Robust 3D Registration and Tracking with RGBD Sensors

Abdenour Amamra
PhD

Centre for Electronic Warfare

Robust 3D Registration and Tracking with RGBD Sensors

By Abdenour Amamra

This thesis is submitted in partial fulfilment of the degree of

Doctor of Philosophy

In the

Centre for Electronic Warfare

Cranfield University

Date of submission: May 2015
Supervisor: Dr. Nabil Aouf

Abstract

This thesis investigates the utilisation of cheap RGBD sensors in rigid body tracking and 3D multiview registration for Augmented and Virtual reality applications. RGBD sensors can be used as an affordable substitute for the more sophisticated, but expensive, conventional laser-based scanning and tracking solutions. Nevertheless, the low-cost sensing technology behind them has several drawbacks such as the limited range, significant noisiness and instability.

To deal with these issues, an innovative adaptation of Kalman filtering scheme is first proposed to improve the precision, smoothness and robustness of raw RGBD outputs. It also extends the native capabilities of the sensor to capture further targets. The mathematical foundations of such an adaptation are explained in detail, and its corrective effect is validated with real tracking as well as 3D reconstruction experiments. A Graphics Processing Unit (GPU) implementation is also proposed with the different optimisation levels in order to ensure real-time responsiveness.

After extensive experimentation with RGBD cameras, a significant difference in accuracy was noticed between the newer and ageing sensors. This decay could not be restored with conventional calibration. Thus, a novel method for worn RGBD sensors correction is also proposed.

Another algorithm for background/foreground segmentation of RGBD images is contributed. The latter proceeds through background subtraction from colour and depth images separately, the resulting foreground regions are then fused for a more robust detection.

The three previous contributions are used in a novel approach for multiview vehicle tracking for mixed reality needs. The determination of the position regarding the vehicle is achieved in two stages: the former is a sensor-wise robust filtering algorithm that is able to handle the uncertainties in the system and measurement models resulting in multiple position estimates; the latter algorithm aims at merging the independent estimates by using a set of optimal weighting coefficients. The outcome of fusion is used to determine vehicle's orientation in the scene.

Finally, a novel recursive filtering approach for sparse registration is proposed. Unlike ordinary state of the art alignment algorithms, the proposed method has four advantages that are not available altogether in any previous solution. It is able to deal with inherent noise contaminating sensory data; it is robust to uncertainties related to feature localisation; it combines the advantages of both $\boldsymbol{L_2}, \boldsymbol{L_\infty}$ norms for a higher performance and prevention of local minima; it also provides an estimated rigid body transformation along with its error covariance. This 3D registration scheme is validated in various challenging scenarios with both synthetic and real RGBD data.

In memory of aunty ***Aicha Amamra***

Acknowledgment

First and foremost I offer my sincerest gratitude to my supervisor, Dr Nabil Aouf, who has supported me throughout my thesis with his patience and knowledge whilst allowing me to work in my own way. I am also grateful to my committee members, particularly Prof Mark Richardson, for his supportive approach in discussing and reviewing my progress.

To my parents who, although being so far away, have been with me in every step. To my dearest wife, thank you for your encouraging words and patience for all the time I spent away from you. My brothers, sisters and my uncle Abdelkader Attalah; thank you all for your love and encouraging support.

To Luke Feetham for his continual help in thoroughly proofreading most of my research papers. To Dr Gray Greer and Dr Dowling Stuart for their assistance with English. Many thanks to Dr Lounis Chermak for his fruitful technical and moral advice throughout my research. To Oualid and Saif my office mates. To Riad, Tarek, Mohammed, Ivan, Abdennour and Ozgun my colleagues in the lab. To my home mates Badis and Rahim.

I would like to express my gratitude to the staff of the Military Polytechnic School of Algeria. I would also thank Cranfield University staff, in particular, those of Defence Academy campus, Shrivenham.

