Long-Range Outdoor Odometry using a Red-Green-Blue-Depth Camera and an Inertial and Magnetic Measurement Unit

A Thesis

submitted in fulfilment of the requirements for the degree of
Bachelor of Engineering, Mechatronics (Space)

THE UNIVERSITY OF SYDNEY

School of Aeronautical, Mechanical and Mechatronic Engineering

The University of Sydney

Author:
Alexander Brett Bunting
SID: 311190162

Supervisor:
Dr Ali Haydar Göktoğan
Australian Centre for Field Robotics
The University of Sydney
NSW 2006 Australia

October 26, 2014

Declaration

I hereby declare this project is my own work and that, to the best of my knowledge, it contains no material from previously published sources, except where due acknowledgement is made in the text. This project is submitted to the School of Aerospace, Mechatronic and Mechanical Engineering at the University of Sydney in fulfilment of the requirements of a Bachelor of Engineering: Mechatronics (Space). The tasks I completed are listed below.

- Carried out a literature review of existing odometry techniques
- Designed a software system for odometry using a Kinect Red-Green-Blue-Depth camera capable of switching between visual and depth-based odometry methods appropriate to terrain
- Presented an Extended Kalman Filter for fusing the odometry motion estimate with orientation computed using an Inertial and Magnetic Measurement Unit
- Designed and built a protective structure to shield the Kinect from infrared light so that it can be used outdoors
- Performed calibration procedures to make effective use of hardware components
- Implemented odometry processing steps in parallel on a Graphics Processing Unit to reduce processing time
- Evaluated the system positioning performance over hundreds of metres and in a variety of outdoor terrain environments
- Submitted a conference paper to the International Conference on Robotics and Automation describing my experimental method, hardware configuration and results

The above represents an accurate summary of the student's contribution.

Signed _____ _____

Alexander Brett Bunting (student) Dr Ali Haydar Göktoğan (supervisor)

Acknowledgement

I would like to thank my supervisor Dr Ali Haydar Göktoğan for his support over the last year, for being a bountiful technical resource and for always providing detailed and useful feedback. In particular I'd like to recognise his assitance and encouragement for my submission of a conference paper to the International Conference on Robotics and Automation.

I would also like to thank my father, Greg, for partaking in innumerable trips to Bunnings for more parts or tools and for providing a virtual practical design wall to bounce ideas off. To my mother, Carolyn, for sitting through several iterations of my seminar presentation and for suggesting a method for calibrating the magnetometer compass using her sailing knowledge. Finally, thanks to my girlfriend, Emma, for her patience and support through some trying but rewarding times.

Abstract

Localisation is an ongoing challenge in the field of mobile robotics, however accurate GPS units may be prohibitively expensive while other external reference infrastructure may be unavailable in unknown or unstructured terrain. This paper presents a system in which Red-Green-Blue-Depth (RGB-D) odometry is fused with Inertial and Magnetic Measurement Unit (IMMU) orientation data for long range, outdoor motion estimation. Experimental results are presented which demonstrate the effectiveness of the system over a variety of outdoor terrain environments over distances of hundreds of meters with drifts of between 8-25mm/sec. The data is captured using a low-cost IMMU, and a Kinect for Windows v2 RGB-D camera physically shielded against interference from sunlight.

Contents

List of Figures

List of Tables

Acronym List

AGAST	Adaptive and Generic Accelerated Segment Test
AHRS	Attitude-Heading Reference System
BRIEF	Binary Robust Independent Elementary Features
CCD	Charge Coupled Device
CenSurE	Centre Surround Extrema
CMOS	Ceramic Metal Oxide Semiconductor
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DGPS	Differential Global Positioning System
DOF	Degree of Freedom
DoG	Difference of Gaussian
DVO	Dense Visual Odometry
EKF	Extended Kalman Filter
FAST	Features from Accelerated Segment Test
FOV	Field of View
fps	frames per second
GPS	Global Positioning System
GPU	Graphics Processing Unit
ICP	Iterative Closest Point
IMMU	Inertial-Magnetic Measurement Unit
IMU	Inertial Measurement Unit
IR	Infrared
LoG	Laplacian of Gaussian
NED	North-East-Down
NMEA	National Marine Electronics Association
ORB	Oriented FAST and Rotated BRIEF
PCL	Point Cloud Library

RANSAC	Random Sample Consensus
RGB-D	Red-Green-Blue-Depth
RMSE	Root Mean Square Error
RPY	Roll Pitch Yaw
SBA	Sparse Bundle Adjustment
SFM	Structure From Motion
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localisation and Mapping
SSE	Streaming SIMD Extensions
SURF	Speeded Up Robust Features
SVD	Singlar Value Decomposition
TOF	Time of Flight
TORO	Tree-based netwORk Optimizer
VO	Visual Odometry

Nomenclature

$(c_{x,d}, c_{y,d})$	Depth camera centre pixel coordinates in the x and y directions
$(f_{x,d}, f_{y,d})$	Depth camera focal lengths in the x and y directions
(u, v)	Pixel in an image at column u and row v measured from the top left
(u_d, v_d)	Pixel coordinates in the depth image
(u_k, v_k)	Pixel coordinates of a keypoint
(u_p, v_p)	Pixel coordinates of a predicted keypoint location under a camera motion hypothesis
$(u_{k,curr}, v_{k,curr})$	The measured pixel coordinates in the current frame of a keypoint matched to keypoint k in the previous frame
(x_c, y_c, z_c)	Coordinates of a 3D point in the colour camera frame
(x_d, y_d, z_d)	Coordinates of a 3D point in the depth camera frame
(x_k, y_k, z_k)	Coordinates of a 3D keypoint location
(x_p, y_p, z_p)	Coordinates of a predicted 3D keypoint location under a transformation
D_c	A depth image in the frame of the colour camera
D_d	A depth image in the frame of the depth camera
R_c	A range reading measured in the colour camera frame (mm)
R_d	A range reading measured in the depth camera frame (mm)
R_k	A range reading of a keypoint (mm)
T_{ICP}	A transform between frames estimated by ICP on the dense point clouds produced by projecting the depth image
T_{RANSAC}	A transform between frames estimated during one iteration of RANSAC
T_{RGBD}	A transform between frames estimated by aligning 3D keypoints found in the colour image
T_{SBA}	A transform between frames estimaged by the SBA library for bundle adjustment
T_{VO}	The final odometry estimate of the transform between frames, selected as the more suitable of T_{SBA} or T_{ICP} depending on terrain
ϵ_{ICP}	Error metric for ICP between tangent planes in one point cloud and matched points in the other cloud

ϵ_{RANSAC}	Error metric for RANSAC between a predicted keypoint location and a measured keypoint location under a particular camera motion hypothesis
ϵ_{SBA}	Error metric for SBA between predicted and measured feature point locations under a given set of 3D feature locations and camera positions
ϵ_{align}	Error metric for visual keypoint cloud alignment
ϵ_{select}	Error metric for a planar fit to the reduced image to determine whether ground is flat for selecting the odometry type
$\mathbf{A}$	Jacobian of the dynamic model which is used in the Kalman Filter prediction equation
$\mathbf{H}$	Kalman Filter measurement prediction matrix
$\mathbf{K}_k$	Kalman Filter gain at time k
$\mathbf{Q}$	Kalman Filter process noise covariance matrix
$\mathbf{R}$	Kalman Filter measurement noise covariance matrix
$\hat{\mathbf{x}}_k$	Kalman Filter state vector estimate at time k
${}^{C}q_{IMMU}$	The quaternion orientation between the IMMU body frame and the Kinect colour camera
${}^{D}T_{C}$	Homogenous transform representing the pose of the colour camera frame with respect to the depth camera frame
kp_{curr}	Set of 3D keypoint locations found in the current frame
kp_{prev}	Set of 3D keypoint locations found in the previous frame
q_{AHRS}	The quaternion orientation output from the Attitude Heading Reference System
$w_{i,j}$	Weighting for depth interpolation applied at pixel at row i and column j relative to the keypoint location

Chapter 1

Introduction

The objective of this thesis is to develop and investigate the capabilities of a Red-Green-Blue-Depth (RGB-D) odometry system operating in a variety of unstructured environments over long distances. RGB-D cameras simultaneously output colour and depth information. An Inertial-Magnetic Measurement Unit (IMMU) is also used which outputs acceleration, magnetic and rotational velocity readings, used to measure orientation. The camera odometry is fused with the IMMU orientation using an Extended Kalman Filter (EKF) to estimate the camera position and orientation.

Such a system using RGB-D cameras has not previously been tested in outdoor or unstructured terrain, nor over long distances on the order of hundreds of metres. This thesis will describe the operation of the odometry system including both visual and depth odometry modalities and a switching strategy between them depending on terrain. A low-cost IMMU is used with the Microsoft Kinect for Windows version 2 RGB-D camera, including a hardware setup to make it useable outdoors. The results of offline system testing are presented for a number of different terrain environments over distances greater than 500m.

1.1 Motivation

Localisation is an important part of the guidance, navigation and control of autonomous mobile robots. The Global Positioning System (GPS) is a common option for performing localisation. However it is not always available, such as when underground, underwater or on the surface of other planets. The quality of a GPS signal may also be degraded indoors or in natural or urban canyons with limited visibility to the sky. Most significantly however, the GPS position signal is usually quite noisy in low-cost units, which may be unsuitable for positioning a mobile robot. While more accurate units exist, such as Differential Global Positioning System (DGPS) units, these can be prohibitively expensive.

Another common technique for localisation is to use external infrastructure as a reference, such as floor guidelines, laser reflectors or sonar beacons [1]. However this in-

frastructure is not applicable when operating in new or unknown environments, in which case localisation must be performed using only data which can be captured with on-board sensors. This is called odometry.

Wheel odometry is one solution which estimates distance travelled by counting wheel revolutions. However errors from wheel odometry can accumulate quickly due to wheel slippage, variation in tyre size over time, or different wheel turn rates over rough terrain. An Inertial Measurement Unit (IMU) can also be used to perform motion estimation by integrating acceleration and gyroscopic data [2], however low-cost units suffer from large amounts of noise which corrupts the estimate.

Visual odometry is a more recent solution to the localisation problem which finds incremental pose changes by aligning image data from visual sensors [3]. However, the nature of incremental estimation means that odometry techniques suffer from errors which grow with time. The challenge is to maintain a high degree of accuracy so the accumulated error does not become intractably large.

RGB-D cameras are a new technology capable of providing additional information in the form of dense depth images of a scene which can be used to constrain odometry estimates, while the availability of 3D colour data is also attractive for detailed map-building applications. As such, this thesis aims to address the localisation problem using RGB-D cameras for odometry.

1.2 Background

The availability of dense colour and depth information has led to a number of techniques being proposed which use RGB-D cameras for odometry. Huang *et al.* [4] estimate the pose change between images by aligning visual keypoints projected to 3D using the depth image. This process is illustrated in Figure 1.1. The filtering of outliers and a bundle adjustment procedure are used to increase robustness. The depth image alone is used in the KinectFusion 3D scanning package [5], where point clouds are projected from the depth image and aligned using a modified version of Iterative Closest Point (ICP) [6]. However both of these methods do not make full use of the dense availability of colour and depth data from RGB-D cameras.

Kerl *et al.* use the entire depth and colour images and attempt to minimise a photometric reprojection error [7], while the RGBD-ICP algorithm of Henry *et al.* [8] uses sparse visual keypoints to initialise ICP on dense point clouds. This combines the speed and robustness of keypoint matching with the accuracy of correctly initialised cloud alignment. However, these systems have demonstrated results only in indoor environments and over short distances of up to around 100m. An RGB-D odometry system for a variety of outdoor and unstructured terrain environments has yet to be presented in the literature, as these cameras rely on Infrared (IR) light to measure depth which suffers from

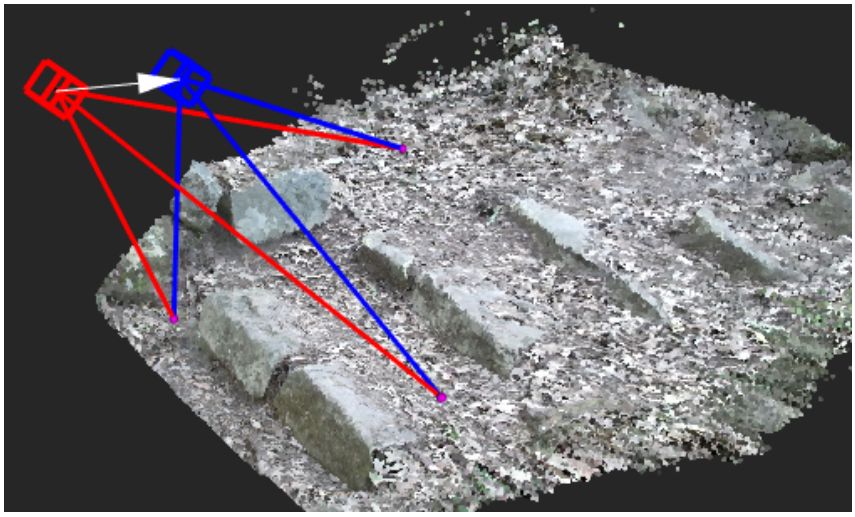


Figure 1.1: Visual odometry tracking 3D keypoints to estimate camera motion

interference in sunlight.

To constrain orientation errors in stereo odometry, Konolige *et al.* [9] fuse odometry estimates with absolute roll and pitch and relative yaw measured by a navigation-grade IMU. Adding these constraints to the orientation was found to improve the estimate over long distance trials. There exists the opportunity to employ a similar approach to long-range RGB-D odometry.

1.3 Research Question and Objectives

With the above motivation and background in mind, this thesis aims to address the following research question:

> *What accuracy in long range outdoor odometry can be achieved by combining an RGB-D camera with an IMMU?*

In addressing this research question, there are a number of objectives to be met:

- Construct a hardware implementation suitable for outdoor use
- Design an odometry system that makes full use of data available from RGB-D sensors
- Fuse with inertial and magnetic measurements
- Test positioning accuracy on a variety of outdoor terrain environments over hundreds of meters

1.4 Project Overview

For the purpose of gathering colour and depth data, the Microsoft Kinect for Windows version 2 RGB-D camera is used. The Kinect is a low-cost RGB-D camera popular among roboticists for its ease-of-use, price and combined colour and depth sensing capabilities [10]. To date, RGB-D cameras have rarely been used in outdoor applications because of their reliance on IR light to measure depth using either structured light or Time of Flight (TOF) techniques [10].

In order to meet the objectives of an outdoor odometry system the hardware implementation includes the design of a truncated pyramid, or frustum, shaped shield which blocks external sunlight in outdoor tests, allowing the Kinect to gather depth data. The frustum design is shown in Figure 1.2.

Figure 1.2: A frustum-shaped shield protects the Kinect field of view from sunlight

This thesis proposes an odometry system which combines RGB-D odometry with an Attitude-Heading Reference System (AHRS) computed using a low-cost IMMU. The odometry system performs two types of odometry, taking inspiration from the RGB-D-ICP algorithm of Henry *et al.* [8]. Visual odometry is performed by finding keypoints in the colour images, matching them across images, projecting them to 3D using the depth images, and aligning the keypoint point clouds. Depth odometry is then performed using the full dense point clouds projected from the depth images, initialised with the visual odometry estimate and aligned using the ICP alignment algorithm. The nature of the terrain environment in which the camera is operating is then used to select between these two odometry types.

The IMMU is used to compute an orientation relative to a North-East-Down (NED) coordinate system using an AHRS. This is fused with the selected odometry estimate using an EKF to constrain orientation errors. Constraining the orientation estimate allows

errors of less than 5% to be achieved over distances of hundreds of metres in a variety of unstructured outdoor environments. Graphics Processing Unit (GPU) accelerated implementations of several key odometry processing steps are also demonstrated, with improvements in speed by a factor of 2-60.

1.5 Thesis Structure

The remainder of this thesis paper is structured as follows:

- **Chapter 2** reviews the existing literature on existing odometry techniques, data fusion and RGB-D camera hardware
- **Chapter 3** describes the combined visual, depth and IMMU odometry system
- **Chapter 4** details the hardware components used and their calibration procedures
- **Chapter 5** presents experimental results from system testing in a variety of conditions
- **Chapter 6** summarises the thesis in the conclusion and presents recommendations for future work

Chapter 2

Literature Review

With its importance to the field of autonomous robotics, odometry is an area of significant research. Many different approaches are taken in the analysis of visual, depth and inertial and magnetic data, some emphasising computational efficiency, others aiming for accuracy of positioning. This chapter will begin by reviewing methods of performing Visual Odometry (VO) using standard digital cameras, in either mono or stereo configurations. Although this thesis employs RGB-D cameras, this is important background as many techniques in RGB-D odometry were inherited from VO. Existing methods for performing odometry using RGB-D cameras will then be covered, followed by techniques of motion and orientation estimation using an IMMU. Then techniques for fusing these different odometry techniques will be considered, and finally options for the choice of RGB-D camera will be covered.

2.1 Visual Odometry

The beginnings of VO as it stands today lie in the Structure From Motion (SFM) techniques developed in the 1980s [11]. These techniques operated on sets of images collected from a static scene to reconstruct both a three dimensional model and the camera motion. Figure 2.1 shows an example of reconstructing a scene from circular camera motion about a car. However, processing all the images at once meant tracking could only be performed offline, as the required computation time grew with the number of images. This made it an impractical solution for what has historically been the driving factor for VO; its use in planetary rovers exploring the surface of Mars, where GPS is unavailable and wheel odometry is susceptible to errors from wheel slippage in the soft Martian sand [12].

Compared to SFM, the process of VO computes camera motion only between successive pairs of frames or a bundle of a few frames, and compounds these incremental position changes to recreate the camera trajectory. Now involving much smaller datasets, methods of computing pose change could be defined which allowed execution at video frame rates. The phrase 'Visual Odometry' was first used by Nister *et al.* in their 2004

paper [3] which described a method of real-time estimation of camera pose using either a single camera or a stereo camera pair.

The method of Nister *et al.* uses Harris corner detection to find features in images and matches them in subsequent images using a correlation measure applied to a window around the feature. Three dimensional positions of features were found using either stereo camera triangulation or in the mono camera case by using the five-point method [13] to estimate camera motion at an unknown scale and then triangulating for scale using feature correspondences found across multiple images.

To reject incorrect matches Random Sample Consensus (RANSAC) was used [14] to select an inlier set. RANSAC takes a random minimal subset of the feature correspondences to compute a camera motion hypothesis and then finds how many other correspondences agree with this hypothesis within some error threshold. If the consensus set of agreement is sufficiently large then a refined motion estimate is generated by finding the best fit to the consensus set. If the consensus set is not large a new random sample is chosen and the process is repeated. This offers a great advantage over model fitting using naive least squares estimation by its ability to reject outliers [15]. Implementation of this system achieved errors of around 1-5% compared to a DGPS ground truth reading over a several-hundred metre outdoor drive.

Figure 2.1: Estimating camera poses and scene structure using SFM [14]

Since the work of Nister *et al.* there have been many improvements to VO systems proposed. Evolution in the field of feature detection by corner finding includes the Features from Accelerated Segment Test (FAST) [16] and Adaptive and Generic Accelerated Segment Test (AGAST) [17] methods. These detectors offer reductions in computation time compared to Harris corners by using machine learning algorithms. Other types of keypoint detectors are blob-based, looking for a region of pixels which differ from their surroundings in some way. The Star detector is the OpenCV implementation of the Centre Surround Extrema (CenSurE) algorithm [18], which looks for a region of light pixels surrounded by darker or vice versa at a range of sizes, performing this search efficiently using integral images.

For the purpose of matching detected features by way of a similarity measure, new, more robust descriptors than normalised correlation have also been proposed. The first

and most well-known of these is the Scale-Invariant Feature Transform (SIFT) descriptor [19] which computes a histogram of image intensity gradients in a window surrounding a point of interest and which has been shown to be robust to changes of illumination and viewpoint changes up to 60° [20]. However SIFT suffers from a long computation time and so the Speeded Up Robust Features (SURF) descriptor [21] was proposed which makes use of integral images to approximate the Hessian second derivative matrix of image intensity. Even quicker to compute is the Binary Robust Independent Elementary Features (BRIEF) descriptor [22] which performs random pixel intensity comparisons in a surrounding window to create a binary descriptor for the keypoint feature. These can be quickly compared using a Hamming distance: a distance measure between two binary strings which counts the number of different bits.

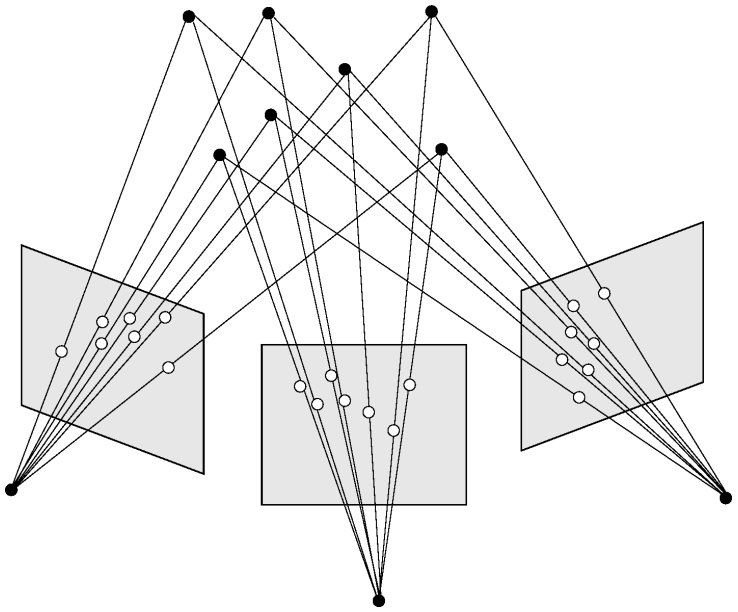


Figure 2.2: SBA estimates both feature and camera positions using measurements from multiple images [23]

In addition to new feature detectors and descriptors for robustness and efficiency, other research in feature-based odometry aims to reduce drift using techniques like bundle adjustment and loop closure [20]. Windowed bundle adjustment optimises the estimate of the camera pose by comparing feature observations across more images than just the preceding image, as shown in Figure 2.2 [24]. The open source Sparse Bundle Adjustment (SBA) library performs bundle adjustment efficiently by exploiting the sparse nature of the Jacobian derivative matrix used in the iterative nonlinear optimisation process [23]. Loop closure aims to detect when the camera path returns to a point it has already observed by comparing current image features to a library of features from past key-frames. The loop closure provides a position fix for the path returning to the same point, which is then added as a constraint to a pose-graph optimisation. This technique is borrowed from research into Simultaneous Localisation and Mapping (SLAM) concerned with creating

globally consistent maps, and can be used to reduce drift in VO systems [4].

