering method also has the advantage of being adaptable to sudden movements of the camera, as IMU data is sampled at 100Hz, whereas the vision-based system works at much lower sampling rate of 10Hz. With respect to orientation tracking, although Particle Filtering still performs better than the pure vision-based system, IMU on its own generates the best results.

4 CONCLUSION

This paper presents a Particle Filtering-based data fusion method to combine pose data from an IMU and a vision-based pose tracking system. The simulation shows that after initial start-up time, the inertial-visual tracking system produces better results for position tracking compared with either the IMU or vision-based tracking systems alone. However for orientation tracking, although the Particle Filtering-based method performs better than the vision-based system, it is less accurate than the IMU system.

As future work, methods such as Direction Cosine Matrix (DCM) [19] will be employed to refine the output of the IMU for camera orientation tracking, particularly in relation to dealing with centripetal forces and correction of errors due to finite sampling. Also the frequency response of the system will be evaluated in order to better understand the limitations of this method.

5 REFRENCES

[1] Novarma Technology, PSP - Invizimals Shadow Zone; http://www.novarama.com/

[2] L. E. Miller, “Indoor Navigation for First Responders: A Feasibility Study”
http://www.antd.nist.gov/wctg/RFID/RFIDassist.htm, February 2006

[3] X. Zhang, S. Fronz and N. Navab. “Visual marker detection and decoding in AR systems: a comparative study” In ISMAR ’02, pp. 97-106, 2002

[4] ArToolKit - http://www.hitl.washington.edu/artoolkit

[5] P. Corke, J. Lobo, J. Dias, “An Introduction to Inertial and Visual Sensing” , In International Journal of Robotics Research, Volume 26 , Issue 6, pp 519-535, June 2007

[6] M. Isard, A. Blake. “CONDENSATION - conditional density propagation for visual tracking”, In International Journal of Computer Vision, 29:5–28, 1998

[7] M. Pupilli and A. Calway. “Real-time Camera Tracking Using a Particle Filter.” In British Machine Vision Conference (BMVC), pp: 519–528, Oxford, England, September 2005

[8] G. Klein and D. Murray. “ Parallel Tracking and Mapping for Small AR Workspaces.”
In International Symposium on Mixed and Augmented Reality (ISMAR), Nara, Japan, November 2007

[9] B. Lucas and T. Kanade. “An Iterative Image Registration Technique with an Application
to Stereo Vision“, In International Joint Conference on Artificial Intelligence (IJCAI),pages 674–679, 1981

[10] L.Y.Siong; S.S.Mokri,.; A.Hussain; N. Ibrahim;
M.M. Mustafa; ”Motion detection using Lucas Kanade algorithm and application enhancement” In 2009 International Conference on Electrical Engineering and Informatics pp. 537 – 542, August 2009, Malaysia

[11] F. Kendoul, I. Fantoni, and G. Dherbomez. Three Nested Kalman Filters-Based Algorithm for Real-Time Estimation of Optical Flow, UAV Motion and Obstacles Detection. In IEEE International Conference on Robotics and Automation (ICRA), 2007

[12] J. Shi, C.Tomasi , Good Features to Track
In Computer Vision and Pattern Recognition, Proceedings CVPR '94, 1994 IEEE Computer Society Conference-
pp.593 – 600

[13] J. Lobo and J. Dias, “Vision and inertial sensor cooperation using gravity as a vertical reference,” In IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, no. 12, pp. 1597–1608, Dec. 2003

[14] P. Corke, J. Dias, M. Vincze, and J. Lobo, “Integration of vision and inertial sensors,” presented at the IEEE Int. Conf. Robotics Automation (ICRA), Barcelona, Spain, Apr. 2004

[15] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp,” A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking”, In IEEE Transactions on Signal Processing, Vol. 50, No. 2, February 2002

[16] B. Ristic, S. Arulampalam, N. Gordon,
“Beyond the Kalman filter: particle filters for tracking applications”, Artech House Publishers, Boston, London - January 2004

[17] W. Kusakunniran, Q. Wu, H. Li, and J. Zhang, “ Multiple Views Gait Recognition using View Transformation Model Based on Optimized Gait Energy Image”, In IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, 2009