Finally, I'm grateful to all those who helped in a way or in another in the achievement of this research work.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

2D	Two Dimensions
3D	Three Dimensions
AR	Augmented Reality
CAD	Computer Aided Design
CPU	Central Processing Unit
CSHOT	Colour Signature of Histograms of Orientations
DOF	Degrees of Freedom
DOG	Difference of Gaussians
EKF	Extended Kalman Filter
EM	Expectation Maximisation
EMICP	Expectation Maximisation Iterative Closest Point
FOV	Field of View
FPS	Frames Per Second
GMM	Gaussian Mixture Model
GPGPU	General Purpose GPU
GPR	Gaussian Process Regression
GPU	Graphics Processing Units
GTLS	Generalised Total Least Squares
HD	High Definition
HMD	Head Mounted Device
HSV	Hue Saturation Value
Hz	Hertz
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
IR	Infra-Red
KF	Kalman Filter
LED	Light Emitting Diode
LS	Least Squares
ML	Maximum Likelihood
MP	Mega Pixels
MR	Mixed Reality
PCA	Principal Components Analysis
RAM	Random Access Memory
RANSAC	RANdom SAmple Consensus
RF	Reference Frame
RGB	Red Green Blue

RGBD	Red Green Blue Depth
RLS	Recursive Least Squares
RMSE	Root Mean Squares Error
SHOT	Signature of Histograms of OrienTations
SIFT	Scale Invariant Feature Transform
SOCP	Second Order Cone Programming
SURF	Speed Up Robust Features
SVD	Singular Value Decomposition
TOF	Time of Flight
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
UKF	Unscented Kalman Filter
UPF	Unscented Particle Filter
VGA	Video Graphics Array
VR	Virtual Reality
WICP	Weighted Iterative Closest Point
WLS	Weighted Least Squares

1 Introduction

1.1 Background & Objectives

Sophisticated mixed (i.e. Virtual and Augmented) reality platforms require a complete and accurate real-time information about the geometry and the photometry of the physical world [1]. Advanced laser scanners and tracking systems can be a straightforward solution to acquire these cues [2]. Yet this is not an optimal one, due to its cost, size and complicated usage. Such properties remain a prohibitive challenge for many budget applications [3].

On the other hand, 2D imagery can be an affordable substitute for the previous expensive advanced tools [4]. The latter has achieved a significant success among industrial and research actors [5]. This success is underlined by the abundance of cheap cameras, continually developing image processing and computer vision algorithms as well as the simplicity of acquisition and exchange between different devices.

Despite their attractive cost and ease of access, 2D images still suffer from an inherent drawback in their informativeness due to the loss of depth information, i.e. the geometry. This deficiency is undergone during the projection from the

3D world onto the 2D camera plane. Many attempts have already been carried out in order to recover the missing geometry, but only a few impractical solutions have reached the required accuracy at a near real-time performance, i.e. greater than 10 frames per second (FPS) [6].

Two other types of depth sensors, namely Time of Flight (TOF) and structured light cameras, have recently become available to the public at an affordable cost [8]. Both sensing technologies are capable of delivering the colour image (RGB) and the depth map (D) at a satisfactory frame rate (30 FPS). In addition, the release of the two versions of Microsoft Kinect has featured various successful solutions for tackling the traditional problems of Mixed Reality. Several applications have consequently benefited from such an optimal trade-off between performance and cost to realise an interesting product [7].

Regardless of their economical capabilities of simultaneous visual and structural features acquisition, RGBD sensors have several limitations owing to their low-cost [8]. They are able to deliver fair quality depth readings for a relatively narrow field of view. For instance, the view can be extended by involving multiple sensors in order to cover the entire scene. Nevertheless, because of their active nature, they emit and scan infrared (IR) light to infer the range of objects, concurrent utilisation of multiple RGBD cameras results in interference between the respective IR beams. The depth image is accordingly corrupted with void pixels due to confusion during the estimation of their depth [9].

Furthermore, the effective range of these cameras stretches from 0.8m to 4.0m, which is insufficient in many practical scenarios. More importantly, the depth measurements can be altered by two types of noise [10]: The former is due to the quantification process, i.e. the action of discretising the continuous real world in order to accommodate its depth image in the available storage capacity; The latter, on the other hand, is the fluctuations in measurements due to perturbing lighting conditions, materials of imaged objects as well as the wear incurred by the sensor over time.