2.2 RGB-D Odometry

With a basic understanding of the history of VO, other odometry techniques which use the colour and depth information simultaneously available from RGB-D cameras will now be covered. One early method that only used the depth data was the KinectFusion 3D scanning and reconstruction algorithm [5]. KinectFusion estimates camera motion by using ICP to align the 3D point clouds projected from the depth image. The original ICP algorithm described in [6] used a closest point search between the point clouds to establish matches and iteratively minimised a point to point error. However, the closest point search has quadratic complexity in the size of the point clouds, which is impractical when the Kinect outputs over 300,000 points per frame at 30 frames per second (fps). Instead, KinectFusion uses projective data association, first described in [25], to perform matching with linear complexity.

The depth images are then integrated into a voxel, or three-dimensional pixel, model of the scene which is used as a reference to align further point clouds against. The accuracy of this tracking scheme in the presence of sufficient 3D features is confirmed by the smoothness and completeness of the models generated. Figure 2.3 shows the original noisy depth images, coloured by normal direction, and the smooth 3D surfaces created by the reconstruction.

To achieve an execution rate of 100Hz, the KinectFusion algorithm uses a high-end GPU running the NVidida Compute Unified Device Architecture (CUDA) framework for parallel processing. However, KinectFusion is limited to extremely small reconstruction volumes, no larger than a cube of 2m sides, and relies on the presence of sufficient features in the depth image for camera localisation.

Other odometry methods using RGB-D cameras make use of both the colour and depth data. In a manner similar to stereo odometry, the system of Du *et al.* detects SIFT features in the Kinect colour image and aligns the keypoints, with the difference that 3D keypoint locations are found using the Kinect depth image rather than stereo triangulation [26]. This allows them to perform what they call 3D RANSAC, a variation of 3-point RANSAC that rejects poor camera motion hypotheses by the number of visibility conflicts - whether the camera can see points that the previous depth image and motion hypothesis indicate should be obscured behind an obstacle. The system runs at interactive rates of 4-7Hz on a GPU. However, good tracking and alignment requires interaction with a user moving as directed by the program, limiting its potential for autonomous systems.

An alternative approach is taken by Kerl *et al.* in their Dense Visual Odometry (DVO) system, aiming to achieve fast odometry processing on a Central Processing Unit (CPU) suitable for localising an autonomous quadcopter indoors. DVO uses the entire RGB-D

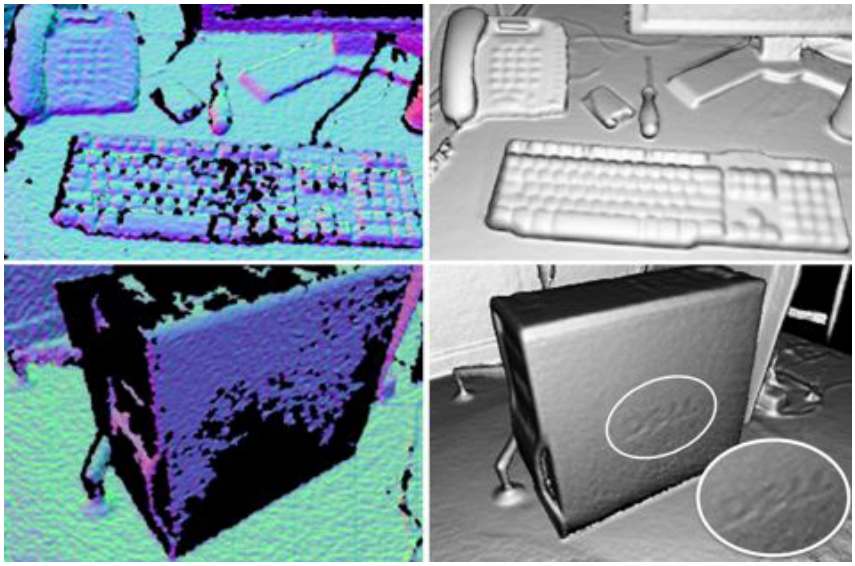


Figure 2.3: Smooth voxel surfaces generated with KinectFusion from raw depth data [5]

image to find the camera motion maximising photo-consistency between subsequent colour and depth frame pairs [27], [7]. This is performed by projecting the previous colour image to 3D, transforming it by the camera motion estimate, then reprojecting the points back to an image. The difference between the mapped previous image and the current colour image is the photometric error. The error is minimised with respect to the camera motion using least-squares. While a fast execution rate of 12.5Hz on a CPU was attained, the drift between the true camera position observed by a motion capture system and the odometry estimate was found to be 5-7cm/sec. This would result in large position errors when operating for prolonged periods.

The Kintinuous system of [28] extends both DVO and KinectFusion by combining the photo-consistency condition with ICP localisation to perform coloured 3D mapping of arbitrarily large volumes. The photometric error and the ICP point to plane error functions are minimised simultaneously with respect to the camera motion, and colour and depth data is added to the map in real time by processing on a GPU. While good reconstructions were achieved over small volumes, camera position drifts of between 3-10cm/sec were observed, which again would cause significant errors over long distance paths.

Henry *et al.* instead combine visual odometry with ICP in their RGB-D-ICP algorithm [8]. They detect features in successive frames using a GPU implementation of the SIFT algorithm and find the best transformation between 3D features using 3-point RANSAC. This is then used to provide an initial estimate for a point to point ICP algorithm. Global consistency is provided by detecting loop closure by comparing SIFT features to previous keyframes. This adds a constraint to the relative poses between individual frames, which is optimised using the Tree-based netwORk Optimizer (TORO) graph optimisation framework of [29] to correct the trajectory and generated map. The

reconstructions of indoor environments over reasonable distances, such as Figure 2.4 at 114m in length, show good consistency. As such, this method provides the inspiration for the long-range outdoor odometry system presented in this thesis as it combines the properties of robust alignment of visual features over multiple frames with the dense constraints from localisation with correctly initialised ICP.

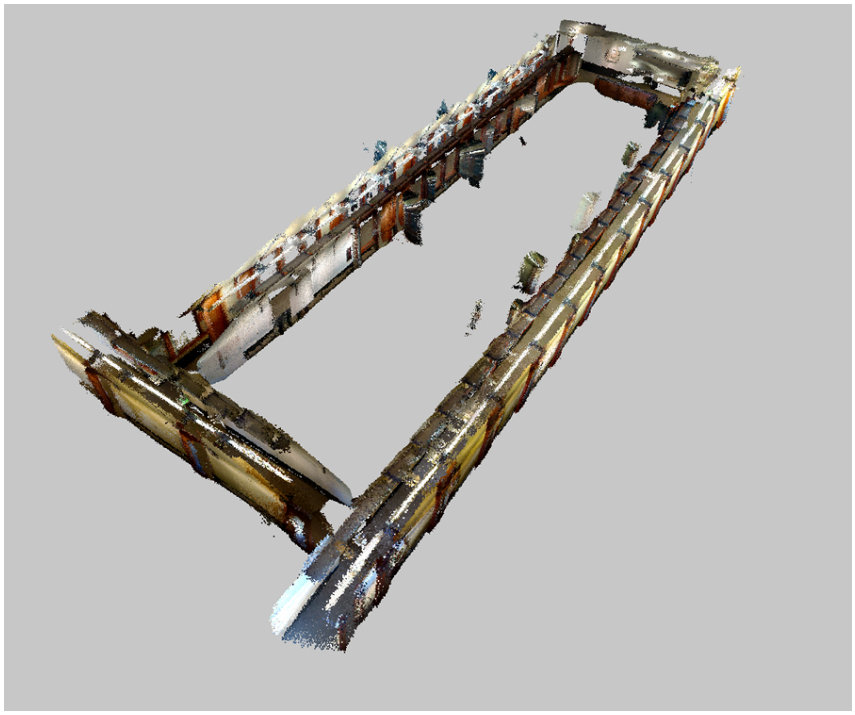


Figure 2.4: Indoor reconstruction using RGB-D-ICP [8]

2.3 IMMU Odometry

The use of the accelerometers and gyroscopes in an IMU, and the additional magnetometer in an IMMU, have a long history in performing odometry, particularly in the aerospace field. However its application to mobile robotics has been limited in the past by the cost of the high-end inertial sensors typically used in aerospace applications [2]. Barshan and Durrant-Whyte in [2] proposed one of the earlier examples of the use of an IMU for ground robot navigation using an EKF to fuse gyroscopic rotation rate estimation with accelerometer measurements of the gravity vector to compute attitude and heading angles. They also tested double-integrating the acceleration to get velocity and position readings, however the high susceptibility of this process to noise made it unusable for operation times longer than about 5 seconds. In 3 minutes of driving, errors of up to 30m in position were recorded from double-integration, with an average drift of 1-8cm/sec. They conclude that a low-cost IMU is unsuitable for providing position estimates and is better suited to calculating orientation in an AHRS.

This is also of interest to the biomedical community in the effort to track human body parts and motion profiles as part of rehabilitation after injury. Sabitini provides an in-depth review of descriptions of rotation in addition to deriving an EKF for combining accelerometer, gyroscopic and magnetic measurements to form an AHRS [30]. During periods of small linear accelerations the known vectors of acceleration due to gravity and Earth's magnetic field are used to establish absolute orientation, and during periods of fast motion gyroscopic angular rate information can be integrated to estimate changes in orientation. Errors of less than 1° were observed over a thirty second window as observed by an external motion capture system.

Despite the popularity of the EKF formulation there are other approaches, such as the popular open-source quaternion gradient descent algorithm of Madgwick *et al.* [31]. The authors claim that the complexity of the EKF to implement, its computational load and the required IMMU sampling rate are all prohibitive to its use. Instead, their quaternion implementation uses steepest descent optimisation with an analytically derived Jacobian to efficiently find orientation within a known vector field. The ability to reject local magnetic distortion through calibration is also presented. Sampling rates of 50Hz to 512Hz all perform consistently with an angular Root Mean Square Error (RMSE) of around 1° for both static and dynamic cases.

Mahony *et al.* also describe a computationally simple AHRS using rotation matrices in a complementary filter [32]. Integration of the gyroscopic rates is tuned by feedback from the acceleration due to gravity and magnetic field vectors, providing a reference for the estimated downwards and North directions respectively. This system was used to compute an AHRS as the low-cost IMMU used in this thesis includes an open source implementation of the algorithm of Mahony *et al.* in its firmware.

2.4 Data Fusion

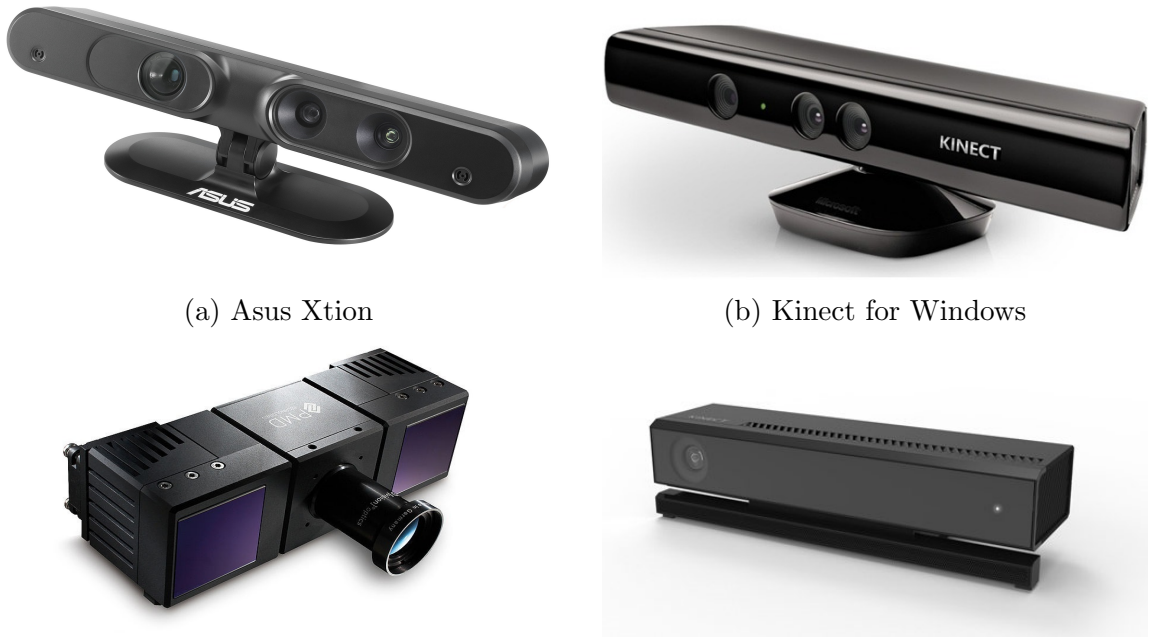
A complete AHRS has been shown to be beneficial for purposes of visual odometry. Kneip *et al.* use the orientation estimate to constrain the search for matching features and to provide an initial frame to frame orientation estimate [33]. Intended for application to a micro air vehicle, they claim that integration of the rotation prior from the IMU allows for the operation of the visual odometry system at real time rates on the low-power on-board processor.

Konolige *et al.* go further than just using the IMU to constrain the feature matching, but rather use it to compute an orientation reading to combine with the stereo odometry estimate. They employ a navigation grade IMU to provide absolute measures of pitch and roll using the accelerometer gravity vector and a measure of relative yaw by integrating gyroscopic data, and fuse the orientation as an EKF update step for an odometry relative pose change as the EKF prediction. They demonstrate good results on kilometres of

camera motion, and this forms the basis for the fusion of the AHRS orientations in this thesis. However, magnetometer data from a full IMMU is also used with Mahony's AHRS to compute orientation relative to an NED coordinate system. This orientation is then fused with the odometry pose estimate using a quaternion based EKF.

2.5 Hardware

There are a number of choices available on the market for providing RGB-D data, including both TOF and structured light varieties. The first depth cameras developed, such as the PMD CamCube shown in Figure 2.5c, were TOF cameras that measured range by pulsing IR light at high frequencies and measuring the time taken for the signal to return to an imaging chip [34]. Although able to achieve depth accuracy on the order of millimetres they were initially limited by very low resolutions, 64×64 being state of the art in 2004. Furthermore these cameras were originally quite expensive for robotic applications.



(a) Asus Xtion

(b) Kinect for Windows

(c) PMD CamCube

(d) Kinect for Windows version 2

Figure 2.5: Examples of RGB-D cameras

A more recent and affordable development is the structured light camera, which projects a grid of IR dots and performs stereo-style triangulation on the unique dot pattern to estimate range [10]. The simplicity of operation allows high frame rates and resolution to be achieved. Primesense developed the technology behind the first of these cameras, such as the Asus Xtion shown in Figure 2.5a, however it was the release of the Kinect by Microsoft - Figure 2.5b, that brought RGB-D cameras into prominence. With

a large multinational corporation behind it the Kinect had extensive support, drivers and documentation with intended applications in the natural user interaction and casual video games and fitness markets.

Although the Kinect has lower accuracy than the TOF based models [10], in addition to its 640×480 resolution depth camera it also included a colour camera at the same resolution, making it a true RGB-D sensor capable of outputting both data types simultaneously. This has resulted in its widespread adoption in the computer vision and robotics community, including the development of open source drivers, programming interfaces and processing libraries on top of the official Windows driver and software development kit [35].

Most recently, a new version of the Kinect for Windows sensor, the Kinect for Windows version 2 shown in Figure 2.5d, was released in mid-2014. Although full operation details have not been released, it uses the TOF technique to provide significantly increased accuracy in depth at resolutions of 512×424, well in excess of previous TOF cameras, while maintaining a hobby consumer price point. It also features a wider Field of View (FOV) in both cameras and a full high definition colour camera. For its low cost, accuracy, high resolution data outputs, and the Microsoft developer support network, the Kinect for Windows version 2 is the RGB-D camera used in this thesis.

Chapter 3

Software Architecture

This chapter describes the software architecture of the RGB-D and IMMU odometry system, beginning with a system overview and then continuing in more detail about each step. The flowchart in Figure 3.1 shows the key steps in the odometry implementation. The aim is to determine the camera pose relative to the starting location in a NED coordinate frame.

Two separate odometry types are used. A combined colour and depth odometry system finds sparse visual keypoints in the colour image using the Star feature detector, and describes them using the BRIEF feature descriptor for brute-force matching. Matches are filtered to select good matches, and then keypoints are projected to 3D using the depth image. The keypoint clouds are initially aligned using a closed form solution before being used in a bundle adjustment process to create a refined estimate. Note that the term 'colour image' is used to refer to images from the Kinect colour camera, but for the purpose of performing odometry greyscale intensity images are used.

The other odometry system uses ICP to align the dense point clouds projected from the depth image. As an iterative method ICP can settle into an incorrect local minimum if the two clouds are not initially closely aligned, so it is initialised with the visual keypoint alignment. Of the two odometry estimates the one more likely to be useful in the terrain environment is selected with a switching condition.

In order to constrain the orientation estimate, the IMMU is used to compute an orientation relative to the NED frame using an AHRS. This orientation is fused with the frame to frame pose estimate using an EKF to find the estimated camera coordinates for each image.

Although real-time processing was not an objective of this thesis, GPU implementations of several odometry processing steps were also developed in order to reduce computation times. Where appropriate, a high-level description of the implementation using the NVidida CUDA architecture for parallel computing will be provided.

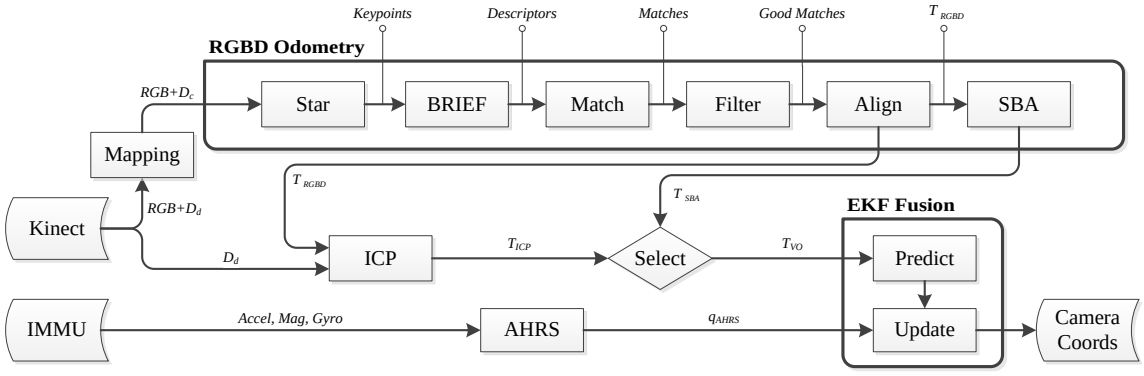


Figure 3.1: Software block diagram

3.1 Image Mapping

In current RGB-D cameras the visual and the depth cameras are usually not located at a common point, and they may have different camera properties including focal length, resolution and field of view. To use the depth data to find the 3D locations of keypoints, the depth image needs to be mapped from the coordinate frame of the depth camera into the coordinate frame of the colour camera.

Figure 3.2 illustrates the mapping process and camera parameters. The depth image in the depth frame D_d contains a range value R_d in millimetres at each pixel coordinate (u_d, v_d), where (u, v) denote column and row coordinates respectively measured from the top left pixel. Using a pinhole camera model, the depth pixels can be projected to 3D points in the depth camera frame (x_d, y_d, z_d) as shown in Equation 3.1. This projection makes use of the similar triangle geometry in the pinhole camera model, and requires the focal length of the depth camera in the x and y directions $(f_{x,d}, f_{y,d})$ as well as the centre pixel location $(c_{x,d}, c_{y,d})$.

$$(x_d, y_d, z_d) = \left(\frac{R_d(u_d - c_{x,d})}{f_{x,d}}, \frac{R_d(v_d - c_{y,d})}{f_{y,d}}, R_d \right) \tag{3.1}$$

Camera parameters including focal length, image centre, and the relative pose between the Kinect colour and depth cameras ${}^{D}T_{C}$ were found using Bouget's toolbox for stereo calibration [36]. This toolbox and the calibration parameters will be covered in more detail in Section 4.3.1. The homogeneous transformation matrix ${}^{D}T_{C}$ is the pose of the colour camera with respect to the depth frame. Points in the depth camera frame are mapped to the colour frame using the inverse transformation ${}^{C}T_{D} = {}^{D}T_{C}^{-1}$, which is the pose of the depth camera with respect to the colour frame, as per Equation 3.2.

$$\begin{bmatrix} x_c & y_c & z_c & 1 \end{bmatrix}^T = {}^{C}T_{D} \begin{bmatrix} x_d & y_d & z_d & 1 \end{bmatrix}^T \tag{3.2}$$

Finally Equation 3.3 shows how these points are reprojected to pixel coordinates in

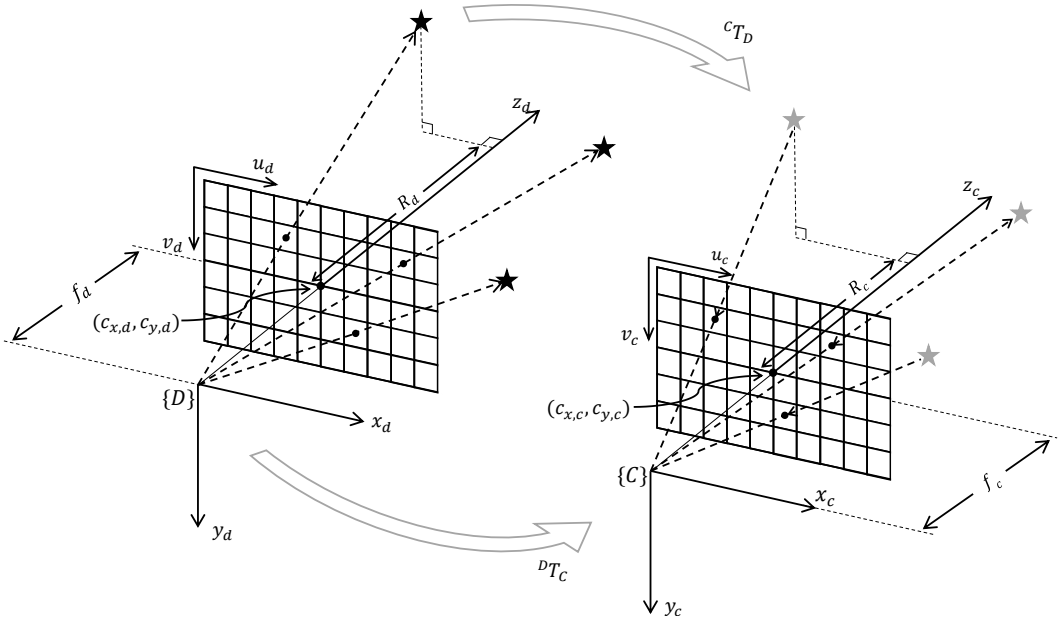


Figure 3.2: Mapping from the Kinect depth to colour frames

the colour image. The notation $\lfloor x \rceil$ describes the round-to-nearest-integer function on x, and the pixel (x_c, y_c, z_c) in the colour image is assigned a range value $R_c = z_c$. Focal lengths and camera centres for the colour camera follow the same subscript notation as for the depth camera.

$$(u_c, v_c) = \left(\left\lfloor \frac{x_c f_{x,c}}{d} - c_{x,c} \right\rceil, \left\lfloor \frac{y_c f_{y,c}}{d} - c_{y,c} \right\rceil \right) \tag{3.3}$$

The image mapping process is performed in parallel on a GPU using a single GPU thread per depth image pixel. The mapping process output is the mapped depth image D_c, an image with the same resolution as the colour image, with a depth reading wherever there is one available from the mapping process, and a reading of zero otherwise.