[18] Sparkfun Electronics, “IMU 6 Degrees of Freedom v4 Data Sheet”, 2008

[19] W. Premerlani and P. Bizard ,”Direction Cosine Matrix IMU: Theory”, 2009

Appendix B : Geodetic – ECEF – ENU Coordinate Conversions

```
//  GeodeticCalculator.m

#import "GeodeticCalculator.h"

#define DEGREES_TO_RADIANS (M_PI/180.0)

#define WGS84_A (6378137.0)             // WGS 84 semi-major axis constant in
meters
#define WGS84_F (1/298.257223563)    // recipricol flattening

#define WGS84_E (8.1819190842622e-2)    // WGS 84 eccentricity

//Convert ECEF to Latitude and Longitude
#define a   6378137              /*Semimajor axis*/
#define f   0.003352810664747     /*Flattening*/
#define e2  0.006694379990141     /*Square of first eccentricity*/
#define ep2 0.006739496742276     /*Square of second eccentricity*/
#define b   6356752.314245179     /*Semiminor axis*/
#define radsToDegs  57.295779513082323


@implementation GeodeticCalculator



// Converts latitude, longitude to ECEF coordinate system

void geodeticToEcef(double lat, double lon, double alt, double *x, double *y, double
*z)
{
    double clat = cos(lat * DEGREES_TO_RADIANS);
    double slat = sin(lat * DEGREES_TO_RADIANS);
    double clon = cos(lon * DEGREES_TO_RADIANS);
    double slon = sin(lon * DEGREES_TO_RADIANS);

    double N = WGS84_A / sqrt(1.0 - WGS84_E * WGS84_E * slat * slat);

    *x = (N + alt) * clat * clon;
    *y = (N + alt) * clat * slon;
    *z = (N * (1.0 - WGS84_E * WGS84_E) + alt) * slat;
}



// Coverts ECEF to ENU coordinates centered at given lat, lon
void ecefToEnu(double lat, double lon, double x, double y, double z, double xr,
double yr, double zr, double *e, double *n, double *u)
{
    double clat = cos(lat * DEGREES_TO_RADIANS);
```

```
    double slat = sin(lat * DEGREES_TO_RADIANS);
    double clon = cos(lon * DEGREES_TO_RADIANS);
    double slon = sin(lon * DEGREES_TO_RADIANS);
    double dx = x - xr;
    double dy = y - yr;
    double dz = z - zr;

    *e = -slon*dx  + clon*dy;
    *n = -slat*clon*dx - slat*slon*dy + clat*dz;
    *u = clat*clon*dx + clat*slon*dy + slat*dz;
}


//Convert ECEF to Lat, Long and Alt

void ecefToGeodetic ( double x, double y, double z, double *lat, double *lon, double
*alt)
{


    double lambda;
    double rho;
    double beta;
    double sbeta;
    double cbeta;
    double phi;
    double sphi;
    double betaNew;
    int count;
    double N;
    double h;
    double cphi;



        /* Longitude*/
        lambda = atan2(y, x);

        /* Distance from Z-axis*/
        rho = sqrt(x*x+y*y);

        /* Bowring's formula for initial parametric (beta) and geodetic (phi) latitudes*/
        beta = atan2(z, (1 - f) * rho);
        sbeta = sin(beta);
        cbeta = cos(beta);

        phi = atan2(z+b*ep2*sbeta*sbeta*sbeta, rho-a*e2*cbeta*cbeta*cbeta);
        sphi = sin(phi);
        cphi = cos(phi);

        /* Fixed-point iteration with Bowring's formula*/
        /* (typically converges within two or three iterations)*/
        betaNew = atan2((1 - f)*sin(phi), cos(phi));
        count = 0;
        while ((beta!=betaNew) && count < 5){
```

```
            beta = betaNew;
            sbeta = sin(beta); cbeta = cos(beta);
            phi = atan2(z+b*ep2*sbeta*sbeta*sbeta, rho-a*e2*cbeta*cbeta*cbeta);
            sphi = sin(phi); cphi = cos(phi);
            betaNew = atan2((1 - f)*sphi, cphi);
            count++;
        }

        /* Calculate ellipsoidal height from the final value for latitude*/
        N = a / sqrt(1 - e2 * sphi* sphi);
        h = rho * cphi + (z + e2 * N* sphi) * sphi - N;

       *lat = radsToDegs*phi;
       *lon = radsToDegs*lambda;
       *alt =h;

}

//Convert ENU to ECEF coordinates

void enuToEcef( double refLat, double refLon, double refAlt, double e, double n,
double u, double *x, double *y, double *z)
{
    //find reference location in ECEF coordinates

    double xr, yr, zr;

    geodeticToEcef(refLat, refLon, refAlt, &xr, &yr, &zr);

    *x = (-e * sin(refLon * DEGREES_TO_RADIANS)) - (n * sin(refLat *
DEGREES_TO_RADIANS) * cos(refLon * DEGREES_TO_RADIANS)) + (u *
cos(refLat * DEGREES_TO_RADIANS)* cos(refLon * DEGREES_TO_RADIANS)) +
xr;
    *y = (e * cos(refLon * DEGREES_TO_RADIANS)) - (n * sin(refLat *
DEGREES_TO_RADIANS) * sin(refLon * DEGREES_TO_RADIANS)) + (u *
cos(refLat * DEGREES_TO_RADIANS) * sin(refLon * DEGREES_TO_RADIANS))
+yr;
    *z = (n * cos(refLat * DEGREES_TO_RADIANS)) + (u * sin(refLat *
DEGREES_TO_RADIANS)) + zr;

}

@end
```