Several challenges should, therefore, be overcome in order to improve the accuracy and stability of RGBD data with adequate rapid filtering strategies. The purpose is the delivery of a clean (eliminate outliers), accurate (reduce inherent measurement noise effect) and smooth (eliminate discreteness) outputs. After such a pre-processing procedure, the outputs should bear maximally useful information.

The result of the previous procedure can be used in tracking and registration algorithms. The objective of tracking is the delivery of accurate and robust (not affected by fluctuations in measurements) position and orientation (6 Degrees of Freedom) of objects at a high frame rate ($\geq$ 20 FPS). On the other hand, the registration must be able to support MR applications with accurate and robust alignments between stored templates and acquired models in real-time.

1.2 Research Statement

In light of this background and challenges, the objective of this thesis is set to:

The enhancement of the capabilities of cheap depth sensors to develop robust and accurate real-time tracking and registration solutions for Mixed Reality applications.

1.3 Thesis Organisation & Contributions

To fulfil this research statement, the thesis was organised in seven chapters. The first is the Introduction. It should be noted that this section does not include any detailed discussion of the state of the art. A detailed analysis of the related works underlying each contribution is included at the beginning of each chapter.

The remaining elements of this thesis consist of a theoretical primer followed by the four seminal chapters, where novel contributions are treated, and a conclusion.

Chapter 2: Background (contribution *5*)

In this chapter, an overview of the different concepts and tools regarding image acquisition and object tracking are addressed:

- Virtual and Augmented reality are presented and illustrated with several established applications. In addition, their respective specifications and parameters are exposed in order to assess the quality of emersion in virtual space or augmentation of real data.
- The main strategies to image registration (sparse and dense) as well as rigid-body tracking (marker or markerless) are debated.
- Mono, stereo and multiple camera models are investigated.
- A thorough analysis of the available 3D keypoint extractors and descriptors is also presented.
- Finally, Graphics Processing Units (GPU) are investigated for the purpose of leveraging their processing capabilities for sizeable 3D data processing.

Chapter 3: GPU-based real-time RGBD data filtering (papers *1, 4, 6*)

In this chapter, a novel filtering scheme is contributed to improve the precision of Kinect as a real-time RGBD capture device.

- The complete architecture of the filtering system is presented along with its principal components.
- The working principle of RGBD sensors is thoroughly analysed in order to identify the source of inaccuracy regarding measurements. In addition, the technical specifications characterising the sensors are highlighted.

- The Mathematical formulation of depth camera properties that have been discovered by the author are then detailed. The adaptation of a Kalman filter to the model of measurements discretisation is then theoretically established. The Kalman filter will, therefore, serve as a real-time de-noising/enhancement solution for the raw outputs of depth cameras.

- The efficiency of the proposed scheme is tested in both tracking and registration scenarios.

- The former application (tracking) consists in the localisation of a moving vehicle where the author's findings are verified to be beneficial to recognise these sensors as an accurate real-time tracking device. The results of this scenario are tested on moving ground robot localisation with a single camera. The accuracy of the proposed filter has demonstrated its superiority over equivalent algorithms. The purpose of this type of application is the evaluation of precision regarding the contributed filter.

- The latter scenario consists of a demonstration of the performance of the proposed scheme on a 3D registration pipeline. The potential of such a strategy was noticeable after testing it against equivalent methods. Here, the purpose of tests is the assessment of the visual quality of the geometry built upon filtered 3D data.

- A GPU implementation of the proposed approach is also provided with different levels of optimisation.

Chapter 4: RGBD Data Correction and Background Removal (papers *7, 8, 9*)

In this chapter, two innovative pre-processing algorithms for the correction of depth measurements and the segmentation of foreground regions are presented.

- A novel method to calibrate accurately the ageing depth cameras is proposed. This approach can be based on either simple interpolation or on more sophisticated Gaussian Process Regression (GPR).

- Another innovative background/foreground segmentation algorithm is proposed. The latter is based on Gaussian Mixture Models (GMM) and colour/depth image-fusion strategy for a more robust segmentation.