3.2 Star Keypoint Detection

There are a number of requirements on the detection of visual features in an image if they are to be useful keypoints for camera motion estimation. As described by Scaramuzza *et al.* [11] and Fraundorfer *et al.* [20], the appearance of keypoints should be invariant to scale, orientation and lighting, and they must be able to be repeatedly and accurately located in an image. The first three conditions require the keypoint to maintain a consistent appearance both as the camera moves and observes the features from different positions, and as lighting conditions change such as under shifting outdoor shadows. The last condition requires the keypoint to have a well defined, stable location, as opposed to potentially being detected anywhere within a surrounding region. Combined, these con-

ditions define keypoints that can be consistently and precisely tracked in order to infer corresponding camera motion.

Some popular feature detectors include Harris corners, FAST [16], or Oriented FAST and Rotated BRIEF (ORB) [37]. These detectors all search for corners in images, due to their distinctive response in terms of image intensity gradients which are generally very fast to compute. Corners have a well-defined location, however they can suffer in moving-camera odometry implementations because of their lack of scale invariance [18], with the observed keypoint appearance changing as the camera moves closer or further away. Instead in this system a blob detector called Star is used, which is the OpenCV implementation of the CenSurE algorithm of [18]. Blob detectors search for image patches which differ from their surroundings in some way. CenSurE was chosen as it is still reasonably fast, and it has demonstrated results in long-distance outdoor stereo odometry [9].

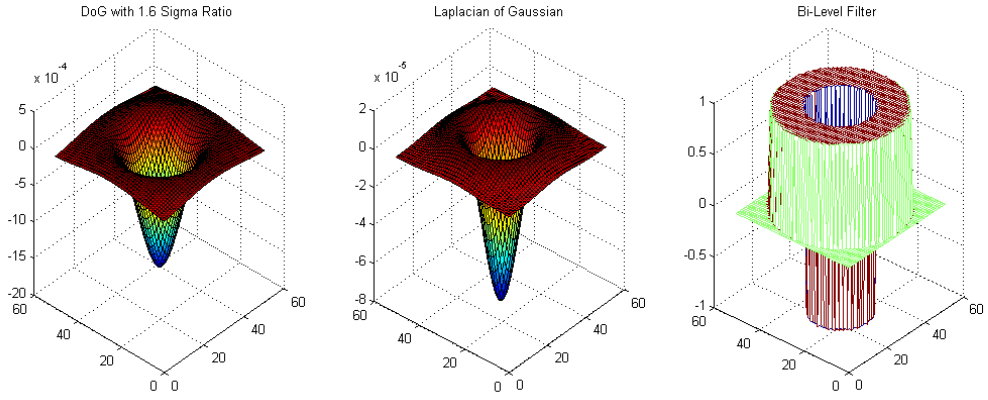


Figure 3.3: Examples of filters used by blob detectors

The CenSurE detector looks for a blob of light pixels surrounded by dark pixels or vice versa by approximating the Laplacian of Gaussian (LoG) filter operator. The widely used SIFT detector approximates the LoG with a Difference of Gaussian (DoG) [19] operator, which is produced by subtracting two Gaussian distributions with some ratio between their standard deviations, sigma. By contrast, the LoG in CenSurE is approximated by a bi-level filter. The bi-level filter has values of either 1 or -1 rather than the true smooth Laplacian of Gaussian distribution. The ideal bi-level filter would be circular in nature, which is shown in Figure 3.3 along with LoG and DoG distributions for comparison. However, the use of an octagonal bi-level filter is a reasonable approximation to a circle and has the advantage that the filter response can be efficiently computed through the use of trapezoidal integral images. In a trapezoidal integral image, each pixel value is the sum of original pixel values above and to one side of it, and diagonally above on the other side. In practice therefore in the original CenSurE implementation an octagonal bi-level filter was used [18].

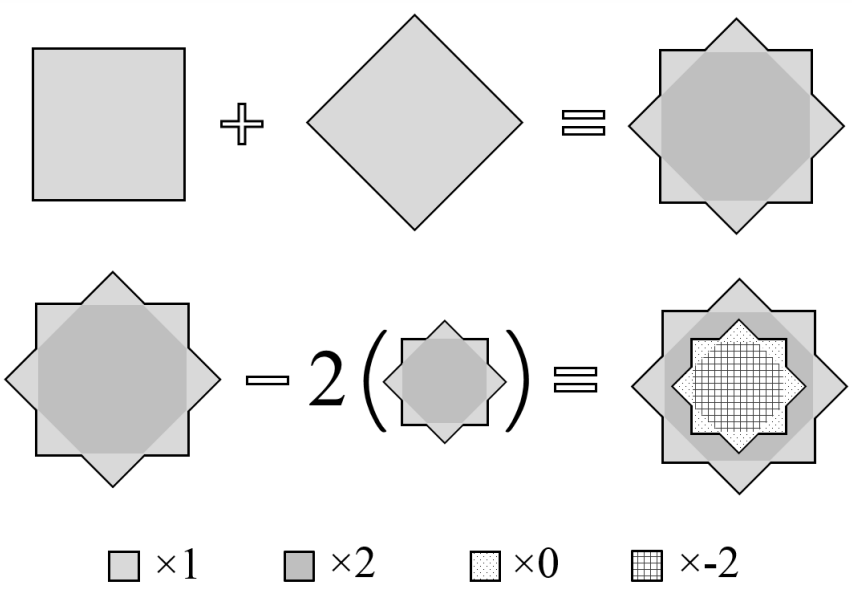


Figure 3.4: The Star keypoint detector kernel

An open source implementation of the CenSurE algorithm, called the Star detector, is available as part of the computer vision processing library, OpenCV [38]. The Star detector differs slightly from the original CenSurE in the nature of its filter kernel, which is a quad-level star shape shown in Figure 3.4. The signature star shape is formed by the addition of a square and a diamond shape which can be quickly computed at arbitrary sizes using rectangular and triangular integral images. In a rectangular integral image each pixel is the sum of all pixels above and to one side, while in a triangular integral image each pixel is the sum of all pixels in a isosceles triangle shape above it.

The filter response is the difference between a larger star shape and two times a smaller one; the first subtraction serving to cancel out the inside of the larger star and the second creates the negative central core of the filter. Specifically in Figure 3.4 the filter response is the sum of multiplying every pixel intensity in the light grey region by 1, the dark grey by 2, the dotted by 0 and the inner hatched area by -2. This process of multiplication and summation is also known as the convolution of an image with a filter. Considering that the greyscale image pixel values range from 0-255 from black to white, roughly circular regions of light pixels surrounded by dark will have a large negative response, while dark pixels surrounded by light will have a large positive response. Uniform pixel intensity patches will cancel out to zero filter response.

The filter is applied to each pixel at a number of different sizes in order to detect different scales of blob features. The absolute value of the response at each size is taken, and if the greatest response of the sizes is above a threshold then the pixel is stored as a feature candidate. Local non-maximums are then suppressed to reject multiple keypoints being found within the same blob. Keypoints with a strong edge response are also rejected due to the instability of keypoint locations along the uniform appearance of

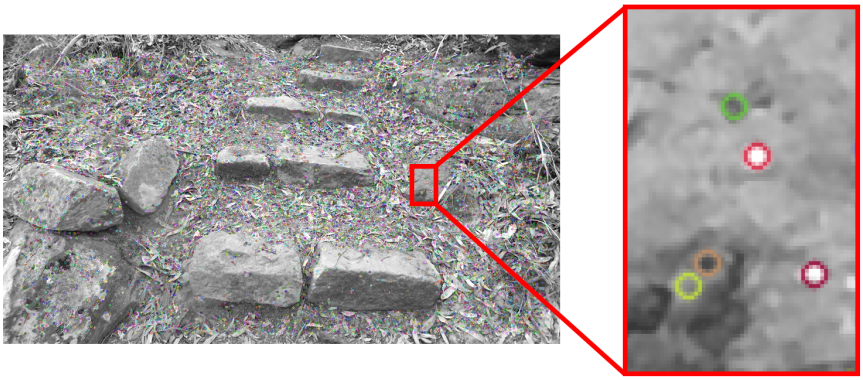


Figure 3.5: Examples of Star keypoints

an edge. The remaining candidates are the Star keypoints. Figure 3.5 shows an example of keypoints found on an outdoor image using the Star detector, including a close-up of several keypoints of blobs of lighter coloured pixels surrounded by darker ones and vice versa.

In the OpenCV implementation all steps are performed on the CPU. However in this thesis a modified implementation computed only the integral image and non-maxima suppression on the CPU, while the GPU was used to compute the pixel responses. Each parallel GPU thread computes the Star kernel response of a single pixel at all filter sizes.

3.3 BRIEF Keypoint Description and Matching

With features detected in each image frame, some metric must be applied to correlate features in different images to identify them as the same keypoint. To this end it is possible to compute a descriptor vector summarising the appearance of each keypoint and establish similarity between keypoints by distance between the descriptors.

This descriptor can look at image gradients around a keypoint, like the SIFT descriptor. SIFT computes a 128-element vector which is a histogram of image intensity gradients around the point [19]. It has achieved good results in scale and rotation invariance but the descriptor calculation and finding Euclidean distances between the vectors is computationally expensive. An alternative method called BRIEF [22] uses pixel intensity value comparisons to provide both speed and robustness.

BRIEF selects pairs of pixels in a window surrounding a keypoint and compares their intensity values, with a result of 1 if the first pixel in the pair is brighter and a 0 result otherwise. This result is written into a binary string which forms the keypoint descriptor. The sampling pattern is originally randomly generated, with points taken in a Gaussian distribution, such that more points are selected closer to the centre pixel with less further out. The original pair comparison pattern used by the authors in [22] is shown in Figure 3.6, along with an example of how pixel pairs around a keypoint are compared to

generate the binary results. In total 256 pairs are compared to create a 256-bit binary string occupying only 32 bytes of storage to describe each keypoint.

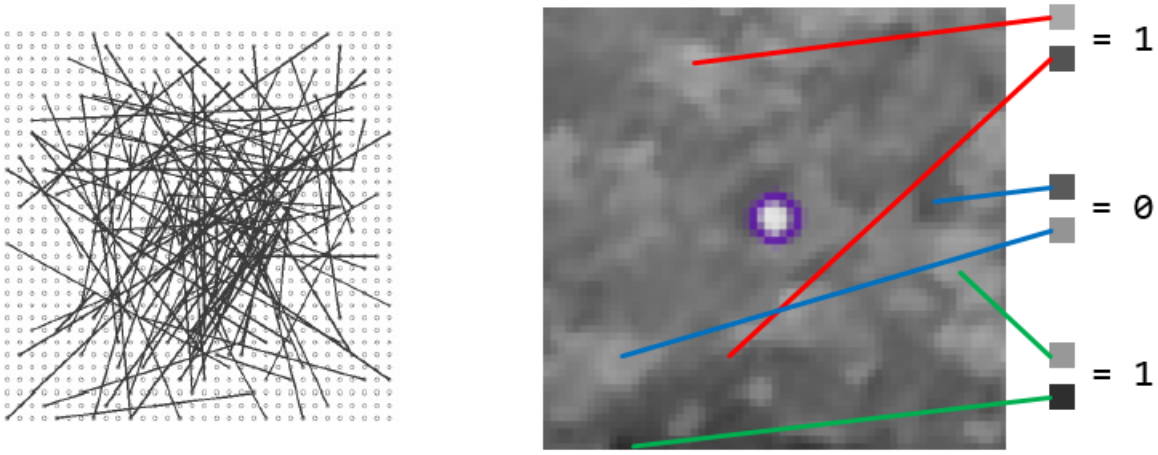


Figure 3.6: BRIEF sampling pattern and pixel intensity comparisons example [22]

A	=	1	0	0	1	0	1	1	0	1
		↕	↕	↕	↕	↕	↕	↕	↕	↕
B	=	1	0	1	1	0	1	0	0	1
Same?		✓	✓	×	✓	✓	✓	×	×	✓

Hamming Distance = 3

Figure 3.7: Example computation of a Hamming distance between binary strings

As the descriptor vectors are binary they can be compared using a Hamming distance, which is the total number of bits which are different between two descriptor strings. A small Hamming distance between the BRIEF descriptors of two keypoints indicates high visual similarity between those keypoints. An example of computing the Hamming distance between two random binary strings A and B, each 9 bits in length, is shown in Figure 3.7.

The Hamming distance can be very efficiently computed on modern processors using an exclusive or operation followed by a bit count of the 1-bits. For example, the Intel Streaming SIMD Extensions (SSE)4 instruction set includes the population count POPCNT instruction for quickly performing a bit-count on 128-bit binary strings. The speed of matching meant a brute-force search strategy was implemented in this thesis, where every keypoint descriptor in one image was compared to every keypoint descriptor in the next image. A match is selected as the smallest Hamming distance between keypoints.

An image of example matches is shown in Figure 3.8. The left and right pictures are cropped images from the Kinect camera, taken one-thirtieth of a second apart at the

Kinect camera frame rate. Keypoints such as those at the bottom left corner of the stone step have been correctly matched across the two images.

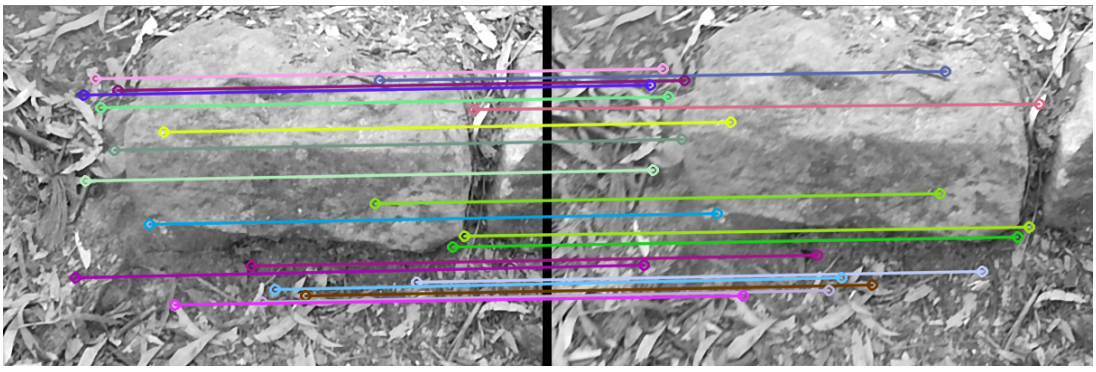


Figure 3.8: Example keypoint matches using BRIEF on a small image region

BRIEF matching was implemented in parallel on the GPU using each GPU thread to compare a single descriptor in the set for one image against every descriptor in the set for the other image. Each thread output for its descriptor the best and second best match in the other set.

3.4 Keypoint Projection

With the keypoints matched, they can now be projected to 3D coordinates using the depth image. However, the different resolutions of the Kinect colour and depth cameras mean that even after the mapping described in Section 3.1, many colour pixels do not have an associated depth reading. In this case, a range reading R_k for a keypoint k at pixel coordinates (u_k, v_k) is interpolated from the valid depth readings in a surrounding 7×7 window.

The size of the window was chosen to make it likely to find valid depth readings, however if multiple potential readings are found then a higher weight should be given to readings close to the centre, as they will more accurately reflect the depth at the centre pixel. The weighting function, illustrated in Figure 3.9 and given in Equation 3.4, uses the square of the Euclidean distance of a pixel from the centre as the weighting. Here $D_c(u, v)$ is the range at coordinates (u, v) in the depth image D_c mapped to the colour coordinate space. The weighting $w_{i,j}$ is calculated at coordinates (i, j) relative to the keypoint location, and the weighting is set to zero if there is no valid depth reading at that pixel or at the centre pixel to avoid divide-by-zero.

$$w_{i,j} = \begin{cases} \frac{1}{i^2+j^2} & if\ D_c(u_k + i, v_k + j) > 0 \\ 0 & otherwise,\ or\ if\ i = j = 0 \end{cases} \tag{3.4}$$

The sum of weighted pixel values is normalised by the sum of the total weightings to give the range estimate, as per Equation 3.5. If there are no valid readings in the window

a depth of zero is returned and the keypoint is rejected from the useful set. This includes both keypoints at the edges of the colour image, which due to its wide FOV are outside the depth camera FOV, and keypoints on edges or reflective surfaces which do not have any corresponding valid depth readings.

$$R_k = \Sigma_{i=-3}^{3}\Sigma_{j=-3}^{3}\left(\frac{w_{i,j} \times D_c(u_k + i, v_k + j)}{\Sigma_{i=-3}^{3}\Sigma_{j=-3}^{3}(w_{i,j})}\right) \tag{3.5}$$

Once the depth of the keypoint has been established using this interpolation, it can be projected to 3D coordinates using Equation 3.1, albeit using the colour camera parameters. For all the matches between a pair of images, two sets kp_{curr} and kp_{prev} are created each containing the 3D locations of all matched keypoints in the current and previous image frame respectively.

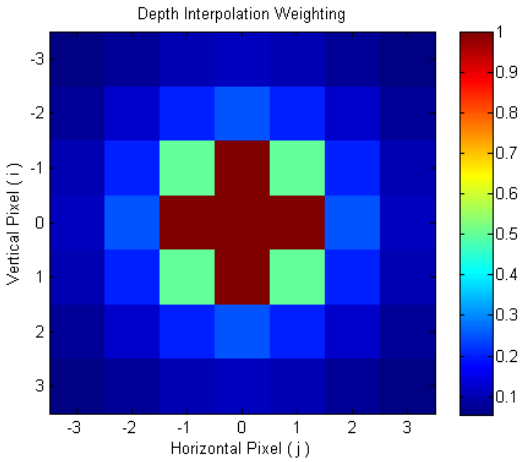


Figure 3.9: Weighting applied to surrounding pixels for depth interpolation

3.5 Match Filtering

Once the keypoints have been matched and projected, they are filtered to select good matches. First, visually ambiguous matches are removed by applying the test originally proposed by Lowe for SIFT descriptors [19], but which is equally applicable to BRIEF descriptors. Lowe's test requires the distance between the descriptors for the best match to be better than the distance to the second-best match by some factor. This test stops keypoints of similar appearance from being incorrectly matched to each other by requiring a strong and unique match. In this thesis a factor of 0.6 is used such that the distance to the best match must be less than 0.6 times the distance to the second best match.

The second filter, RANSAC [15], uses geometrical constraints to estimate a mutually consistent inlier set. This operates under the assumption that the motion of the camera should cause the keypoints in the first image to move in a consistent way to appear in the second image, which is true when keypoint tracking is perfect. However it is also

expected that there will be some incorrect matches which will likely have a very different keypoint motion between frames. In order to reject these outliers, RANSAC searches for the largest set of keypoint matches where the keypoint motion is consistent with a single camera motion hypothesis.

The process is outlined in Algorithm 1. For a fixed number of iterations $maxIter$, a random sample set s of 3 matches is drawn from the total set of N good matches output from Lowe's test. The set s selects the 3D keypoint locations from the current and previous image keypoint sets kp_{curr} and kp_{prev} and a transform T_{RANSAC} which aligns these sample point sets is found using the method described in Section 3.6. This transform is a frame to frame camera motion hypothesis.

Using this transform T_{RANSAC} each keypoint k in the previous image at point (x_k, y_k, z_k) is mapped to a predicted point (x_p, y_p, z_p) and reprojected to predicted image coordinates (u_p, v_p). An error ϵ_{RANSAC} is the square of the Euclidean distance between the predicted pixel coordinates and the measured match location in the current image $(u_{k,curr}, v_{k,curr})$. A threshold on this reprojection error determines whether the match to keypoint k is an inlier to the motion hypothesis. A threshold on the distance squared of 9 is used, requiring the match keypoint to be at most 3 pixels distant from its predicted location to be counted as an inlier. The largest inlier set across all iterations is the output of RANSAC, and all matches in the inlier set are retained while the others are discarded.

Algorithm 1 RANSAC inlier finding procedure

maxInliers ← Inliers ← 0
for $i < maxiters$ **do**
    s ← GetRandSample(N,3)
    T_{RANSAC} ← EstimateTransform($kp_{curr}(s), kp_{prev}(s)$)
    for each keypoint k in kp_{prev} **do**
        (x_p, y_p, z_p) ← TransformPt($x_k, y_k, z_k, T_{RANSAC}$)
        (u_p, v_p) ← Reproject(x_p, y_p, z_p)
        $\epsilon_{RANSAC} \leftarrow (u_p - u_{k,curr})^2 + (v_p - v_{k,curr})^2$
        if $\epsilon_{RANSAC} < thresh$ **then**
            Add k to Inliers
        end if
    end for
    if size(Inliers) > size(maxInliers) **then**
        maxInliers ← Inliers
    end if
end for

In this thesis the CPU was used to select the random sample and estimate a transform at each iteration of RANSAC, while simultaneously parallel GPU threads performed the inlier counting for the previous iteration. Each GPU thread performed the transform, reprojection and error computation for a single keypoint and output whether the matching keypoint was an inlier within the reprojection threshold.

3.6 Keypoint Cloud Alignment

The problem of estimating camera motion between the two frames now becomes one of estimating the transform that will align the two point clouds of the matched and projected keypoints. Letting these point sets be a and b, the aim is to find the homogeneous transformation T_{RGBD}= $[R, t]$, combining orthonormal rotation matrix R and translation vector t, which minimises a point to point error metric given in Equation 3.6. This error ϵ_{align} is the square of the Euclidean distance between matched points under the given transformation.

$$\epsilon_{align} = \Sigma_{i=1}^{n} ||Ra_i + t - b_i||^2 \tag{3.6}$$

Equation 3.6 is formulated as a least-squares estimation problem, and a number of solutions have been proposed. Horn *et al.* [39] and Arun *et al.* [40] have closed-form solutions using quaternions and Singlar Value Decomposition (SVD) respectively. These methods however can return reflections instead of rotations in noisy data. Umeyama's method [41] extends their work on alignment using SVD for a closed form solution which is robust against noise and enjoys an open-source implementation in Point Cloud Library (PCL) [42].