Appendix C : Trilateration Calculation Results

Actual receiver tile location coordinates		Calculated receiver tile location using trilateration		Distance of calculated location from actual location in tile units	Distance of calculated location from actual location in metres
x	y	x	y		
0	0	Location of iBeacon			
1	0	3	2.6	3.28	1.64
2	0	3.6	1.4	2.13	1.07
3	0	4	-8.5	8.56	4.28
4	0	4.3	-16.9	16.9	8.45
5	0	4.7	-0.8	0.85	0.43
6	0	4.6	-2.2	2.61	1.31
7	0	4.6	0.5	2.45	1.23
8	0	4.6	-6.4	7.25	3.63
9	0	Location of iBeacon			
0	1	2.1	4.2	3.83	1.92
1	1	-0.1	6.7	5.81	2.91
2	1	2.8	4.2	3.30	1.65
3	1	3.5	2	1.12	0.56
4	1	6	1.1	2.00	1.00
5	1	3.6	-1.3	2.69	1.35
6	1	5.2	0.3	1.06	0.53
7	1	4.7	1.8	2.44	1.22
8	1	4.1	2.7	4.25	2.13
9	1	5	1.8	4.08	2.04
0	2	-1.7	8.6	6.82	3.41
1	2	0.3	7.3	5.35	2.68
2	2	4.5	4.2	3.33	1.67
3	2	4.5	2.6	1.62	0.81
4	2	3.5	2	0.50	0.25
5	2	3.5	4.3	2.75	1.38
6	2	5.1	-1.5	3.61	1.81
7	2	5.7	-2.4	4.59	2.30
8	2	4.2	-2.8	6.12	3.06
9	2	7.4	3.3	2.06	1.03
0	3	3.6	4	3.74	1.87
1	3	2.9	2.1	2.10	1.05
2	3	2.7	7.7	4.75	2.38
3	3	1.8	5.6	2.86	1.43
4	3	3.4	0.5	2.57	1.29
5	3	3.6	2.5	1.49	0.75
6	3	3.1	2.1	3.04	1.52
7	3	3.1	3.7	3.96	1.98
8	3	1.8	6.2	6.98	3.49
9	3	7.4	1.2	2.41	1.21
0	4	-3	10.2	6.89	3.45
1	4	2.4	1.4	2.95	1.48
2	4	2.6	6.2	2.28	1.14
3	4	3	-1.9	5.90	2.95
4	4	0.3	3.9	3.70	1.85
5	4	2.6	-6.4	10.67	5.34
6	4	3.6	0	4.66	2.33
7	4	6	-6.2	10.25	5.13
8	4	4.2	4.9	3.91	1.96
9	4	5	3.5	4.03	2.02

0	5	0	8.4	3.40	1.70
1	5	2.8	4.7	1.82	0.91
2	5	1.9	5.4	0.41	0.21
3	5	0.2	8.3	4.33	2.17
4	5	-10.2	18.4	19.52	9.76
5	5	5.1	3.7	1.30	0.65
6	5	9.5	3	4.03	2.02
7	5	1.8	4.9	5.20	2.60
8	5	5.1	2.7	3.70	1.85
9	5	4.5	-5.6	11.52	5.76
0	6	0.9	7	1.35	0.68
1	6	1.7	6.2	0.73	0.37
2	6	-0.2	9	3.72	1.86
3	6	-0.8	9.7	5.30	2.65
4	6	2.9	5.9	1.10	0.55
5	6	5.3	7	1.04	0.52
6	6	5.1	5.3	1.14	0.57
7	6	7	3.5	2.50	1.25
8	6	5.7	3.5	3.40	1.70
9	6	6.4	3.7	3.47	1.74
0	7	3	4.7	3.78	1.89
1	7	2.9	4.3	3.30	1.65
2	7	4	3.5	4.03	2.02
3	7	2	8.1	1.49	0.75
4	7	3.1	3.7	3.42	1.71
5	7	2.9	6.3	2.21	1.11
6	7	12.8	5.6	6.94	3.47
7	7	3.9	6.2	3.20	1.60
8	7	6.4	2.2	5.06	2.53
9	7	3.9	7.5	5.12	2.56
0	8	-4.2	13	6.53	3.27
1	8	3.9	3.3	5.52	2.76
2	8	3.6	6.3	2.33	1.17
3	8	-4.6	13	9.10	4.55
4	8	-2.4	11.5	7.29	3.65
5	8	4.5	6.3	1.77	0.89
6	8	8.8	5.4	3.82	1.91
7	8	14.5	6.4	7.67	3.84
8	8	8.8	2	6.05	3.03
9	8	6.4	4	4.77	2.39
0	9	0.5	7.7	1.39	0.70
1	9	-2.6	8.9	3.6	1.8
2	9	-19.2	27.2	27.94	13.97
3	9	0.5	6.7	3.40	1.70
4	9	1.1	8.4	2.96	1.48
5	9	3.8	4.7	4.46	2.23
6	9	4.5	6.1	3.26	1.63
7	9	5.2	4.7	4.66	2.33
8	9	5.3	7.5	3.09	1.55
9	9	Location of iBeacon			