- The experiments carried out on real data prove the weaknesses of the standard calibration in correcting age-related decay of accuracy and the corrective asset of the proposed solution. They also show the robustness of the suggested segmentation algorithm in coping with illumination changes, shadows and reflections.

- The two algorithms were implemented in the GPU. In addition, several additional optimisation policies were taken into account to ensure an entire profit from the frame rate delivered by the sensor (30 FPS).

Chapter 5: Real-Time Multiview Data Fusion for Object Tracking with RGBD Sensors (paper *2*)

In this chapter, a novel approach for accurate tracking of moving vehicles with multiple RGBD cameras is presented. This chapter exploits the smoothed results of a Kalman filtering scheme and the pre-processing achieved by both previous algorithms. It is designed in the form of a pipeline of processing where the input is raw RGBD data, and the output is accurate localisation of the objects.

- The sensors are initially corrected for a possible deficiency in the accuracy of depth measurements.

- A second depth data correction/enhancement algorithm is run to cope with the inherent noise contaminating the sensing. This phase is based on the filter of Chapter 3.

- The markers employed to localise the object are extracted from the acquired images by means of the background removal solution that has been developed in the preceding chapter.

- A novel tracker based on the Robust H_∞ filter is developed in order to improve each sensor's tracking ability. The motion model of the moving vehicle used in Chapter 5 (see Figure 5.1) is assumed to be unknown. Hence, the algorithm should be endowed to cope with modelling and measurement uncertainties. In other words, it must be able to ensure a high level of robustness in the output whatever perturbed the input. The result of this procedure is the sensor-wise position estimates.

- Another innovative weighting strategy is used in a Covariance Intersection Fusion scheme in order to combine optimally sensor-wise results into a single estimate.

- The computation-costly fragments of the solution are implemented in the GPU. As a consequence, the whole tracking system, from the acquisition of raw data to the delivery of the result, is capable of operating at up to 25 FPS.

Chapter 6: A Recursive Robust Filtering Approach for 3D Registration (paper *3*)

In this chapter, a novel recursive robust filtering approach for sparse 3D registration is proposed. Unlike the ordinary state of the art alignment algorithms, the proposed algorithm has four advantages that, in the author's knowledge, have not yet cohabited in any previous solution in the literature. This iterative registration approach has the ability to align large datasets

iteratively using just a few key points contaminated with measurement noise. It is also robust to the perturbations caused by the uncertain key point localisation. In addition, it combines the advantages of both L_∞ and L_2 norms for a higher performance and a more likely prevention of local minima. The result is an estimated rigid body transformation along with its error covariance.

- The detailed modelling behind Weighted Least Squares (WLS) is initially discussed along with its recursive counterpart (RLS). The link between the Kalman filter model and the RLS is, therefore, settled.
- The 3D registration problem is then fitted into a Kalman filter framework in order to express it in an iterative manner.
- The parametric uncertainty of the 3D points is quantified then used in the formulation of a 3D registration problem with a Robust H_∞ filtering model.
- The mathematical rationale of the proposed approach is explained in detail. Moreover, the results are validated on various challenging scenarios with both synthetic and real data.

Chapter 7: Conclusion & Future Works

In this section, the thesis is concluded. The degree of achievement regarding the objectives raised in the Introduction is assessed, and potential future works are covered.

1.4 Contributed Papers

Journals

1. A. Amamra, N. Aouf "GPU-based real-time RGBD data filtering," *Journal of Real-Time Image Processing*, Sep. 2014. **(Published)**

2. A. Amamra and N. Aouf, "Real-time multiview data fusion for object tracking with RGBD sensors," *Robotica*, pp. 1–25, Dec. 2014. **(Published)**

3. A. Amamra, N. Aouf, D. Stuart and M. Richardson "A Recursive Robust Filtering Approach for 3D Registration," *Signal, Image and Video Processing (SIVP).* **(Accepted)**

Conferences

4. A. Amamra and N. Aouf, "Robust and Sparse RGBD Data Registration of Scene Views," in 2013 17th International Conference on Information Visualisation, 2013, pp. 488–493. **(Published)**