For the full derivation see Umeyama's 1991 paper [41], but in brief the method states for matched point clouds a and b the minimising transform is found by taking the SVD of the covariance Σ_{ab} between the clouds by first finding the mean values of the two clouds μ_a and μ_b.

$$\mu_a = \frac{1}{n}\Sigma_{i=1}^{n} a_i \qquad \mu_b = \frac{1}{n}\Sigma_{i=1}^{n} b_i \tag{3.7}$$

$$\Sigma_{ab} = \frac{1}{n}\Sigma_{i=1}^{n} (b_i - \mu_b)(a_i - \mu_a)^T \tag{3.8}$$

Letting $SVD(\Sigma_{ab}) = UDV^T$, and letting S be the identity matrix if $det(\Sigma_{ab}) > 0$, or the identity matrix with the bottom-right element -1 otherwise, then the rotation matrix R and translation vector t between the two point clouds is given by Equation 3.9.

$$R = USV^T \qquad t = \mu_b - R\mu_a \tag{3.9}$$

The resulting point cloud alignment gives the initial RGB-D estimate T_{RGBD} of the camera motion between frames.

3.7 Sparse Bundle Adjustment

Bundle adjustment in the context of visual odometry is the estimation of camera poses by tracking keypoints across multiple images, adding additional constraints which increase

the robustness of the estimate. For a set of n camera poses T_i and the set of m observed points P_j in world coordinates, then an error function ϵ_{SBA} is defined as the sum of the Euclidean distances between predicted and measured keypoint locations. Predicted locations are found by transforming a world point P_j into local camera coordinates by applying transform T_i and reprojecting back to image coordinates, while (u_{ij}, v_{ij}) is the measured keypoint location of point j in image frame i.

$$\epsilon_{SBA} = \Sigma_{i=1}^{n} \Sigma_{j=1}^{m} ||Reproj(R_j, T_i) - (u_{ij}, v_{ij})||^2 \tag{3.10}$$

Minimisation of this error function with respect to camera poses and point locations is a high-dimensional problem for even a small number of keypoints and images, and standard numerical solutions to the non-linear optimisation problem become intractable due to the size of the Jacobian derivative matrix. The open source SBA library [23] addresses this issue by recognising that the Jacobian is sparse, containing mostly zero elements. Knowing the structure of the Jacobian, it only performs those computations which contain interesting data to efficiently estimate both the camera poses and the 3D keypoint locations in world coordinates.

In this thesis SBA is applied to keypoints tracked across the last ten frames. Figure 3.10 shows an example of keypoints being tracked during a period of rotation of the camera. Each line connects the image coordinates at which a particular keypoint was observed. The length of the line is therefore indicative of how many images the keypoint was successfully tracked across.

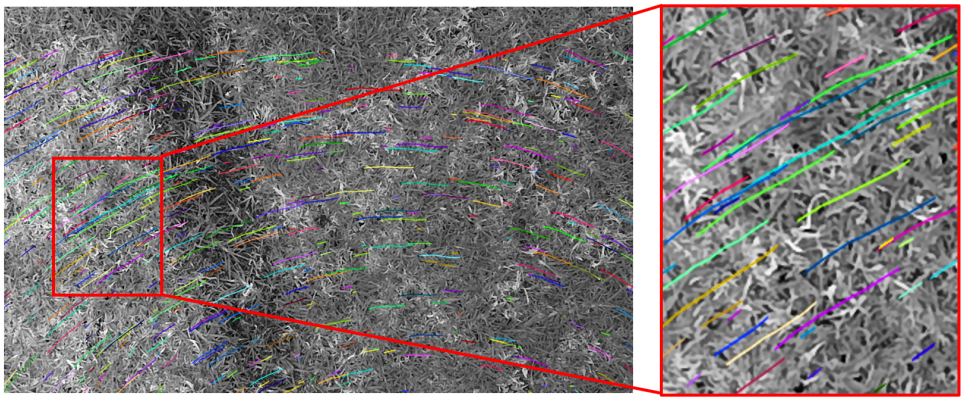


Figure 3.10: Keypoints tracked across frames during a period of rotation

Keypoints are tracked by maintaining a list of observations of keypoints which have been detected in multiple images. Each time a new keypoint is matched to one already in the list, the new keypoint is added as another observation of the same 3D point. If the old keypoint is not on the list, both keypoints are added as observations of a new 3D point. The output of SBA is the refined RGB-D frame to frame pose estimate T_{SBA}.

3.8 Dense Iterative Closest Point

The ICP algorithm attempts to align two point clouds without any prior knowledge of point correspondences [6]. It first attempts to create associations between points in the two clouds in some way, then finds a transform which minimises an error metric between the clouds, and repeats until the process has iterated to a final solution. The original ICP algorithm formed point associations by searching for the closest point in the other cloud, however this is prohibitively slow for large clouds.

The approach taken in Kinect Fusion is called projective data association [5], where a correspondence is registered when two 3D points reproject to the same camera pixel coordinates - a technique only valid at high frame rates or with a good initial estimate of the motion. Furthermore, instead of minimising a point to point error, the error metric ϵ_{ICP} defined in Equation 3.11 is the distance between a point in one cloud b_i and a tangent plane in the other cloud defined by the point a_i and the normal at that point n_i. This has been shown to converge in fewer iterations [43]. The normal vector n_i can be estimated from the point cloud by taking the normalised cross product between vectors connecting adjacent points projected from the depth image. If $P_{u,v}$ is the 3D point projected from pixel coordinates in the depth image (u, v) then the normal can be given by $n_i = (P_{u+1,v} - P_{u,v}) \times (P_{u,v+1} - P_{u,v})$.

$$\epsilon_{ICP} = \Sigma_i((Ra_i + t - b_i) \cdot n_i)^2 \tag{3.11}$$

Projective data association and the point to plane error metric are illustrated in Figure 3.11. Surfaces A and B represent two 3D surfaces projected from the Kinect depth images. Points 1 and 2 on surface A, P_{A1} and P_{A2}, are projected to camera coordinates, and the intersection of the projection ray with surface B defines the corresponding point matches P_{B1} and P_{B2}. The distances d_1 and d_2 are the point to plane distances between matched points in surface B and the tangent planes to the points on surface A.

The point to plane error metric is a non-linear function of the transform, so a linearising approximation is made according to the method in [43] which represents the rotation matrix as three small-angle elementary rotation matrices compounded together, where α, β and γ are small rotations about the x, y and z axes respectively applied in that order.

$$R = \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix} \tag{3.12}$$

Expanding Equation 3.11 with this approximation produces the linearised error function in Equation 3.13, where x, y and z components of vectors are indicated with the corresponding subscript.

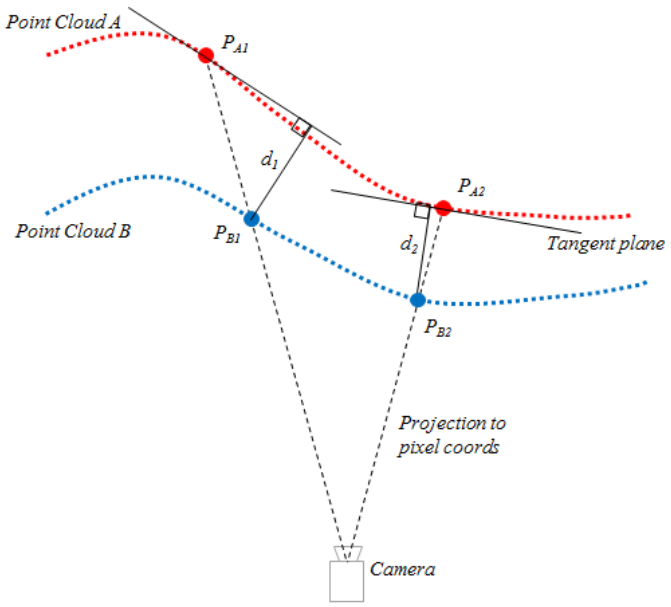


Figure 3.11: Projective data association and the point to plane error metric for ICP

$$\begin{aligned}\epsilon = \Sigma_i \left((a_i - b_i) \cdot n_i + t \cdot n_i + \alpha(a_{iy}n_{iz} - a_{iz}n_{iy})\right. \\ \left. + \beta(a_{iz}n_{ix} - s_{ix}n_{iz}) + \gamma(s_{ix}n_{iy} - s_{iy}n_{ix})\right)^2\end{aligned} \tag{3.13}$$

For simplicity of expression, let c_i be the cross product between the point a_i and its normal, $c_i = a_i \times n_i$, and the vector r be the three small angles $r = (\alpha, \beta, \gamma)$. The simplified linearised error function is given in Equation 3.14.

$$\epsilon = \Sigma_i((a_i - b_i) \cdot n_i + t \cdot n_i + r \cdot c_i)^2 \tag{3.14}$$

To solve for the minimising state vector $\mathbf{x} = [\alpha, \beta, \gamma, t_x, t_y, t_z]$ take the derivative with respect to the state vector and set the derivatives to zero as per Equation 3.15.

$$\begin{aligned}\frac{d\epsilon}{d\alpha} &= \Sigma_i 2c_{i,x}\left((a_i - b_i) \cdot n_i + t \cdot n_i + r \cdot c_i\right) = 0 \\ \frac{d\epsilon}{d\beta} &= \Sigma_i 2c_{i,y}\left((a_i - b_i) \cdot n_i + t \cdot n_i + r \cdot c_i\right) = 0 \\ \frac{d\epsilon}{d\gamma} &= \Sigma_i 2c_{i,z}\left((a_i - b_i) \cdot n_i + t \cdot n_i + r \cdot c_i\right) = 0 \\ \frac{d\epsilon}{dt_x} &= \Sigma_i 2n_{i,x}\left((a_i - b_i) \cdot n_i + t \cdot n_i + r \cdot c_i\right) = 0 \\ \frac{d\epsilon}{dt_y} &= \Sigma_i 2n_{i,y}\left((a_i - b_i) \cdot n_i + t \cdot n_i + r \cdot c_i\right) = 0 \\ \frac{d\epsilon}{dt_z} &= \Sigma_i 2n_{i,z}\left((a_i - b_i) \cdot n_i + t \cdot n_i + r \cdot c_i\right) = 0\end{aligned} \tag{3.15}$$

These equations can be collected into a matrix equation in the form $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a 6×6 symmetrical matrix, and so the equation can be solved using $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. The state vector $\mathbf{x} = [r, t]$ contains the small angle rotations and the translation vector. Initial values for r and t are set using the RGB-D odometry estimate T_{RGBD} so that small

angle assumptions are valid and ICP is more likely to iterate to a correct local minimum solution.

The process of projective data association and minimising the point to plane error is iterated until the change in the state vector since the last iteration is below a certain threshold, indicating that the alignment process has settled to a local minimum. A threshold on the norm of the change in the state vector of 10^{-8} was used. Converting the final state vector to a homogeneous transformation matrix creates the output of the ICP odometry stage, T_{ICP}. The construction of a rotation matrix from the small angles can be found in [43].

As per the original Kinect Fusion implementation, dense ICP was implemented on a GPU for faster computation. Each parallel GPU thread operates on a single point in one cloud, transforming it using the current T_{ICP} estimate then finding a correspondence in the other cloud using projective data association. Each GPU thread is then used to sum the terms in Equation 3.15 for all the points corresponding to a single column of pixels in the depth image, and then the matrix equation is solved on the CPU.

3.9 Selecting the Odometry Source

With two different odometry modalities available, the one more likely to give a better result is selected as the odometry output T_{VO} using a switching strategy. ICP is generally preferred since T_{RGBD} is used to initialise the dense point cloud alignment process. This means that when the visual keypoint alignment is accurate the point clouds will be initialised with an accurate alignment and ICP should return the same pose between frames. When T_{RGBD} is less accurate due to outliers ICP should still iterate towards the correct pose change, albeit taking more iterations and with a higher chance of iterating to an incorrect local minimum.

However this is only true when there is some depth variation to constrain the point cloud alignment. On near planar surfaces like short grass ICP is unlikely to provide any useful data. So a method is required to determine whether a surface is approximately flat, containing few depth features. In this case the refined T_{SBA} pose estimate should be used, otherwise the presence of depth features mean T_{ICP} should be selected.

The test for flat ground is performed by applying a planar fit to the depth image averaged down to an 8×8 image and projected to 3D using a hypothetical pinhole camera model with focal length $f = 1$. Simulated examples of flat and rough terrain, including the planar fit and residuals, are shown in Figure 3.12.

The planar fit is found by minimising the error ϵ_{select} in Equation 3.16 between the points (x_i, y_i, z_i) and the plane defined by the equation $z = Ax + By + C$. This error is only measured in the z-direction rather than the perpendicular distance to the plane, however given that most planar surfaces will be observed with the camera z axes facing

into them from some angle this is an appropriate description for the plane and error term.

$$\epsilon_{select} = \Sigma_{i=1}^{n}((z_i - Ax_i - By_i + C))^2 \tag{3.16}$$

The closed form solution to the planar fit is given in Equation 3.17 [44].

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} \Sigma_i x_i^2 & \Sigma_i x_i y_i & \Sigma_i x_i \\ \Sigma_i x_i y_i & \Sigma_i y_i^2 & \Sigma_i y_i \\ \Sigma_i x_i & \Sigma_i y_i & n \end{bmatrix}^{-1} \begin{bmatrix} \Sigma_i x_i z_i \\ \Sigma_i y_i z_i \\ \Sigma_i z_i \end{bmatrix} \tag{3.17}$$

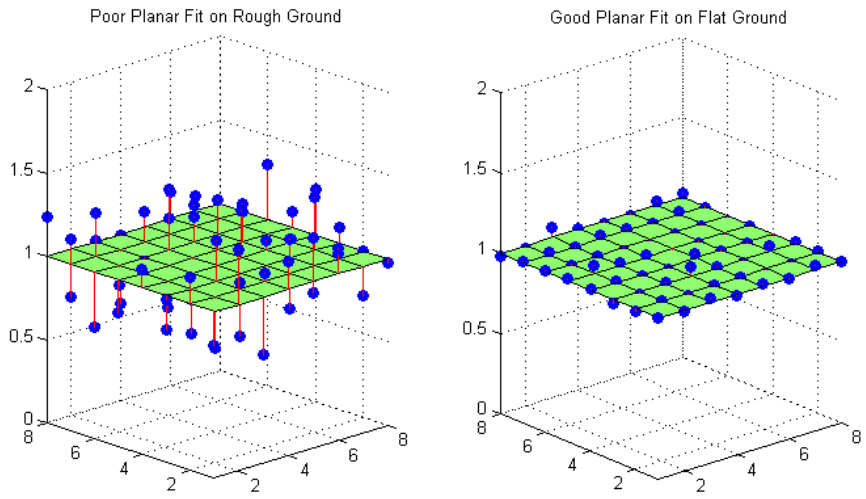


Figure 3.12: Planar fits and residuals for rough and flat ground

The sum of the residuals between the planar fit and the data points was then used to determine whether the surface was rough. A threshold of 40 was used such that when the sum of the residuals were greater than this value then the terrain was considered to have sufficient depth features and T_{ICP} was selected, otherwise T_{SBA} was selected.

3.10 Attitude Heading Reference System

An AHRS provides a measure of orientation, usually with respect to an NED coordinate system. The IMMU used in this thesis comes with firmware to implement the AHRS of Mahony *et al.* [32].

As briefly introduced in Section 2.3, the basic principle of operation is to initialise the orientation estimate using accelerometer and magnetometer vector measurements as a reference for downwards and North respectively. Each new IMMU data line includes three-axis accelerometer, magnetometer and gyroscope readings. The gyroscope rotation rates are measured with respect to the body frame of the IMMU, so for each new data line they are rotated into the NED world frame using the rotation estimate of the previous

time step.

Yaw is then calculated using the magnetometer vector, and roll and pitch are calculated using the accelerometer. These angles are inputs to a proportional-integral controller which modifies the gyroscopic rotation rates to account for the estimated roll, pitch and yaw. The corrected rotation rate is then integrated to give the orientation estimate. A weighting is also applied to the accelerometer correction in inverse proportion to the difference between the accelerometer reading and the expected value due to gravity. This is because periods of rapid motion when the acceleration vector includes significant linear accelerations of the IMMU body frame make the gravity vector less useful as a reference.

Internally the AHRS in the IMMU firmware computes orientation as a rotation matrix, however a quaternion representation of orientation is used in the state vector of the EKF described in Section 3.11. A review of several methods used to describe rotations, including Euler angles, quaternions and rotation matrices, as well as ways to convert between these representations can be found in [45]. A rotation matrix R can be used to construct a quaternion q according to Equation 3.18 [45], where R_{xy} indicates the matrix entry at row x and column y.

$$q(R) = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} 0.5\sqrt{1 + R_{00} + R_{11} + R_{22}} \\ \frac{R_{21} - R_{12}}{4 * q_w} \\ \frac{R_{02} - R_{20}}{4 * q_w} \\ \frac{R_{10} - R_{01}}{4 * q_w} \end{bmatrix} \tag{3.18}$$

The AHRS orientations are asynchronous to the camera images so the orientation measurements before and after are spherically interpolated to find the orientation q_{im} at the time of the image frame. Where q_0 is quaternion at time t_0 before an image frame at time t_{im}, and q_1 is the quaternion afterwards at time t_1, the spherical interpolation is performed in Equation 3.19 by defining the interpolation ratio r and intermediary values θ_0 and q_2 [45].

$$\begin{aligned} r &= \frac{t_1 - t_{VO}}{t_1 - t_0} \\ \theta_0 &= \sin^{-1}(q_0 \cdot q_1) \\ q_2 &= ||(q_1 - ((q_0 \cdot q_1) \times q_0)|| \\ q_{AHRS} &= q_0 \cos(r\theta_0) + q_2 \sin(r\theta_0) \end{aligned} \tag{3.19}$$

Using the relative rotation between the IMMU and camera ${}^{C}q_{IMMU}$ found with the toolbox of Lobo and Dias [46], described in more detail in Section 4.3.3, the AHRS rotations can be rotated into the camera frame by quaternion multiplication, to create the output q_{AHRS} as shown in Equation 3.20. Here $\otimes$ is used to denote the quaternion multiplication operator.

$$q_{AHRS} = {}^{C}q_{IMMU} \otimes q_{im} \tag{3.20}$$

3.11 Orientation Fusion

The selected odometry incremental pose estimates T_{VO} are now fused with the orientation reading q_{AHRS} provided by the AHRS using an EKF. The basic equations of the EKF can be found in [47]. The state vector $\mathbf{x} = [q, t]$ comprises a quaternion orientation $q = (q_w, q_x, q_y, q_z)$ and a translation $t = (t_x, t_y, t_z)$. A quaternion representation of rotation is used due to its avoidance of the singularity problem encountered by Euler angles and its increased computational and memory efficiency compared to the matrix multiplication required for rotation matrices [45].

The first step in the Kalman Filter is to obtain an *a priori* prediction $\hat{\mathbf{x}}_k^-$ of the current state at time k using the previous state estimate $\hat{\mathbf{x}}_{k-1}$, any known inputs at the previous time u_{k-1} and a dynamic model $f(x_{k-1}, u_{k-1})$ which describes how the system evolves over time. In this thesis, the prediction is provided by taking the odometry relative pose change since the last image frame T_{VO} and representing it as a quaternion and translation vector q_u and t_u respectively, using Equation 3.18 to convert the rotation matrix component of T_{VO} to a quaternion. The prediction equation compounds the quaternion component and rotates the translation component into the world frame before adding it to the previous position as shown in Equation 3.21.

$$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}) = \begin{bmatrix} q_{k-1} \otimes q_u \\ t_{k-1} + q_{k-1} \otimes t_u \otimes q_{k-1}^{-1} \end{bmatrix} \tag{3.21}$$

The next step is to estimate the *a priori* state covariance $\mathbf{P}_k^-$ using the previous state covariance $\mathbf{P}_{k-1}$ and the process noise covariance matrix $\mathbf{Q}$ as per Equation 3.22. For the regular Kalman Filter, propagation of the state covariance requires the dynamic model matrix, for the EKF however the state covariance is propagated using the Jacobian, $\mathbf{A}$, of the prediction function. Equation 3.23 gives the exact form of $\mathbf{A}$. Here $\mathbf{0}_{x,y}$ and $\mathbf{I}_{x,y}$ denote the zero and identity matrices respectively of size x rows by y columns.

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^{-1} + \mathbf{Q} \tag{3.22}$$

$$\mathbf{A} = \frac{df}{d\mathbf{x}} = \begin{bmatrix} q_w & -q_x & -q_y & -q_z & \\ q_x & q_w & q_z & -q_y & \mathbf{0}_{4,3} \\ q_y & -q_z & q_w & q_x & \\ q_z & q_y & -q_x & q_w & \\ & \mathbf{0}_{3,4} & & & \mathbf{I}_{3,3} \end{bmatrix} \tag{3.23}$$

Having previously pre-computed the AHRS measurement to a quaternion, the matrix $\mathbf{H}$ which maps the predicted state $\hat{\mathbf{x}}_{k-1}^{-}$ to a predicted measurement $\hat{z}_k$ simply selects the quaternion component of the state vector.

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{4,4} & \mathbf{0}_{3,4} \end{bmatrix} \tag{3.24}$$

$$\hat{z}_k = \mathbf{H}\hat{\mathbf{x}}_{k-1}^{-} \tag{3.25}$$

With this measurement prediction and the actual measurement quaternion $z_k = q_{AHRS}$ the updated *posteriori* state estimate $\hat{\mathbf{x}}_k^{+}$ is computed by first determining the Kalman Gain term $\mathbf{K}_k$ for the current step, using the measurement noise covariance matrix $\mathbf{R}$. The Kalman Gain is also used to find the *posteriori* state covariance $\mathbf{P}_k^{+}$.

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_k^{-}\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^{-}\mathbf{H}^T + \mathbf{R}_k)^{-1} \\ \hat{\mathbf{x}}_k^{+} &= \hat{\mathbf{x}}_k^{-} + \mathbf{K}_k(z_k - \hat{z}_k) \\ \mathbf{P}_k^{+} &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^{-} \end{aligned} \tag{3.26}$$

At the beginning of an odometry experiment the state vector is initialised using a location of $(0, 0, 0)$, representing the camera being at the origin of its path. The orientation is initialised using the AHRS output after one second, allowing it some time to initialise. The state covariance is initialised only along the diagonal elements - that is ignoring any cross-coupling of the covariances between the state vector elements. Values of 1×10^{-6} are used for the initial covariance of the translational components, representing high confidence that the initial camera position is at the origin. The rotational covariance components however are initialised with a much larger covariance of 0.01 to represent the uncertainty in the initial AHRS orientation estimate.