5. A. Amamra and N. Aouf, "Indoor 3D Augmented Reality," in 11th Electro Optics and Infrared Conference, Shrivenham, Defence Academy of the UK, Jun. 2013. **(Presented)**

6. A. Amamra and N. Aouf, "Real-Time Robust Tracking with Commodity RGBD Camera," in 2013 IEEE International Conference on Systems, Man, and Cybernetics, 2013, pp. 2408–2413. **(Published)**

7. A. Amamra and N. Aouf, "RGBD sensors correction with Gaussian process regression," in Proceedings ELMAR-2014, 2014, pp. 1–4. **(Published)**

8. A. Amamra, T. Mouats, and N. Aouf, "GPU based GMM segmentation of kinect data," in Proceedings ELMAR-2014, 2014, pp. 1–4. **(Published)**

9. A. Amamra, T. Mouats, and N. Aouf, "Real-Time Background/Foreground Segmentation of RGBD Data", in Proceedings IC-STCC, 2014. (**Accepted**)

1.5 Software Tools

The most notably used software tools to achieve the objectives of this research work are:

- **C/C++** it is an object oriented programming language characterised by a high performance and a widespread usage in most computer vision and graphics applications. In this research, it has been used to develop real-time algorithms destined to run on the CPU.

- **CUDA** stands for Compute Unified Device Architecture, is a heterogeneous programming language that executes in both CPU and GPU. It was created by NVIDIA. In this thesis, CUDA is used to write the online parallel algorithms that run in the GPU.

- **Matlab** contraction of Matrix Laboratory, is a multi-paradigm numerical programming framework that facilitates mathematical and particularly linear algebra operations. In this thesis, Matlab was used as a quick test tool for different algorithms as well as to plot the results.

- **PCL** (Point Cloud Library) is a 2D/3D image and point cloud processing library. It encloses a broad range of functions for 3D point cloud processing such as the computation of normals and curvature, downsampling, outlier removal, filtering, feature extraction as well as alignment to name a few. In this research, PCL is used for standard 3D point clouds operations.

- **OpenCV** is a real-time image processing and computer vision library. It includes most of the standard algorithms such as image de-noising, binarisation, feature extraction, description and matching, pose estimation and calibration to name a few. It is used to capture and treat 2D images in real time.

- **Google SketchUp** is a 3D model design tool. It can be utilised to build architectural, mechanical and video games' 3D samples. In the thesis, SketchUp is employed for the design of the 3D model regarding the lab where the experiments have been conducted.

- **OpenGL** (Open Graphics Library) is a multi-platform 2D/3D graphics rendering library. In this research, it is used to ensure hardware-accelerated rendering and interaction with 3D models.

- **Boost** it is a collection of **C++** libraries that supplies this research with functions for thread and process creation, management as well as synchronisation.

2 Background

One of the most challenging aspects of human-machine interaction systems design is the integration of physical and digital worlds in the same environment. This fusion involves the development of *Mixed Reality Systems* [11] (see Figure 2.1). In other words, the technology behind includes both the *Augmented Reality* and the *Virtual Reality* concepts. The mixed reality paradigm allows the digital world to be extended into the user's physical world. It studies how the human could interact with machines, and to what extent this interaction could be successful [12]. In such a system, the user perceives both the physical and the digital worlds around them. They can even be immersed in a digital scene using the 3D immersion hardware (head mounted devices, tracking systems, haptic devices, rear-projection screens, etc.).

This field of research is highly interdisciplinary as it engages many domains such as computer vision, signal processing, computer graphics, user interfaces, information visualisation and the design of displays and capture sensors.

In this chapter, an overview of the different tools required for the acquisition of the 3D structure of the physical scene, and the motion (pose and position) of the mobile entities are discussed.

Figure 2.1 Virtuality Continuum; Courtesy of Milgram et al. [12]

2.1 Augmented Reality (AR)

The concept of Augmented Reality was invented for the purpose of combining computer-generated models (synthetic data) with image data captured from a real scene. The origin of the word *Augmented* does not make sense until one focuses on the perceptual sensing observed by human beings towards their surroundings [13]. The Reality itself cannot be improved, but the level of perception can be. As a result, the term AR in this manuscript means *the enhanced perception of the real world.*

Azuma et al. [14] contributed an interesting survey paper on the trend of the state of the art of AR research. The recent widespread abundance of smartphones and gaming sensors at an affordable price has opened a new perspective for more sophisticated applications. The last mentioned of these emerged in the computer vision research community and afterwards was extended to the industrial level. As a result, many commercial applications are now available in the market.

Figure 2.2 (a) is taken from a commercial technology AREngine™ [1] that augments the real world with 3D virtual content after the recognition and pose estimation of the target image. Figure 2.2 (b) depicts an augmented reality

[1] http://www.metaio.com/solutions/vision-solutions-new/arengine/. 2015

exhibit that enhances the images of visitors in an Arctic tundra during the Wildlife Film Festival 2012[2]. Figure 2.2 (c) illustrates some visitors who witnessed an AR show of the wildlife in the Rotterdam Central.

The two primary requirements for AR application to work realistically are:

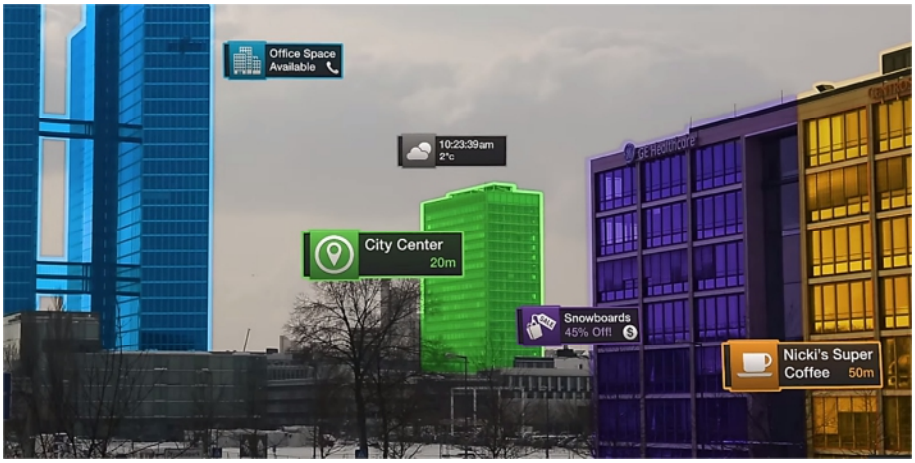
- Real-time response to the images streamed by the camera (most applications designed for human require a frame rate of 30 FPS [15]); in addition to the ability to find and render the corresponding virtual models.
- A correct registration of the augmenting virtual data on the real scene. Otherwise, the misplacement of the extra information normally renders the application entirely useless.

2.2 Virtual Reality (VR)

Unlike AR, the Virtual Reality concept aims at realistically immersing the user in a computer-generated world (virtual world). Such an immersion can be achieved through visual, auditory, and sometimes tactile sensors. The sensory data serve as an extension of the human senses in order to perceive the virtual world. It provides an intuitive, powerful and easy framework for human/computer interaction. As a result, the user would be able to interact with the simulated environment in the same way they act in the physical world, without any requirement to learn how the complicated interface operates.

The ultimate purpose of the VR discipline is a complete isolation of the users from their familiar surroundings. As a result, they experience the illusion of a faithful interaction with the computer generated environment.

[2] http://www.wildlife-film.com/films.html. 2015



(a)

(b)

(c)

Figure 2.2 Some augmented reality applications. **(a)** AREngine. **(b)** Wildlife film festival. **(c)** Rotterdam Central

Four requirements are crucial for any VR system to work correctly [1]:

- Blocking out any contradictory sensory impressions from the real world.
- The graphical rendering system should be able to refresh the 3D view at a minimum rate of 20 frames per second.
- An excellent tracking system that continually reports the position and orientation of the user's head and limbs.

2.3 Mixed Reality Applications (MR)

In this section, some concrete examples of AR/VR applications are presented to demonstrate the importance of such concepts in practice.