Values for the process noise $\mathbf{Q}$ and the measurement noise $\mathbf{R}$ were selected in proportion to the noise in the odometry prediction and AHRS updates respectively, with relative scale intended to reflect the trust in the measurement. The translational components of the odometry process noise $\mathbf{Q}$ were assigned a covariance 0.0001, a conservative approximation of the noise in the frame to frame odometry translation estimate while stationary. A covariance noise of 0.0001 was also used for the AHRS orientation measurement, while 0.001 was used for the odometry quaternion rotation component to indicate increased trust in the AHRS compared to potential cumulative drift errors in the odometry orientation estimate.

The output $\hat{\mathbf{x}}_k{}^{+}$ of each cycle of the Kalman Filter is the fused estimate of camera position relative to the starting location and orientation related to the NED frame.

Chapter 4

Hardware Implementation

This chapter describes the hardware implementation used in testing the odometry system. First an overview is provided showing the complete assembly used in testing, and the physical locations and interconnections of components. Then each component is discussed in more detail, and finally the calibration procedures for hardware components are described.

4.1 Hardware Overview

The hardware setup is intended to allow RGB-D cameras to be used outdoors in direct sunlight. The reliance of RGB-D cameras on IR light for sensing depth has previously limited their use to underground or indoor applications, as natural IR from sunlight can corrupt the depth readings. To overcome this problem, the system setup shown in Figure 4.1 uses a truncated pyramid, or frustum, shape to shield the Kinect field of view from direct sunlight. The Kinect faces the ground, looking through the opening at the top of the frustum.

In order to be portable for long distance outdoor tests the frustum shield is mounted to a hand-trolley. An extended handle allows for a comfortable walking position and counterbalancing the frustum weight at the front of the trolley are a laptop logging data for later offline processing, and the battery pack providing power for the Kinect.

4.2 Hardware Components

All the hardware components related to data collection are secured to a datachable mounting plate labelled 'Kinect mounting plate' in Figure 4.1. A schematic of the Kinect mounting plate is shown in Figure 4.2. By having all components secured to the mounting plate, it can be seperated from the frustum for hand-held trials whilst still maintaining all fixed translations and rotations between components. The hardware components include the

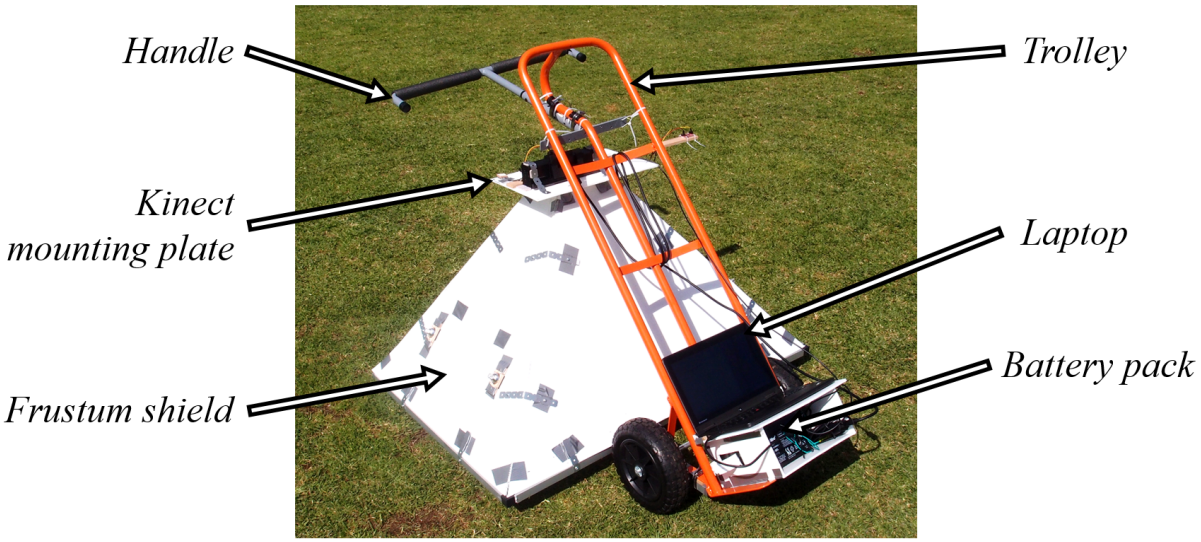


Figure 4.1: Complete system assembly

following:

- Kinect for Windows version 2 RGB-D camera
- Sparkfun Razor IMMU with three axis accelerometers, magnetometers and gyroscopes
- Venus GPS unit and patch antenna
- Arduino Mega 2560 microcontroller

The Venus GPS unit provides a reference for the path travelled in long-distance outdoor tests, while the Arduino collects and timestamps the GPS and IMMU data with a timestamp synchronised with the host laptop. The Arduino and the Kinect both pass data to the laptop via USB connections. The USB 2.0 connection from the Arduino to the laptop also supplies the Arduino and its attached components with power, while the the high-speed USB 3.0 connection to the Kinect is data-only.

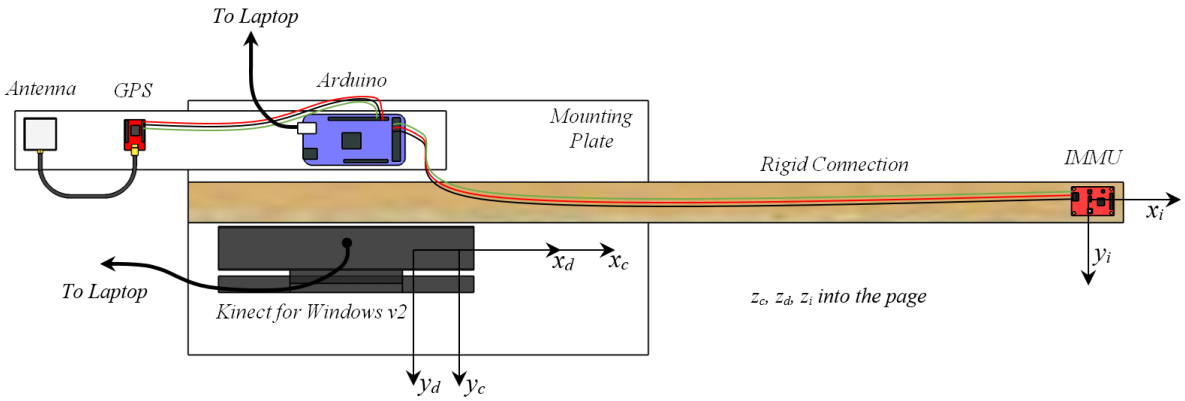


Figure 4.2: Hardware components

4.2.1 Kinect for Windows version 2 Camera

The first Kinect sensor was released towards the end of 2010 with the intention of providing a natural user interface and full body video game interactivity for the Microsoft Xbox 360 games console. For its price point it was unique in including both a colour camera and a depth camera which triangulated range from a random pattern of IR dots projected onto the scene using the structured light technique [35]. When a developer version for PC was released in 2012, called the Kinect for Windows, its RGB-D sensing capabilities gave it many applications in the field of computer vision, including 3D mapping, odometry and obstacle avoidance [35], [48], [49].

In late 2013 a new Kinect called the Kinect for Xbox One was released alongside the new Xbox One console, and in July 2014 it enjoyed a public PC developer version release under the name Kinect for Windows version 2. The Windows versions of the Kinect are of greater interest to robotics than the Xbox versions as custom applications can be designed for them using any computer hardware running Windows. Hereafter the Kinect for Windows version 2 shall be referred to as either the 'new Kinect' or the 'Kinect' while the original Kinect for Windows will be called the 'old Kinect'.

Table 4.1: Comparison of Kinect version specifications, [10], [50], [51]

	New Kinect	Old Kinect
Depth Sensing Method	Time of Flight	Structured Light
IR Projector	High frequency flashes	220×170 dot grid
Operation Range (m)	0.5-4.5	0.5-4.5
Depth Image Resolution (px)	512×424	640×480
Depth Image FOV (°)	70×60	57×43
Colour Image Resolution (px)	1920×1080	640×480
Colour Image FOV (°)	84×54	57×43
Power Draw (W at 12V)	32	13

The new Kinect offers a number of significant advances over the original. See Table 4.1 for a comparison of their specifications, the most noticeable of which include the overall increase in field of view, the significantly higher colour resolution, a change in the depth sensing method, and a reduction in depth image resolution. While the decrease in depth resolution at first appears to be a step backwards, the operation principle of the old Kinect must be taken into account. The old Kinect calculated depth by projecting a 220×170 grid of IR dots and measuring their location to infer range [10]. This means at least two dots must fall on any surface to identify their location in the dot pattern and produce a range reading, and then the valid readings are interpolated to fill up the 640×480 output image. The actual depth information available is thus constrained to a maximum, best-case resolution of 220×170 and has experimentally been found to be closer to 118 pixels in width resolution, due to the requirement of placing multiple dots on a surface to determine a range [10].

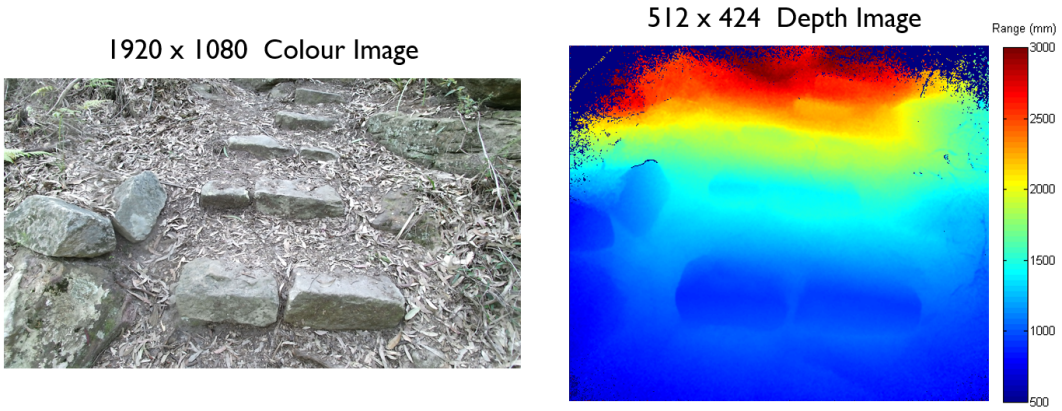


Figure 4.3: Example colour and depth images produced by the Kinect

Figure 4.3 shows example outputs from the Kinect camera, including a full high definition colour picture of some stone steps, and a colour-mapped depth image of the same scene. The difference in field of view can be noted by observing how the rocks on the left edge are cut off in the depth image. Figure 4.4 shows an example of a coloured point cloud formed by mapping the depth image into the colour space and projecting it to 3D.

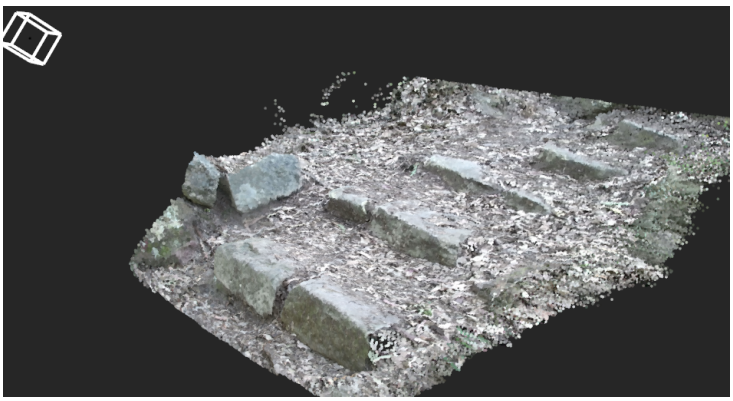


Figure 4.4: Coloured point cloud created by combining colour and depth images

The new Kinect achieves true 512×424 resolution in its depth image by using the TOF measurement principle. A TOF camera operates by emitting pulses of light and measuring the time taken for the signal to return. However given the high speed of light and short distances involved, accurate measurement of the time delay is difficult and instead a phase change in the returning light signal can be used to infer the travel time.

Although the exact implementation details of the new Kinect sensor have not been publicly released at this time, existing TOF cameras pulse IR light at a high, known frequency [34]. Instead of collecting the returning photons as charge accumulating on a standard Charge Coupled Device (CCD)/Ceramic Metal Oxide Semiconductor (CMOS) imaging chip, there is a special CMOS chip where each pixel comprises two sensor gates. These sensor gates have their photo-detection sensitivity controlled by the source of the IR transmitting signal, one gate by the original signal, the other 180° out of phase. How

much charge collects at each gate depends on the gate voltage when photons arrive, and so the ratio in voltage between the two gates provides a measure of the phase with which the photons arrived which is independent of the actual return intensity. This performs an automatic, hardware-based mixing of the transmitted signal with the received signal to estimate the phase change. With the frequency known, the phase change can be used to infer a TOF and thus a range at each pixel [34].

This process also affords better depth resolutions near the maximum range, as the previous structured light technique relied on a high-resolution IR camera to find the dots and a lookup table to determine range, whereas the TOF technique can measure phase and therefore range with approximately consistent precision up to the maximum range [10], [34].

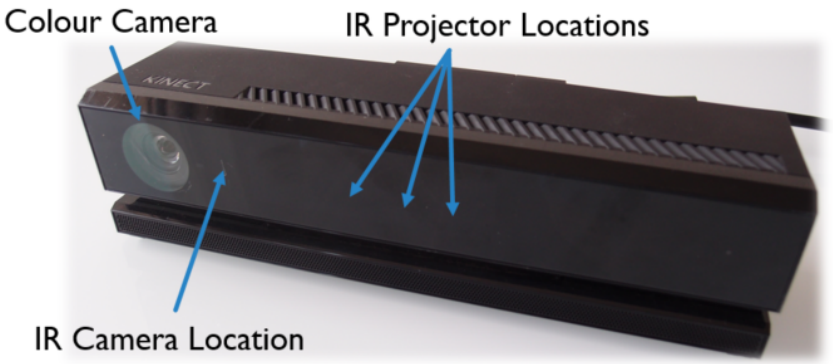


Figure 4.5: Kinect camera components

The IR projectors and camera in the Kinect are not visible when the camera is switched off, although their approximate location is indicated in Figure 4.5. When the camera is operating the IR projectors can be observed by the human eye as three points of blurred red light. In terms of physical design the Kinect sensor is bulkier than the original. The body of the sensor measures 250mm long, 66mm deep and 43mm high when measured with digital callipers.

A schematic of the Kinect is shown in Figure 4.6, including coordinate frames, where red is for the x axis, green for the y axis and blue the z axis. The coordinate frames in the Kinect body are denoted as follows:

- K: Kinect body frame. Attached to the back corner of the camera. Oriented to align with a NED coordinate system when the Kinect is ground-facing and its top surface points North.
- C: Colour camera frame. Centred at the colour camera image centre, the axes directions follow the convention of z pointing along the camera axis, y downwards and x by right-handedness.
- D: Depth/IR camera frame. Axes directions as above.

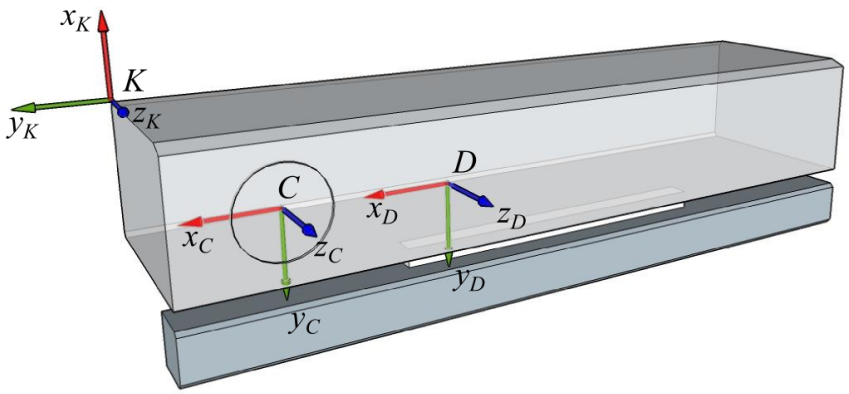


Figure 4.6: Kinect coordinate frames

4.2.1.1 Power for the Kinect

Power for the Kinect used in the frustum configuration is supplied by a 12V, 26Ah gel battery, sufficient to power the Kinect at its 2.67A current draw for just under ten hours. Powering the Kinect with a battery, given it was designed for 240V mains supply, requires cutting the custom coaxial power cable between the transformer and the Kinect adapter and replacing it with a connection to the battery.

When used hand-held, the Kinect is powered by 10 rechargable 1.2V AA batteries in series, for a combined voltage of 12V and with a capacity of 2.5Ah. This is sufficient to power the Kinect for just under 1 hour.

4.2.2 Frustum Shield

As described in Section 4.2.1 the Kinect depth sensor relies on receiving a return from the pulsed IR to measure range. IR sources are uncommon indoors so the camera can operate effectively, however outdoors in daylight the sun provides large amounts of background IR radiation which interferes with the depth readings.

Although the operation details of the new Kinect are not yet available, the old Kinect used IR at a wavelength of 850nm, and could be disrupted by intensities of up to 6-7W/m^2 of IR noise, while direct sunlight can have IR intensities in the 800-900nm band of up to 75W/m^2 [10]. To use the depth camera outdoors it is necessary to provide some way to filter out the IR in sunlight.

An optical filter is one possible solution, however this is very difficult to implement precisely, especially without knowing the desired wavelength of IR that should be allowed through to the camera. Instead the method used, as introduced in Section 4.1, is to physically protect the Kinect camera field of view from sunlight with a frustum-shaped shield.

The frustum dimensions were chosen as a compromise between not interfering with the camera field of view, being small enough to manoeuvre and transport easily, and putting the camera high enough so that parts of the ground observed would not exceed the Kinect minimum range of 0.5m.

To this end a height of 640mm was selected with slopes of the frustum sides set at 74°×60°. These slopes are wider than the depth camera field of view, but slightly obstruct the edges of the wide-angle colour camera. Any wider slopes would have reduced manoeuvrability, and since no depth data is available at those edges, the odometry system would not have been able to make use of it in any case. With a wide opening of 180mm×130mm to provide stability for the mounting plate the resulting dimensions to the nearest mm of the frustum are shown in Figure 4.7.

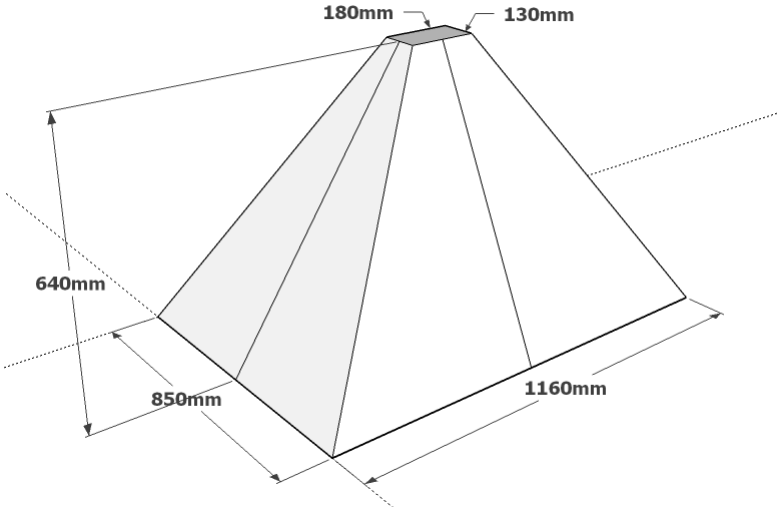


Figure 4.7: Frustum design schematic

The frustum is constructed out of 5mm thick translucent corflute, allowing some transmission of visible light for the colour camera but blocking sufficient IR, which is a poor penetrator, for the depth sensor to operate effectively.

4.2.3 Razor IMMU

The Razor IMMU is a hobbyist unit containing three axis accelerometers, gyroscopes and magnetometers. It includes an onboard microprocessor for applying calibration parameters to its raw sensor data, converting floating point readings to text for output over serial communications, and running an AHRS onboard. Communications interface is by a 3.3V serial connection.

The Razor IMMU measures 41.5mm long by 28.5mm wide. A schematic diagram constructed from measurements made of the device with digital calipers is shown in Figure 4.8. Four coordinate frames are marked on the IMMU, with red for the x axis, green

for the y axis and blue for the z axis. The coordinate frames are denoted A for the accelerometer, M for the magnetometer, G for the gyroscope and R for the Razor IMMU body frame. Note that all frame axes shown are aligned with the internal sensor axes directions.

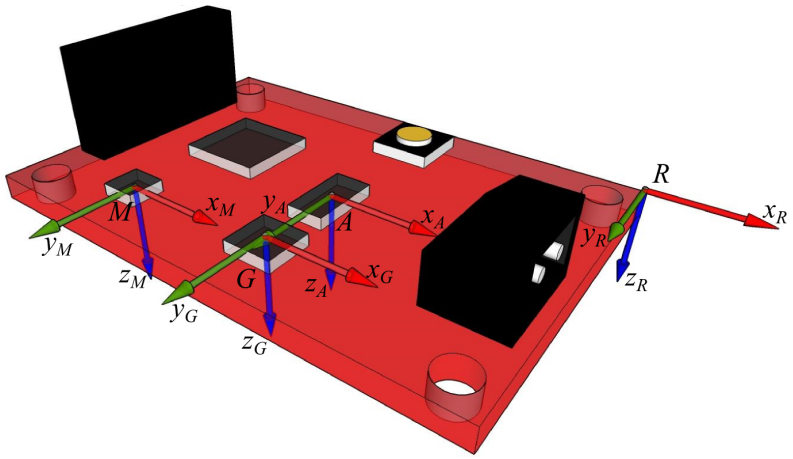


Figure 4.8: IMMU coordinate frames

See Table 4.2 for a list of key specifications of the Razor IMMU. Note that in this case the units 'g' and 'mg' refer to standard accelerations due to gravity and milli-gravities respectively, while 'G' and 'mG' are magnetic field units Gauss and milliGauss. According to the datasheets of the accelerometer and magnetometer components, these resolutions are sufficient to provide 1-2° pointing accuracy in roll and pitch, and 1-2° accuracy in yaw when used in an AHRS.

Table 4.2: IMMU sensor specifications [52], [53], [54]

	A	M	G
Max Data Rate	100Hz	160Hz	100Hz
Sensor Limits	±16g	± 8G	±2000°/sec
Bit Resolution	13	12	16
Physical Resolution	±4mg	±2mG	±0.06°/sec

While the individual sensor components can achieve update rates of greater than 100Hz, the Sparkun IMMU firmware documentation recommends a 50Hz update rate be maintained for stability. Readings for the three axis magnetometer, accelerometer and gyroscope are thus output fifty times a second from the IMMU, with internal calibration compensation applied as described in the IMMU sensor calibration Section 4.3.2.

4.2.4 Venus GPS

The Venus GPS is a compact GPS receiver unit which requires an external antenna. It has a maximum update rate of 10Hz and operates on 3.3V power. It outputs full National

Marine Electronics Association (NMEA) style text output, including measured latitude, longitude and velocity, a list of satellites in view, figures for the dilution of precision due to satellite location, and the current Coordinated Universal Time.

4.2.5 Arduino Mega 2560 Microcontroller

Due to a limited number of USB ports on the laptop collecting data and the need for 3.3V power and communications to the GPS and IMMU, an Arduino Mega 2560 microcontroller was used to collect GPS and IMMU data and pass them back to the laptop. An advantage of using a microcontroller compared to a USB hub is that the Arduino can be dedicated to the process of listening for new data and applying a timestamp as soon as a new message arrives. The laptop by contrast uses most of its processing time to save Kinect image data and cannot process IMMU/GPS serial data immediately, leaving latency in recording the arrival of data. Latency between the Arduino and the sensors by contrast is minimal as the Arduino main loop can be dedicated to listening for new serial data.

However, in order to use the Arduino to capture and timestamp the two data streams its clock must be synchronised with that of the laptop host computer. Then when the first byte of a new data message arrives at the Arduino it can be timestamped with the synchronised timestamp. The synchronisation protocol works as follows:

1. The host computer initiates the protocol by requesting a reset of the Arduino by sending the letter 'R'
2. The Arduino performs a soft reset, clearing all variables in storage, setting its clock back to zero and returning to the start of its program
3. The Arduino prints an ASCII enquiry ENQ, 0x05, requesting the synchronisation
4. The host replies with an 'S' character, confirming the synchronisation, while saving its internal clock time in milliseconds since the host was powered on
5. The Arduino replies with the current time in milliseconds as measured with its clock since the reset
6. The host computes the offset between the time it previously saved and the received Arduino time
7. This offset is returned to the Arduino
8. The Arduino stores this offset to apply to all future timestamps and then ends the protocol by sending an ASCII acknowledgement ACK, 0x06

Any errors in this synchronisation are due only to the delays in communicating the 'S' character which confirms the synchronisation, the Arduino reading it and then saving its

own internal time. At the baud rate of 1,000,000 bits per second, the highest recommended stable rate for the Arduino, this delay is appreciably small.

In fact, this synchronisation is guaranteed to be accurate to within 2ms, as the Arduino time received in step 5 has been observed to vary between 1-2ms since reset. As any difference between the synchronised clocks can only occur between steps 4 and 5 when the two devices save their current times, and the Arduino time since reset in step 2 is 1-2ms, the synchronisation difference must be less than this.

After synchronisation the Arduino receives IMMU and GPS data, prepends it with a flag indicating the sensor source and millisecond precision synchronised timestamps, and sends it back to the laptop over 5V serial at a baud rate of 1,000,000.

4.3 Hardware Calibration

The sensors used in this thesis all require calibration in order to be used effectively. Camera focal length and image centre parameters are required for projecting the Kinect data, while the IMMU needs calibration to map its sensor values to physical units and to correct for hardware variations. Furthermore, the metal frame supporting the frustum shield introduced localised magnetic field distortions which affected the magnetometer readings and had to be calibrated using a separate process.

4.3.1 Kinect Camera Calibration

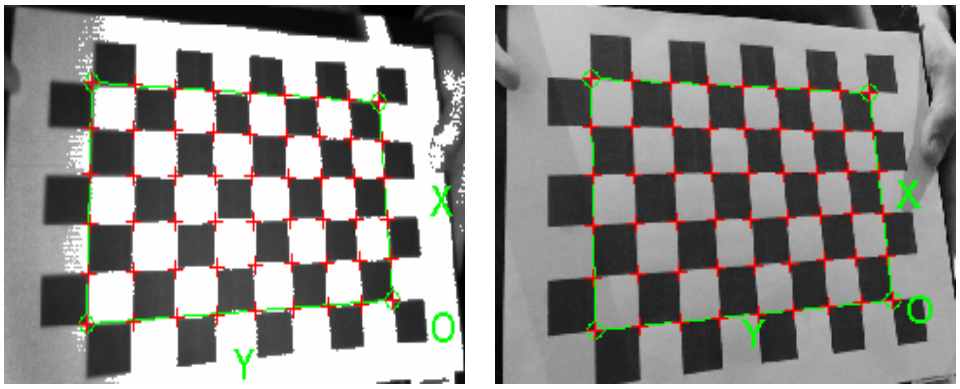
In order to project the depth image to 3D points the intrinsic camera parameters of the depth camera are required. Mapping these depth points to the colour camera frame also requires the extrinsic parameters of the relative location and orientation between the two cameras and the colour camera intrinsics. In the context of camera calibration, intrinsic parameters refers to the four values which define the pinhole camera model; the focal length in the x and y directions (f_x, f_y), and the coordinates of the camera centre (the effective pinhole location) in the camera image (c_x, c_y).

Camera calibration is the process of determining these intrinsic parameters as well as quantifying any distortion effects introduced by the camera lens. Stereo calibration is the process of finding the extrinsic parameters between two cameras which are fixed in relation to each other.

Bouguet's camera calibration toolbox for Matlab [36] was used to perform calibration as it provides a simple and intuitive interface and requires very little additional hardware. The calibration routine requires a number of images to be taken of a planar chessboard target at different orientations. Either the camera or the target can be moved, and more than 20 images are recommended [36]. In each image the four outermost square corners of the chessboard must be manually identified to define the grid, and then the toolbox

guesses the remaining corner locations and asks for verification. The known physical length of the square edges provides scale information. Example images of a chessboard target observed by the Kinect colour and IR cameras along with the detected corners are shown in Figure 4.9.

The set of images and detected corner locations forms a high-dimensional problem of estimating not only the constant camera intrinsic parameters but also the pose of the camera with respect to the target in each image. This nonlinear problem is solved by the toolbox using a gradient descent algorithm to find the most likely set of poses and camera parameters which minimises the error between the predicted and measured corner locations.



(a) IR camera (b) Colour camera

Figure 4.9: Kinect camera calibration target

Stereo calibration is a simple extension to this procedure where synchronised images are taken of the same target. Once the set of poses and intrinsics has been estimated for each camera individually, another nonlinear problem is formed to find the relative pose between the two cameras given the set of poses calculated for each camera relative to the common target. Again, a gradient descent algorithm is used to optimise the relative translation and rotation between the two sensors.

Although the IR and colour cameras in the Kinect are very different in nature, they still are in a stereo camera configuration. To calibrate both intrinsically and extrinsically, the camera was placed on a stable surface and shown a flat chessboard target with 25mm squares at 24 different orientations. At each orientation, a synchronised image from the colour and IR camera was collected. The results of the intrinsic calibration applied to the two sets are shown in Table 4.3 with units in pixels.

The homogeneous transformation matrix relating the two camera frames is given in Equation 4.1 where ${}^{D}T_{C}$ is the pose of the colour camera frame with respect to the depth / IR camera frame. The rotation between the two cameras is close to the identity matrix, as expected, while the translation component indicates a 5cm displacement in the x direction and a small z direction shift.

Table 4.3: Kinect individual camera calibration results

	Colour Camera	IR Camera
f_x	1064.00	361.41
f_y	1063.74	361.13
c_x	946.74	250.92
c_y	539.82	203.69

$$ {}^{D}T_{C} = \begin{bmatrix} 0.9998 & 0.0176 & 0.0080 & 0.0522 \\ -0.0175 & 0.9997 & -0.0154 & -0.0008 \\ -0.0082 & 0.0153 & 0.9998 & 0.0059 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix} \tag{4.1} $$

Although the exact location of the IR camera is not documented nor externally visible, this horizontal displacement was approximately confirmed by pointing a slender object at the Kinect and measuring the point at which it was in the centre of the live IR image stream.

One issue encountered with the IR camera calibration was a trade-off between detail and over-exposure. The closer the target is to the camera, the more accurately the pixel coordinates of the square corners can be located. However the IR camera observes the target using illumination from the pulsed IR projectors in the Kinect, which are designed to illuminate targets between 0.5m and 4.5m away. If the target was held too close to the camera the target image would experience washout, making the corners difficult to locate. An example of the IR washout is shown in Figure 4.10.

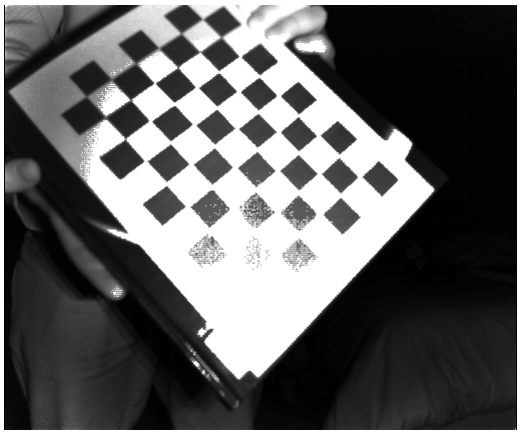


Figure 4.10: Calibration target too close to the IR projector causing washout

4.3.2 IMMU Sensor Calibration

The IMMU requires a separate calibration routine for each of its three sensors. Calibration is performed following the methods described in [55] as part of the documentation for the

Razor IMMU firmware.

4.3.2.1 Accelerometer Calibration

For the accelerometer, calibration is used to find the raw numeric sensor values which equate to ±1g of acceleration in each axis. This is accomplished by holding the sensor steady, pointing approximately downwards along one axis and moving it in small circles to find the maximum (or negative minimum) raw sensor output value. This process is assisted by the firmware calibration routine, which displays the maximum and minimum values recorded in each axis since the last reset command. This is repeated in both directions for each axis to find the values which correspond to the acceleration due to gravity. Results recorded for the Razor IMMU used in this thesis are shown in Table 4.4.

Table 4.4: Accelerometer calibration results

Axis	Min. Value	Max. Value
X	-263	242
Y	-244	264
Z	-288	225

These values are used to scale and centre the raw readings in each axis to calibrated values which range from ±256.0 corresponding to accelerations between ±1g. The Razor IMMU microprocessor performs the calculation in Equation 4.2 to scale the raw acceleration sensor value a_{raw} using the measured minimum a_{min} and maximum a_{max} calibration value in each axis to obtain the calibrated accelerometer reading a_{calib}. The first term in the equation centres the raw value, and the second performs a linear scaling.

$$a_{calib} = \left(a_{raw} - \frac{a_{max} + a_{min}}{2}\right)\left(\frac{256}{a_{max} - a_{min}}\right) \tag{4.2}$$

Calibrated accelerometer readings can then be converted to a physical acceleration reading $a_{ms^{-2}}$ in ms^{-2} given that 256 represents 1g acceleration. Using $g = 9.81ms^{-2}$:

$$a_{ms^{-2}} = 9.81 \times \frac{a_{calib}}{256} \tag{4.3}$$

4.3.2.2 Gyroscope Calibration

The calibration process for the gyroscope requires detection of any steady-state bias errors which may exist. This is performed by leaving the sensor stationary for a period of time and recording the average values from each gyroscope axis, again facilitated by the calibration firmware which outputs the average gyroscope values since the last reset. Results for the Razor IMMU are shown in Table 4.5.

Using the average value as a measure of the steady state bias, calibrated gyroscope readings g_{calib} are found by subtracting the average g_{avg} from the raw g_{raw} sensor value

Table 4.5: Gyroscope calibration results

Axis	Avg. Value
X	9.05
Y	84.52
Z	-5.41

in each axis.

$$g_{calib} = g_{raw} - g_{avg} \tag{4.4}$$

To convert the calibrated gyroscope readings to a real-world measurement in radians/sec the gyroscope gain is applied. The manufacturer specifies a gain of 0.06957 for converting to a rotation rate in degrees per second, thus to get radians per second we use:

$$g_{rad/sec} = g_{calib} \times 0.06957 \times \frac{\pi}{180} \tag{4.5}$$

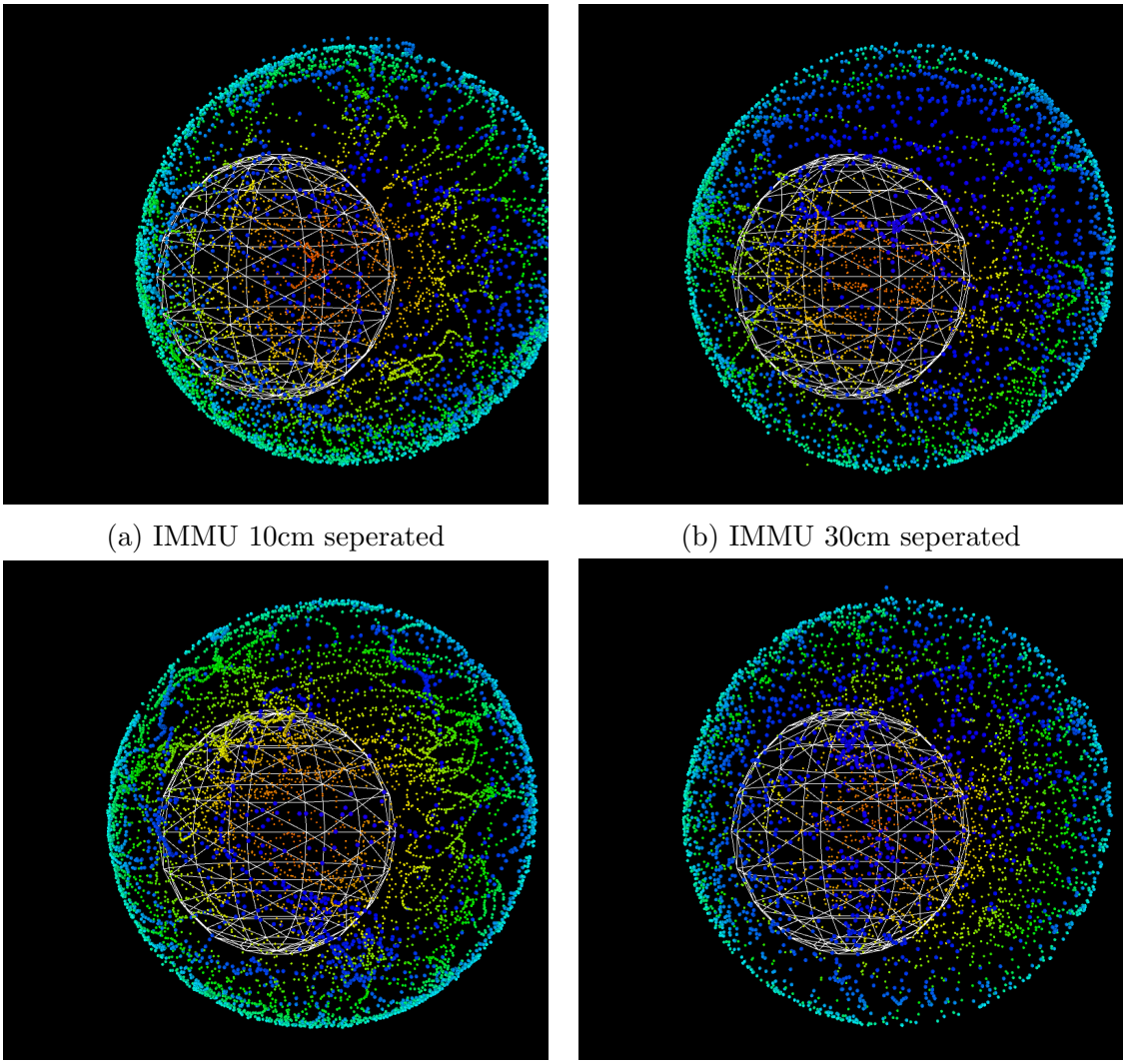
4.3.2.3 Magnetometer Calibration

Two methods of magnetometer calibration are described in [55]; simple and extended. Simple calibration attempts to find the maximum and minimum sensor values for the magnetic field vector along each axis. This functions similarly to the accelerometer calibration, except the IMMU is pointed in the magnetic field direction rather than downwards, and the maximum and minimum values are used to scale the calibrated output to fall between ± 100 for respectively pointing with or opposite to the local field vector. However this assumes a linear operation of the magnetometer which is unlikely in the presence of electromagnetic distortion effects like metal or other electronic components.

The extended calibration routine is a standalone program included with the IMMU firmware intended to account for the local magnetic field distortion caused by any electronic or metallic items to which the IMMU will be attached. The IMMU and all attached metallic and electronic components are rotated in all directions to map the magnetic field and to create a model for the local distortion using an ellipsoid fit. With this fit the IMMU is effectively able to measure the interaction between the Earth's magnetic field and the local distortion components in order to counteract them.

Tests were conducted with the IMMU alone in free space, and also with the IMMU secured to the Kinect mounting plate (Figure 4.2) at three different distances from the camera of 10cm, 30cm and 100cm. The magnetic field mapping patterns from these tests are shown in Figure 4.11, and the numerical results from the ellipsoid fit are given in Table 4.6.

As Table 4.6 and Figure 4.11 illustrate, at a distance of 10cm the magnetic field is significantly distorted compared to the free-space case, with a large change in the centre



(a) IMMU 10cm seperated

(b) IMMU 30cm seperated

(c) IMMU 100cm seperated

(d) IMMU in free space

Figure 4.11: Extended magnetomer calibration results for different IMMU configurations

location in the x and y directions. By 30cm the distortion is greatly reduced although still present, while at 100cm the ellipsoid centre is approximately equal to the free space case within noise levels. The change in the ellipsoid transform is more difficult to observe, but there are noticeable differences in the coefficients between the free space and 10cm cases, which are almost eliminated at 100cm.

It is for this reason that the IMMU is mounted at the end of such a long rigid connection to the Kinect mounting plate, to maintain distance from the camera's magnetic field distortions. With a distance of 100cm, the extended calibration result is sufficient for accurate use of the IMMU magnetometer when using the Kinect mounting plate assembly alone.

Table 4.6: IMMU extended magnetometer calibration results

Location	Elipsoid Centre	Elipsoid Transform
Free Space	$\begin{bmatrix}132.422 & -43.9921 & -24.6058\end{bmatrix}$	$\begin{bmatrix}0.93168 & 0.00724 & -0.00516\\ 0.00724 & 0.92812 & 0.02123\\ -0.00516 & 0.02123 & 0.99359\end{bmatrix}$
10cm	$\begin{bmatrix}221.689 & -58.2359 & -44.1699\end{bmatrix}$	$\begin{bmatrix}0.91737 & 0.01435 & -0.00251\\ 0.01435 & 0.93467 & 0.01393\\ -0.00251 & 0.01393 & 0.99702\end{bmatrix}$
30cm	$\begin{bmatrix}142.079 & -44.0382 & -25.9045\end{bmatrix}$	$\begin{bmatrix}0.93065 & 0.00976 & -0.00420\\ 0.00976 & 0.92588 & 0.02064\\ -0.00420 & 0.02064 & 0.99421\end{bmatrix}$
100cm	$\begin{bmatrix}130.224 & -42.7856 & -26.1298\end{bmatrix}$	$\begin{bmatrix}0.931767 & 0.00898 & -0.00324\\ 0.00898 & 0.92758 & 0.02005\\ -0.00324 & 0.020058 & 0.99443\end{bmatrix}$

4.3.2.4 Frustum Deviation Compensation

While the extended magnetometer calibration described in Section 4.3.2.3 accounts for magnetic interference from the Kinect camera, when used with the full frustum assembly shown in Figure 4.1 a new calibration issue for the magnetometer is introduced. The frustum is supported by aluminium tubing and the hand trolley is made out of steel as other materials are not sufficiently structurally robust for manoeuvring the frustum assembly over large distances of varying outdoor terrain. Even with the physical separation of the IMMU this quantity of metal is sufficient to cause magnetic field distortions. A simple test which demonstrates the distortion was performed by rotating the combined frustum and mounting plate assemblies through right angle rotations and observing the change in yaw from the AHRS, which should be 90° on each turn. The equations used to convert between the quaternion AHRS orientation q_{AHRS} and equivalent Roll Pitch Yaw (RPY) Euler angles can be found in [45]. The order of the Euler angle convention used is first roll about the x axis, then pitch about the y axis, followed by yaw about the z axis.

Table 4.7: AHRS yaw rotation nonlinearity due to the frustum frame

Right Angle Rotation	1	2	3	4
Measured Change in Yaw (°)	85	92	97	86

The results from this test are shown in Table 4.7. Given the magnetometer component datasheet claims pointing accuracy of 1-2° using the magnetic field vector, variations of ±7° compared to the true rotation indicate the existence of a nonlinear distortion.

Although the nature of the distortion can in theory be compensated for with extended magnetometer calibration, this was not practical for the frustum frame - the entire as-

Figure 4.12: GPS path for magnetic deviation estimation

sembly is too large to pick up and rotate in all directions while connected to the IMMU. Instead, the technique used to compensate for the frustum distortion is inspired by the practice of calculating magnetic deviation for the compass on sailing ships [56]. A large circle of 18 cones was laid out on flat ground, approximately 50m in diameter, and the entire frustum assembly was pushed in a star shape pattern back and forth between the cones. The path was tracked using GPS, shown overlaid on Google Earth imagery in Figure 4.12. The Kinect camera was switched on and the IMMU was used to compute AHRS orientations. Each cone was then moved by half the spacing around the circle and the test repeated. Note that some cones were visited twice before the path returned to the starting point. The output from the GPS from the first test is shown alongside the AHRS yaw, equivalent to a compass heading, plotted against time in Figure 4.13.

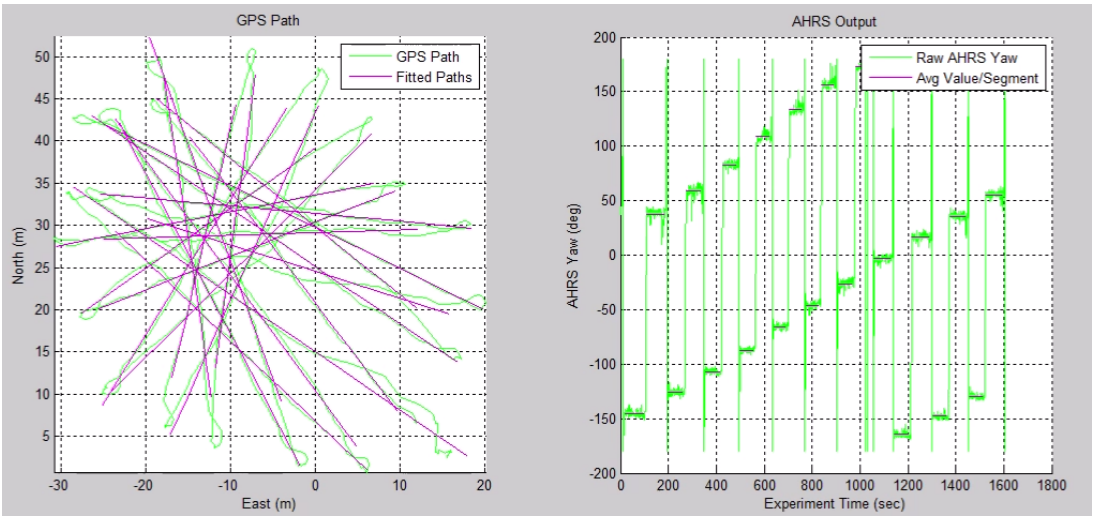


Figure 4.13: GPS path with fitted straight lines and average yaw readings for magnetic deviation estimation

To analyse this data, a rapid change in yaw corresponding to a direction change at each point of the star shape was used to automatically segment the yaw readings into sections corresponding to each straight line path between cones, which correspondingly segmented the synchronised GPS data. Lines were then fit to the GPS path on each straight line segment using a least-squares fit and the bearing of this line of best fit was used as the reference for the true heading of the path. The yaw reading across each path segment was averaged and the deviation was calculated, which is the difference between the GPS measurement of heading and the AHRS yaw reading. The deviation is plotted against AHRS yaw in Figure 4.14 using the data from both tests. With 36 data points a deviation was captured approximately every 10°.

The deviation is quite nonlinear as a function of yaw. Noise in the deviation made linear interpolation an unsuitable method for compensating the yaw, so a parabolic fit was applied using least squares to model the deviation function. The deviation varies from around -12° on a heading of due South, up to 2° on a heading of due North. To use this model to correct the magnetic distortion, the quaternion orientation outputs q_{AHRS} from the AHRS are first converted to RPY Euler angles. Then the magnetic deviation is estimated from the parabolic model and added to the yaw angle. The RPY angles are then converted back to a quaternion for fusion with the odometry estimate in the EKF.

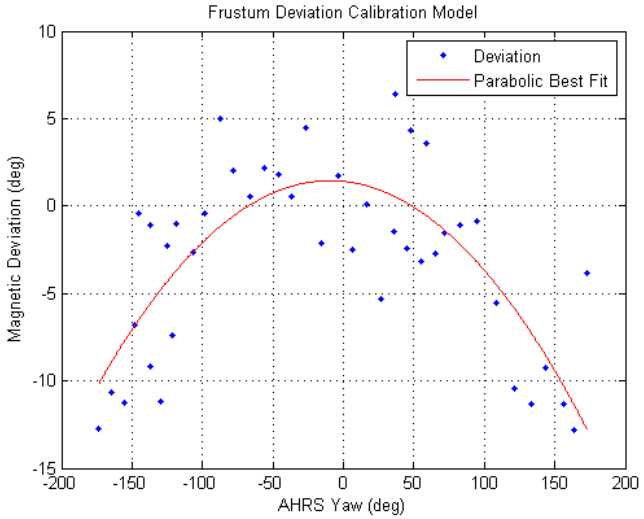


Figure 4.14: Magnetic deviation due to frustum frame

4.3.3 IMMU-Camera Calibration

In order to use the IMMU rotation measurements as a reference for the camera orientation, the relative rotation between the Kinect camera and the IMMU body frame ${}^{C}q_{IMMU}$ must be determined. As shown in Figure 4.2 the Kinect camera axes have been manually aligned with the IMMU body axes, however a precisely calibrated result is required. For this purpose there exists the IMU-Camera calibration toolbox of Lobo and Dias [46].

Table 4.8: Rotation between IMMU and Kinect camera

${}^{C}q_{IMMU}$	[-0.0229, -0.0036, -0.9994, -0.0253]
Roll Pitch Yaw (°)	[-3.091, 0.0459 , 3.1355]

The operation of this toolbox bears similarities to the stereo calibration toolbox described in Section 4.3.1, however instead of having the two sensors observe the same chessboard target, they both observe the local gravity vector from a number of different orientations.

For the Kinect, the local gravity vector is observed by pointing it at a flat, vertical chessboard target and finding corners. The IMU-Camera toolbox actually operates by first using Bouguet's toolbox to estimate the camera poses with respect to the target, before adding the constraint that the chessboard target is vertical. The set of camera poses generated is compared against a set of roll and pitch orientations estimated from an IMU or an IMMU by synchronously sampling the acceleration due to gravity vector when the image was being taken. A gradient descent based nonlinear optimiser is then used to find the relative rotation which minimises the difference between the gravity vectors observed by the two sensors.

The output of the calibration is given in Table 4.8, where ${}^{C}q_{IMMU}$ is the quaternion rotation between the IMMU and the camera, and the equivalent RPY Euler angles are also shown. As expected, there is only a small rotation between the Kinect and the IMMU, with their axes directions deviating by only 3°.

Chapter 5

Experimental Results

In this chapter experimental results will be presented which demonstrate the capabilities of the system. The first set of experiments use the frustum configuration shown in Figure 4.1 with the camera ground facing, protected from direct sunlight by the frustum shield. For these tests it is possible to evaluate a ground truth using direct physical measurements of a marked path to evaluate the system accuracy. Failure cases will also be presented as well as an analysis of the benefit provided by different odometry processing steps including SBA, fusion with the AHRS and the compass calibration.

The second experiment configuration presents a more challenging dataset of hand-held camera motion. The Kinect mounting plate shown in Figure 4.2 is seperated from the frustum and carried along an outdoor bush track in order to provide an environment with greater depth features which makes full use of the RGB-D camera capabilities.

Finally, timing statistics for the odometry processing stages will be presented and analysed, comparing existing serial CPU based implementations with the GPU parallel implementations steps described in Chapter 3.

5.1 Ground Facing Camera

Two ground facing tests will be presented. The first was performed at Tunks Park sports oval on a terrain of flat grass to evaluate accuracy, the second along a road in Harold Reid Reserve to evaluate terrain effects.

5.1.1 Tunks Park Oval Test

The path followed in the Tunks Park oval test is shown in Figure 5.1 overlaid on Google Maps imagery of the terrain. The path consisted of two stages, a pre-defined track of straight lines in the shape of a backwards 'E' marked out with cones, and a random meandering walk with loops and spirals. A speed of 50-60cm/sec was maintained for the duration of the approximately 700m path travelled by the camera.

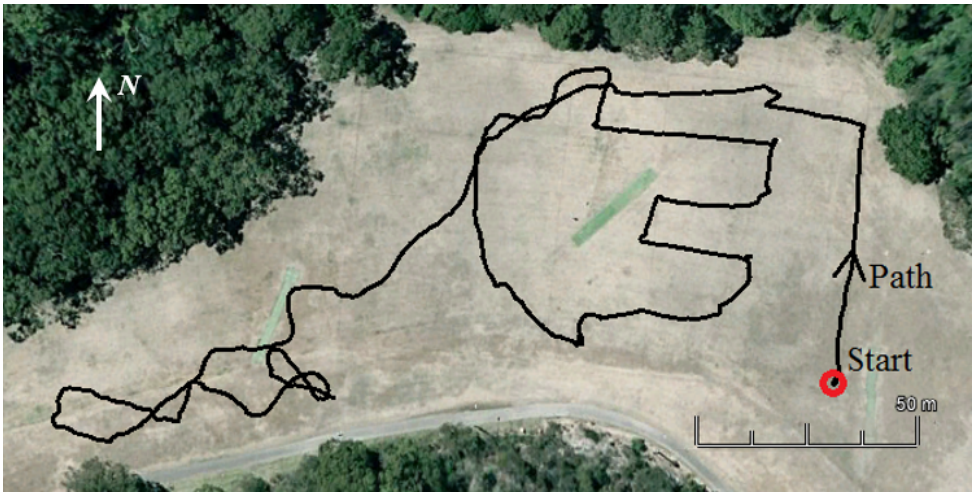


Figure 5.1: GPS path of Tunks Park dataset overlaid on Google Maps

Figure 5.2 shows an example greyscale image of what the Kinect colour camera observes; patches of dark in between blades of grass provide many strong responses to the Star keypoint detector but the short grass is mostly featureless from a depth perspective. As such, the SBA odometry estimate was chosen over the ICP estimate for every RGB-D image frame in the experiment. Note that the colour camera, due to its wide field of view, can see the sides of the frustum shield. This has no effect on the odometry system as the depth camera has a narrower field of view which does not include these sides. The lack of depth data precludes any 3D keypoints being found on the frustum frame.

Figure 5.2: Kinect colour camera view of Tunks Park terrain

The first result to consider is the SBA estimate of the camera trajectory formed by concatenating the frame to frame pose estimates T_{SBA}, shown in Figure 5.3 with the GPS path for comparison. Note that the GPS and SBA start points are not exactly the same because the start point latitude and longitude were manually determined using physical landmarks from the Google Earth imagery. This was required due to noise in the GPS position, particularly towards the start of operation when the GPS unit is still acquiring

satellite signals.

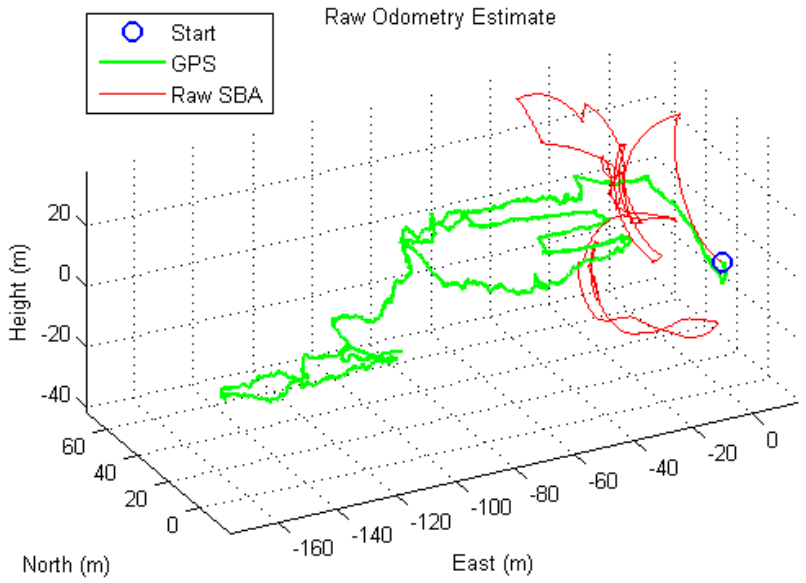


Figure 5.3: Raw odometry estimate without AHRS orientation fusion

Fusion with the AHRS orientation was not been performed to create Figure 5.3, and only the initial camera orientation was set using the AHRS so that the SBA and GPS paths would be initially aligned. While the GPS path is flat as expected, the SBA estimate rapidly spirals up into the air, completely failing to track the true path due to cumulative rotation errors. This is particularly severe given the length of the path - for the first 15m the SBA path is within 1m of the GPS path, but thereafter starts to drift away. This result is indicative of the common problem faced by long range odometry systems where constraining orientation drift is crucial to long range accuracy, and this justifies the inclusion of the AHRS.

The AHRS output in terms of roll about the x axis, pitch about the y axis and yaw about the z axis is given in Figure 5.4 plotted against the experiment time. The roll and pitch measured using the gravity vector is typically a few degrees from zero, with high-frequency noise present due to the bouncing of the frustum trolley over the ground resulting in small and fast orientation changes. The yaw readings shown are the direct output from the AHRS with only internal magnetic calibration routines applied, not the full frustum magnetic deviation calibration. Fusing this AHRS orientation with the SBA odometry estimate using the quaternion EKF correctly constrains the estimated path to the same flat plane as the GPS path, as shown in Figure 5.5.

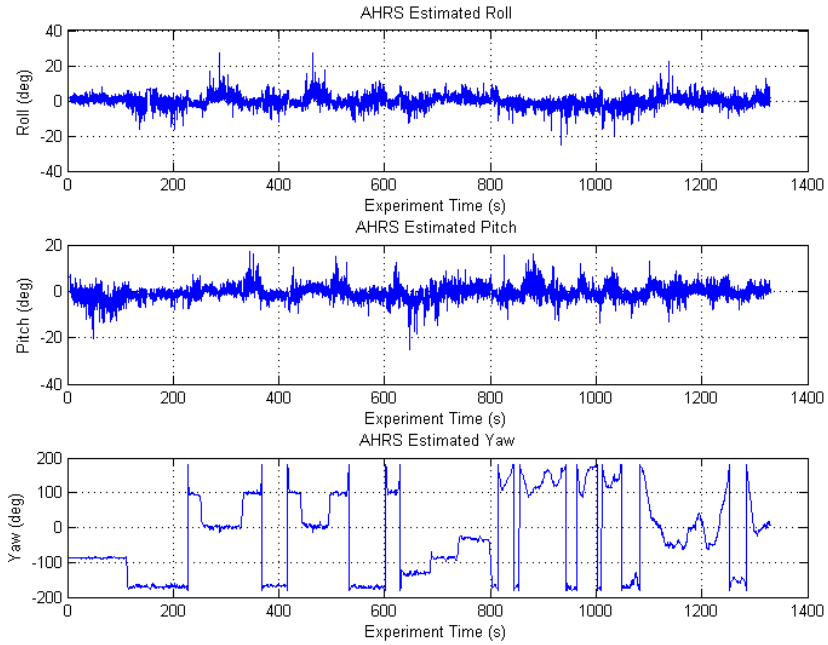


Figure 5.4: AHRS output for Tunks Park dataset

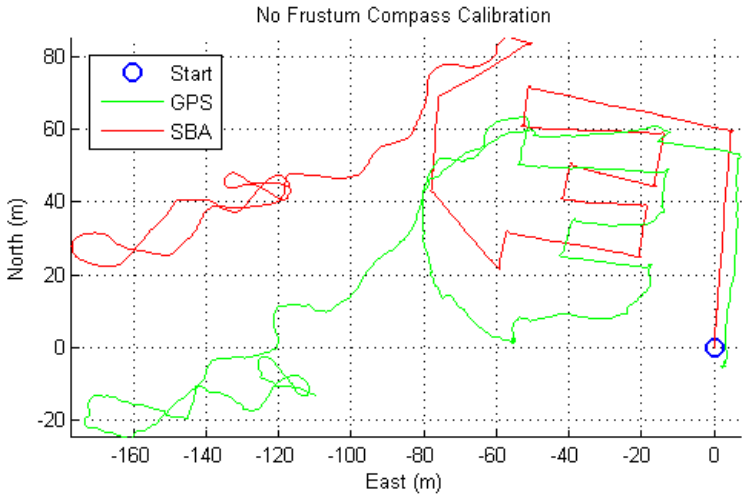


Figure 5.5: SBA odometry without frustum compass calibration

Although the path estimate has been significantly improved, without applying the frustum magnetic deviation there is still significant drift between the odometry and GPS trajectories. At the Western side of the path the position error is upwards of 50m. The effect of the nonlinear magnetic field distortions from the metal in the frustum frame on the AHRS yaw readings can be best observed by considering the approximately East to West straight line sections of the backwards E shape. These lines should be parallel but are clearly not so in the odometry estimate due to variation of the magnetic deviation with heading.

To correct this, the magnetic deviation at each timestep is found from the parabolic model developed in Section 4.3.2.4 and added to the original AHRS yaw readings. The

trajectory was then recomputed using the frustum-compensated yaw values converted back into quaternion rotations, and the result is shown in Figure 5.6. The odometry trajectory now tracks the GPS path with a small cumulative drift.

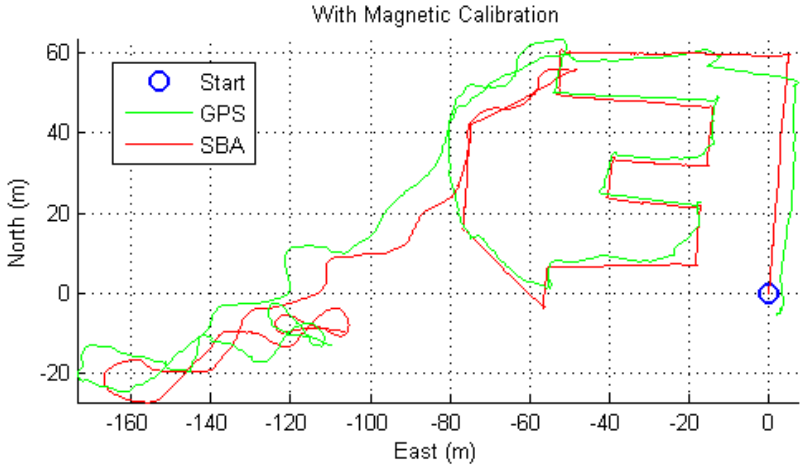


Figure 5.6: SBA odometry with frustum compass calibration

This drift can be evaluated as a linear error along the straight line segments by comparing the estimated path length with the physical path length between the marker cones measured using a surveyor's wheel. Table 5.1 gives the actual length of the first six legs, starting from the start point and traversing the backwards E shape counterclockwise, as well as the odometry estimate of the path length, the percentage error and the drift normalised by the time taken to complete each leg.

Table 5.1: Linear distance accuracy for ground facing path

Leg	Actual (m)	Est. (m)	Err. (%)	Drift (m/s)
1	60.2	59.3	1.49	0.0085
2	59.7	56.9	4.69	0.0250
3	10.8	10.6	1.95	0.0095
4	39.9	38.6	3.25	0.0173
4	15.0	14.5	3.33	0.0156
5	24.9	23.8	4.42	0.0229

Drifts of less than 25mm/sec were achieved on each leg, with linear distance errors between 1.49% and 4.69%. These low drifts indicate the success of the RGB-D and IMMU odometry system. The system accuracy for long distance positioning can be further demonstrated by considering that at the most Western point of the path, the furthest from the start, the error between the GPS position and the SBA odometry estimate is 6.5m, or 1.2% of the 550m travelled up to that point.

This error, however, is subject to noise in the GPS position signal, so a more reliable measure is the loop closure point at coordinates of around 60m North and -50m East in

Figure 5.6. The marker cone at this location was visited twice, with the camera performing a sharp turn at the cone each time, so the camera should return to the same location. The odometry estimate places the two visits to this cone, identified by the sharp corners in the path, about 6.3m apart, an error of 1.6% compared to the 400m travelled to this point.

Despite these good results, it is interesting to note that all odometry length estimates on each leg are underestimates. One possible reason for this is the Kinect camera was observed to infrequently stop supplying frames for up to 1 second. A reason for this could not be established and, with the device only on the market for two months when these experiments were conducted. there was minimal documentation to be found regarding a cause of or solution to this problem. The duration of this frame drop-out is large enough for keypoint tracking to fail between frames, so motion during this time could go unobserved by the odometry system. A possible solution to this would be to use a constant velocity model or short-term IMMU integration to estimate motion until tracking resumed.

This experiment also allows comparison of the benefit of bundle adjustment over direct frame to frame keypoint alignment. Figure 5.7 shows the comparison between using T_{RGBD} for the odometry estimate and using T_{SBA}. The RGB-D alignment path mostly mimics the SBA path, which is expected since the RGB-D alignments are used as initialisation for the bundle adjustment. However the frame to frame alignment has a greater tendency to underestimate the motion, with errors above 10% on the linear segments. The additional constraints of tracking features across multiple frames in the bundle adjustment process is responsible for the increased accuracy of SBA.

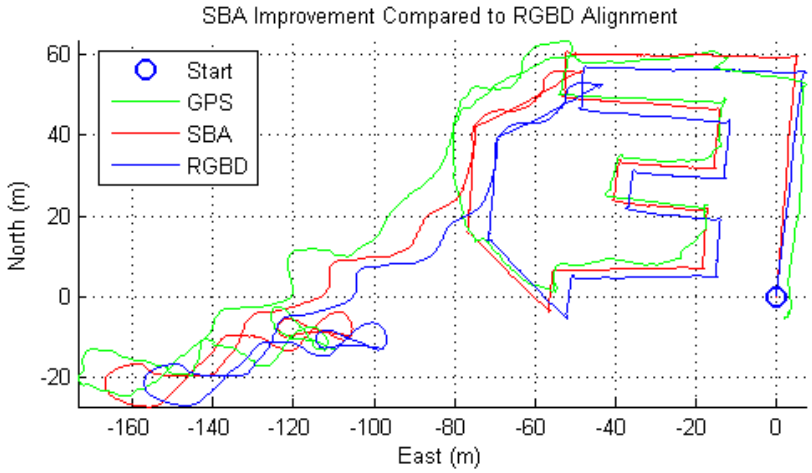


Figure 5.7: Improvement of SBA over frame to frame RGBD alignment

5.1.2 Harold Reid Road Test

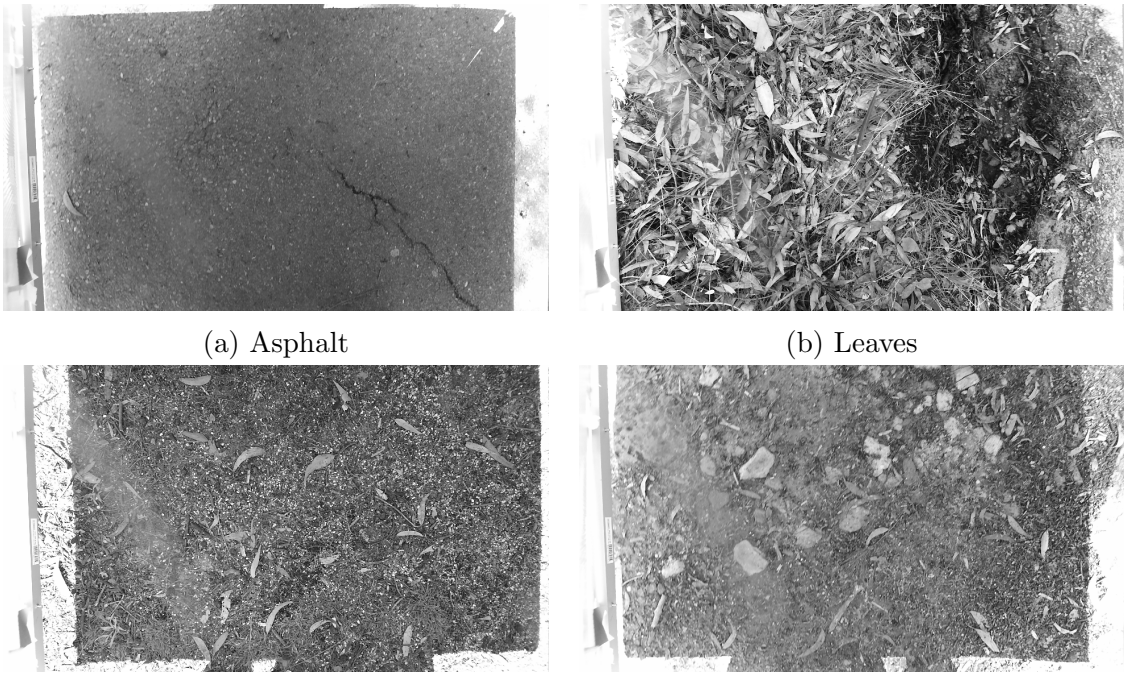
The second ground facing experiment was conducted at Harold Reid Reserve, following a narrow road. This test aimed to investigate the effect of terrain on the ground facing camera configuration. The path is shown in Figure 5.8 overlaid on Google Maps imagery.

Figure 5.8: GPS path of the ground-facing test at Harold Reid

The Northern side of the path taken followed the rough shoulder of the road and included gravel and leaf litter, with appearance as shown in Figure 5.9c and Figure 5.9b. The path then transitioned for a short period to the asphalt road surface, until a figure-eight loop was conducted on a surface of dirt and rocks appearing as shown in Figure 5.9d. The return path on the Southern side was entirely on asphalt and had appearance as shown in Figure 5.9a.

The result of the odometry estimate on this path is shown in Figure 5.10. Even though the leaf litter, which included sticks, small drainage gulleys and rocks, contains more depth features than the flat grass, it was not sufficiently varied in depth to trigger selection of the ICP odometry estimate. This is justified by comparing the SBA and ICP odometry estimates. Whereas SBA tracks the GPS path up until the transition to asphalt, ICP drifts as the lack of substantial depth features mean it is likely to iterate to an incorrect local minimum.

Considering just the SBA path now, it can be observed that the odometry trajectory estimate starts to drift significantly once on the asphalt. This can be attributed to how few visual features could be found on the surface with a mostly uniform dark appearance. In some frames, no keypoints at all were detected and tracking fails completely. The uniform appearance also means visual feature matches often fail Lowe's test in the match filtering stage, as the appearances of the few keypoints are not sufficiently distinct to confidently declare a match. Particularly, the return Southern stretch of the path suffers on the entirely asphalt surface, and the final position estimate is more than 100m from the starting point.



(a) Asphalt (b) Leaves

(c) Gravel (d) Dirt and rocks

Figure 5.9: Examples of terrain in the Harold Reid Road experiment

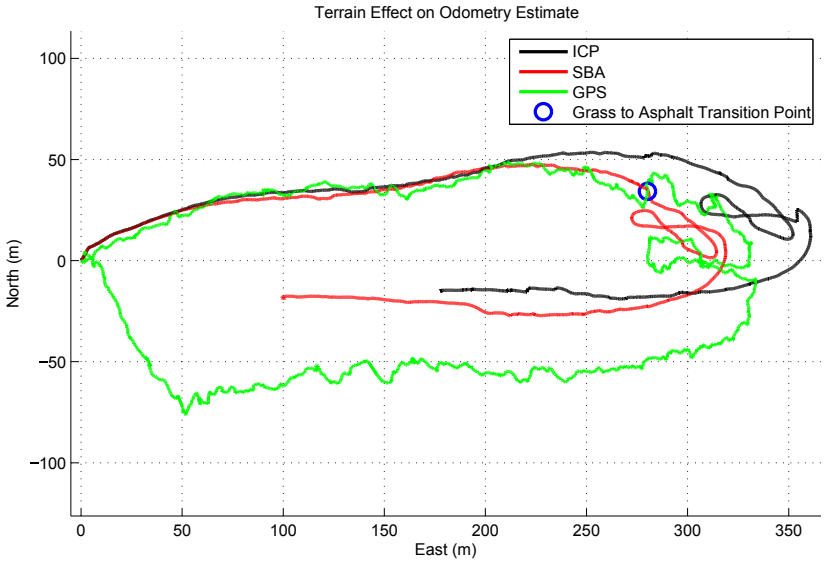


Figure 5.10: Terrain effect on the odometry types with ground facing camera at Harold Reid Reserve

This indicates that, unlike the grass of the Tunks Park test, or the leaf litter along the Northern section, asphalt is a terrain environment on which this odometry system fails. However this failure is not entirely unexpected for a feature based tracker on such a featureless surface, given that the feature detector thresholds are tuned for finding distinctive keypoints in a visually cluttered environment like Figure 5.9c and Figure 5.9b. However, spots of lighter colour are visible on the asphalt surface, so one solution might

be to automatically adjust thresholds until some baseline number of keypoints was found. Lowe's criteria would also need to be relaxed as the uniform appearance of the dark surrounding pixels would filter out many correct matches due to visual ambiguity, and RANSAC would have to be relied upon to remove outliers. Even with these changes the small size of the few keypoint features visible on asphalt would still make them unreliable to detect and match in the presence of motion blur as their appearance would experience significant distortion relative to their size.

The issue of motion blur is difficult to address with the Kinect camera as the exposure time of the colour camera cannot be controlled in software; it is automatically adjusted according to internal hardware settings. Instead, slower motion may be the only answer to reducing motion blur and achieving successful tracking on asphalt using this system, but considering testing speeds are already only around 50-60cm/sec this could be limiting for mobile robots.

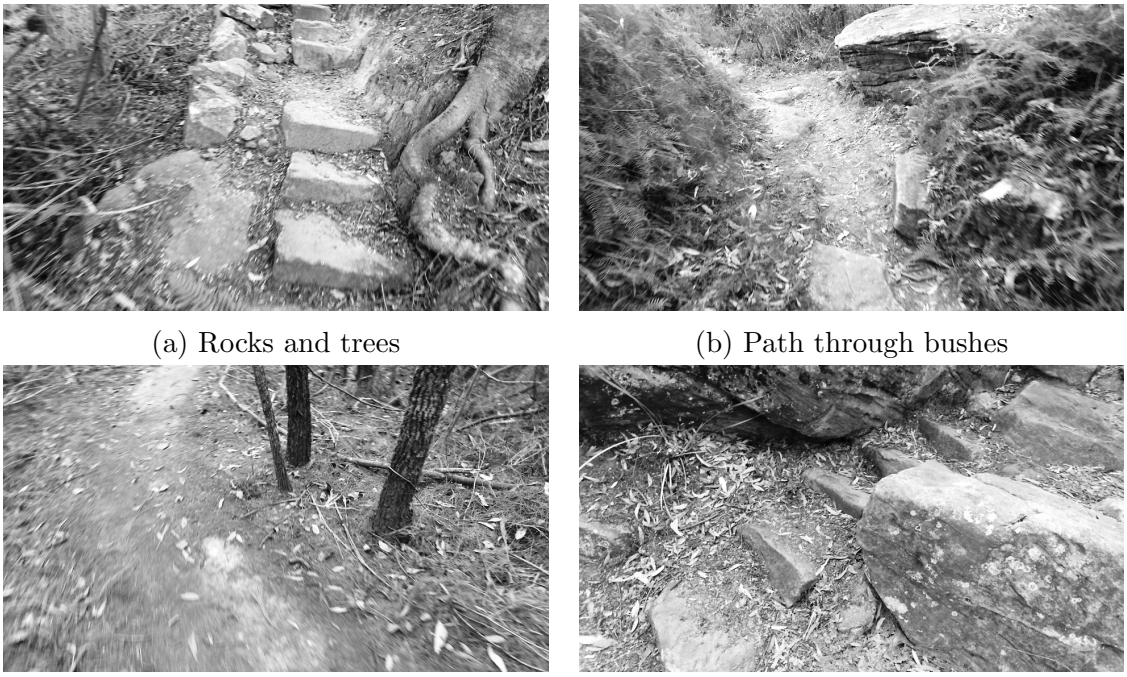
5.2 Hand Held Camera

The hand held camera test was conducted following a bush track at Harold Reid Reserve, shown with a GPS overlay on Google Maps imagery in Figure 5.11.

Figure 5.11: Handheld bushwalk GPS path

This experiment was intended to provide a more difficult motion pattern and more depth features for ICP localisation. In the ground facing frustum configuration, the Kinect was constrained to essentially operate in a 3 Degree of Freedom (DOF) problem of position and rotation in a flat plane, while the need to surmount obstacles with the frustum trolley limited the variation in depth features by the ground clearance of the frustum. Hand held camera motion allows a much more challenging 6 DOF motion pattern of rotation and translation in three axes, and allows terrain of much greater variation in depth to be observed.

The path chosen was intended to demonstrate this more complex motion pattern, and so followed a rough bush track down into a valley and returned, providing altitude changes and several gentle and steep corners along the way. The path also included varied terrain depth features much more suitable for localisation with ICP including trees, boulders, gulleys, leaf litter and stone steps. Example images from the colour camera are shown in Figure 5.12.



(a) Rocks and trees

(b) Path through bushes

(c) Flat path and trees

(d) Stone steps

Figure 5.12: Examples of terrain in hand held experiment

In order for the depth camera to remain unaffected outdoors without the frustum shield, it was necessary to conduct this test in late afternoon shade. While the depth camera supplies its own source of IR radiation and will operate normally in low or no external light, the colour camera suffers in such conditions due to the reduced visible light intensity. The vibrations of hand-held camera motion combined with the low light make motion blur a significant issue in this experiment for the RGB-D odometry system. In frames which suffer from motion blur, less keypoints are found as the distinctive light-surrounded-by-dark points or vice versa searched for by the Star keypoint detector are less apparent as the different coloured regions blur together. The amount of blur also changes the image appearance around a keypoint, reducing the chance of a correct match with BRIEF.

The effect of the motion blur in this experiment is shown in Figure 5.13, where the SBA alignment fails to track the GPS path, estimating less than half the actual path and not returning back to the starting point. This can be explained by considering that, with a small number of good keypoint matches to validate against, outliers may slip through

inlier filtering. In the FAQ of the authors on their SBA library [23], they specify that the process is not robust to the presence of outliers, and trying to perform bundle adjustment with even a few outliers can return incorrect results.

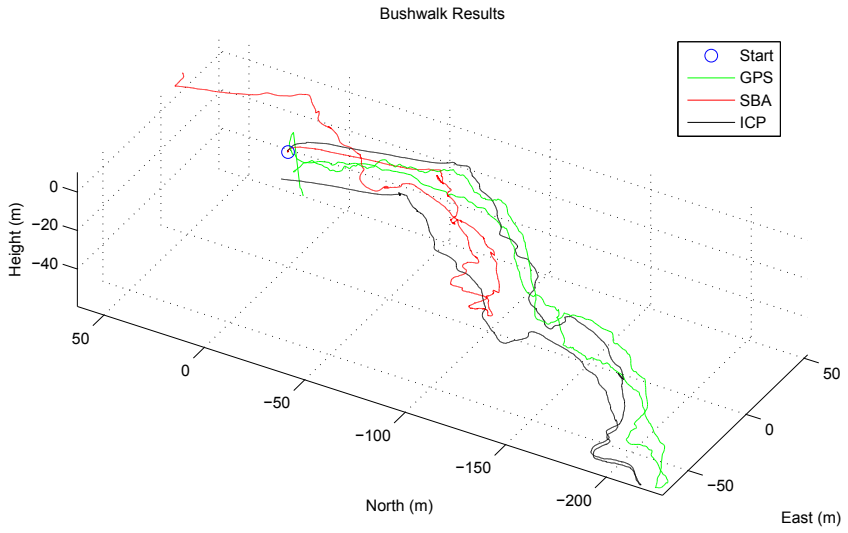


Figure 5.13: Hand held bushwalk results

The system presented in this thesis, however, was correctly selecting the ICP odometry due to the presence of sufficient depth features. The ICP path is significantly better than the SBA path, and follows the GPS path down into the valley and back up to the starting point. After completing the loop the odometry estimate is 20.3m from the starting location, which equates to an error of 3.4% of the total 600m travelled in the experiment.

5.3 Timing

In Chapter 3, odometry steps were presented as well as GPU based implementations of these steps. The timings for a serial and parallel implementation for relevant odometry processing steps, with the corresponding improvement factor in speed, are given in Table 5.2. The input data includes 1920×1080 color images and 512×424 dense depth images, with around 5000 keypoints found per image. These timings are presented using an Intel i5-3570 processor and an NVidia GTX580 graphics card running CUDA.

The reference benchmark used for the serial implementations come mostly from existing open-source libraries. Coordinate transform functions in PCL [42] were used as the reference for Kinect image mapping, while OpenCV [38] provided the reference for the serial implementations of the Star detector, BRIEF descriptor and brute-force matching. The serial RANSAC procedure was written purposely for this thesis to compare against the GPU equivalent version.

Table 5.2: Parallelisation timing

Step	Serial (s)	Parallel (s)	Improvement Factor
Mapping	0.3455	0.0054	63.68
Star	0.2275	0.0904	2.52
BRIEF	0.4464	0.0197	22.70
Matching	0.2409	0.0150	16.07
RANSAC	1.0189	0.1770	5.77

A significant increase in speed is attained for mapping the depth image by parallel processing. However computation of the Star detector response suffers from a bottleneck as the integral image computation and non-maxima suppression are performed by the CPU. The BRIEF descriptor and brute-force matching steps both require a large number of accesses to the slower global CUDA memory, which limit the speed increase. RANSAC also experiences only modest gains, as the CPU is doing the work of selecting the sample and finding the minimising transform for each iteration, providing another bottleneck.

While overall these parallelisations greatly reduce the frame to frame processing time, at this stage the odometry processing is not sufficiently fast to call it a real-time system. There are a number of possible avenues which could be explored to reduce computation time, however they involve reducing the amount of data being operated upon, which could have an associated impact on accuracy. The trade-off between accuracy and speed is left for future study.

Possible ways to reduce processing time include subsampling the colour image to a smaller one. This would mean faster computation of the integral images for Star keypoints. Different thresholds for Star and a smaller image would also reduce the number of keypoints found. This would in turn linearly reduce the time taken to perform the BRIEF descriptor computation and RANSAC, while quadratically reducing the number of operations required in brute-force matching and SBA. ICP could also be performed on a subsampled depth image.

Chapter 6

Conclusion

In this thesis the problem of long-range outdoor odometry estimation using RGB-D cameras was approached. The research question has been successfully addressed through development of an odometry system which combined 3D keypoint matching and ICP localisation and a switching strategy to select between them depending on terrain. An EKF was designed to fuse the odometry estimate with an AHRS orientation computed using a low-cost IMMU to constrain orientation errors. A novel system configuration was presented which shielded the Kinect for Windows version 2 RGB-D camera from IR interference from sunlight so that it could be used outdoors. A compass calibration routine was undertaken to correct for magnetic field distortions caused by the metal frame of the shield.

Odometry accuracy testing was performed in a variety of motion conditions, including using a ground facing camera on flat terrain and hand-held camera motion along a bush track. Different terrain was also investigated including flat grass, asphalt, and trees and boulders along the bush track. The system was found to fail on nearly featureless terrain such as an asphalt surface. However using bundle adjusted keypoint odometry on flat terrain and ICP alignment on depth-varying terrain the system otherwise achieved final position errors of less than 4% and drifts of less than 25mm/sec over distances greater than 500m. GPU implementations of processing steps were used to achieve processing speed increases by a factor of between 2-60.

A conference paper detailing the methods and results from this thesis has also been submitted to the International Conference on Robotics and Automation.

6.1 Future Work

Some exciting possibilities exist to extend the work presented in this thesis, including:

- Experiment with tradeoffs between image size, Star keypoint thresholds, positioning accuracy and execution time, as described in Section 5.3, in order to move towards

real-time processing rates. This is important for the system to be of practical use for autonomous mobile robots.

- Implement on new mobile GPUs. While mobile robot platforms have previously been limited by their low-power onboard CPUs, new development boards like the NVidia Jetson TK-1 include several hundred parallel GPU cores with a total power draw of only a few watts. This makes the possibility of real-time implementation on mobile platforms more feasible.
- Integrate RGB-D images to produce 3D colour maps of the paths travelled. Apart from the usefulness of 3D maps for many applications, it is also an important stage of SLAM style work which can use the map to further refine the pose estimation.
- Implement a motion model for predicting motion when the camera drops frames, such as constant-velocity or IMMU acceleration integration.

Bibliography

[1] J. Leonard and H. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 376–382, June 1991.

[2] B. Barshan and H. F. Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 328–342, 1995.

[3] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. 2004 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2004, pp. 652–659.

[4] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Int. Symp. on Robotics Research*, 2011, pp. 1–16.

[5] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera," in *Proc. 24th Annu. ACM Symp. on User Interface Software and Technology*. ACM, 2011, pp. 559–568.

[6] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," vol. 14, no. 2, pp. 239–256, February 1992.

[7] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proc. 2013 IEEE Int. Conf. on Robotics and Automation*. IEEE, 2013, pp. 3748–3754.

[8] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: using depth cameras for dense 3D modeling of indoor environments," in *Experimental Robotics*. Springer, 2014, pp. 477–491.

[9] K. Konolige, M. Agrawal, and J. Sola, "Large-scale visual odometry for rough terrain," in *Robotics Research*. Springer, 2011, pp. 201–212.

[10] B. Langmann, K. Hartmann, and O. Loffeld, "Depth camera technology comparison and performance evaluation," in *Proc. 1st Int. Conf. on Pattern Recognition, Applications and Methods*. SciTePress, 2012, pp. 438–444.

[11] D. Scaramuzza and F. Fraundorfer, "Visual odometry part I: the first 30 years and fundamentals," *Robotics & Automation Magazine, IEEE*, vol. 18, no. 4, pp. 80–92, 2011.

[12] L. Matthies, M. Maimone, A. Johnson, Y. Cheng, R. Willson, C. Villalpando, S. Goldberg, A. Huertas, A. Stein, and A. Angelova, "Computer vision on Mars," *Int. Journal of Computer Vision*, vol. 75, no. 1, pp. 67–92, 2007.

[13] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[14] D. Nister, "Preemptive RANSAC for live structure and motion estimation," *Machine Vision and Applications*, vol. 16, no. 5, pp. 321–329, 2003.

[15] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

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

[17] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and generic corner detection based on the accelerated segment test," in *Proc. European Conf. on Computer Vision, 2010*. Springer, September 2010, pp. 183–196.

[18] M. Agrawal, K. Konolige, and M. R. Blas, "CenSurE: Center surround extremas for realtime feature detection and matching," in *Proc. 2008 European Conf. on Computer Vision*. Springer, 2008, pp. 102–115.

[19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[20] F. Fraundorfer and D. Scaramuzza, "Visual odometry part II: matching, robustness, optimization, and applications," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 2, pp. 78–90, 2012.

[21] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proc. European Conf. on Computer Vision, 2006*. Springer, 2006, pp. 404–417.

[22] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proc. European Conf. on Computer Vision, 2010.* Springer, 2010, pp. 778–792.

[23] M. I. Lourakis and A. A. Argyros, "SBA: A software package for generic sparse bundle adjustment," *ACM Transactions on Mathematical Software (TOMS)*, vol. 36, no. 1, p. 2, 2009.

[24] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment, a modern synthesis," in *Vision Algorithms: Theory and Practice.* Springer, 2000, pp. 298–372.

[25] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proc. 3rd Int. Conf. on 3-D Digital Imaging and Modeling, 2001.* IEEE, 2001, pp. 145–152.

[26] H. Du, P. Henry, X. Ren, M. Cheng, D. B. Goldman, S. M. Seitz, and D. Fox, "Interactive 3D modeling of indoor environments with a consumer depth camera," in *Proc. 13th Int. Conf. on Ubiquitous Computing.* ACM, 2011, pp. 75–84.

[27] F. Steinbrucker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Proc. 2011 IEEE Int. Conf. on Computer Vision Workshops.* IEEE, 2011, pp. 719–722.

[28] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *Proc. 2013 IEEE Int. Conf. on Robotics and Automation.* IEEE, 2013, pp. 5724–5731.

[29] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent," in *Robotics: Science and Systems*, 2007.

[30] A. M. Sabatini, "Estimating three-dimensional orientation of human body parts by inertial/magnetic sensing," *Sensors*, vol. 11, no. 2, pp. 1489–1525, 2011.

[31] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," in *Proc. 2011 IEEE Int. Conf. on Rehabilitation Robotics.* IEEE, 2011, pp. 1–7.

[32] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[33] L. Kneip, M. Chli, R. Siegwart, R. Y. Siegwart, and R. Y. Siegwart, "Robust real-time visual odometry with a single camera and an IMU," in *BMVC*, 2011, pp. 1–11.

[34] S. B. Gokturk, H. Yalcin, and C. Bamji, "A time-of-flight depth sensor - system description, issues and solutions," in *Proc. 2004 Conf. on Computer Vision and Pattern Recognition Workshop*, 2004, pp. 35–35.

[35] M. Andersen, T. Jensen, P. Lisouski, A. Mortensen, M. Hansen, and T. G. Ahrendt, "Kinect depth sensor evaluation for computer vision applications," Aarhus University, Report Tech. Rep. ECE-TR-6, 2012.

[36] J.-Y. Bouguet. (2004) Camera calibration toolbox for Matlab. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/

[37] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *Proc. 2011 IEEE Int. Conf. on Computer Vision.* IEEE, 2011, pp. 2564–2571.

[38] Willow-Garage and Itseez. (2014) OpenCV. [Online]. Available: http://opencv.org

[39] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Jornal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[40] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 5, pp. 698–700, 1987.

[41] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[42] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[43] K.-L. Low, "Linear least-squares optimization for point-to-plane ICP surface registration," University of North Carolina, Report Tech. Rep. TR04-004, 2004.

[44] Eberly, D., "Least squares fitting of data," Magic Software, Inc., Chapel Hill, NC, 2001.

[45] E. B. Dam, M. Kock, and M. Lillholm, "Quaternions, interpolation and animation," University of Copenhagen, Tech. Rep. Tech. Rep. DIKU-TR98/5, 1998.

[46] J. Lobo and J. Dias, "Relative pose calibration between visual and inertial sensors," *Int. Journal of Robotics Research*, vol. 26, no. 6, pp. 561–575, 2007.

[47] G. Welch and G. Bishop, "An introduction to the Kalman Filter," University of North Carolina - Department of Computer Science, Tech. Rep. TR 95-041, July 2006.

[48] C. Kerl, "Odometry from RGB-D cameras for autonomous quadrocopters," Master's thesis, Technical University Munich, Germany, 2012.

[49] S. Bargoti, "Local navigation of a mars rover using a depth sensor," Honours Thesis, The University of Sydney, 2011.

[50] 123 Kinect. (2014) Everything Kinect 2. [Online]. Available: http://123kinect.com/everything-kinect-2-one-place/43136/

[51] Microsoft. (2014) Kinect for Windows features. [Online]. Available: http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx

[52] "ADXL345 Digital Accelerometer," Analog Devices, Norwood, USA, 2009.

[53] "ITG-3200 3-Axis Gyro," InvenSense, SunnyVale, USA, 2010.

[54] "HMC5883L 3-Axis Digital Compass IC," Honeywell, Plymouth, USA, 2011.

[55] Peter Bartz. (2013) Razor AHRS. [Online]. Available: https://github.com/ptrbrtz/razor-9dof-ahrs

[56] "Handbook of Magnetic Compass Adjustment," National Geospatial-Intelligence Agency, Bethesda, USA, 2004.