2.3.1 3D Rendering

The most straightforward application of AR is the visualisation of computer-generated objects on real image data. W. Qi et al. [16] presented a vision-based AR system for interaction with the rendered scene. Brenner et al. [17] proposed another application named *GeoScope* designed for augmenting the images of cities, landscapes and buildings.

Figure 2.3 illustrates an example of a VR application of the flight simulator regarding the *Boeing 737*. The space where the aircraft flies is rendered on a spherical zonal screen. The simulated model contains two seats one for the pilot and another for the co-pilot. The cockpit of the aircraft is similar to that of a real airplane. The instrument panel conveys faithfully the results of physical avionics.

The entire setup is mounted on a motion control platform that gyrates within a three-storey space. The vehicle's dynamics and simulated motions are highly accurate. The platform also produces realistic engine as well as wind sounds.

(a)

(b)

Figure 2.3 *Boeing 737* Flight Simulator. **(a)** View from the inside. **(b)** View from the outside

2.3.2 Medical

AR can very efficiently facilitate the work of doctors by augmenting ultrasound images with additional information. Bajura et al. [18] proposed an application where the practitioner views a volumetric image of a foetus overlaid on the abdomen of a pregnant woman. The resulting image looks as if it were taken from the inside of the abdomen.

The availability of digitised 3D human models has opened a new perspective on the study and the practice of medical science. Recent innovations have brought

into existence a realistic simulation of the whole physiology with regards to the human body [18].

Surgery is the most targeted medical application where the surgeon is immersed in a virtual operating theatre using a Head Mounted Device (HMD) and Data Gloves. The latter can operate a patient through this virtual scene. Such a facility is now possible and even tele-operational, where the surgeon is distant from the patient [18]. On the other hand, the best way for students to learn how to operate has become risk-free by practising on a 3D virtual patient instead of a real one. As a result, medical errors, frequently made by beginners, have been significantly reduced [18].

2.3.3 Military

Livingston et al. [19] demonstrated how beneficial AR is for military operations in urban terrains. *Arcane*[3] developed a technique to display an animated terrain that can be used in military intervention planning.

VR solutions for training recruits are increasingly involved in modern armies. The simulator renders a 3D environment where soldiers operate as if they were in a real battlefield. To this end, they wear HMDs, and their respective moves are captured by a high accuracy tracking system.

The solutions that exist in the market can even allow a group of soldiers to work cooperatively with mock weapons based on real prototypes.

2.3.4 Robotics

Suzuki et al. [20] claimed that robotic image-guided surgery is one of the most prominent domains where the asset of AR can be clearly seen. Another tele-operated manipulation based on the augmentation of the images of the subject scene was presented by Tachi et al. [21].

[3] http://www.arcane-technologies.com/en/home.html?section=products. 2015

2.3.5 Civil Engineering

An AR architecture was proposed by Webster et al. [22] to improve the methods of construction, as well as the renovation of architectural structures. The contribution of AR technology is constituted in the exploration of the relationship between the perceived architectural structure and the designed plan.

2.3.6 Manufacturing

Reinhart et al.[23] proposed an approach to integrate AR technology in a product assembly line. Animated graphical assembly instructions and sequences are displayed upon request of the workers. The virtual enhancements are overlaid on the products to facilitate the work expected from the operator.

Altair's *HyperWorks*[4] is a good example that illustrates how useful the VR concept is to simulate a Virtual Wind Tunnel. This solution aims at providing a better simulation technology and user experience. It can also faithfully predict an automobile's aerodynamics at a high frame rate for a better-performing, and more fuel-efficient vehicle.

2.3.7 Tourism

The interactive augmentation of images of historical and cultural sites with AR technologies allows greater ease and better information for the visitors. Such an improvement in the experience of tourists has already gained a significant niche in the market of smartphones as has been claimed by Fritz et al. in [24].

2.4 Performance Parameter of the MR Systems

The most important parameters to assess an MR application are listed below [25]:

[4] http://www.altairhyperworks.com. 2015

- *Update rate*: It defines the frequency of sampling and rendering measured either in Hertz (Hz) or Frames per Second (FPS). A higher frame rate is always better for smoothness and precision.

- *Latency*: It is the amount of time elapsed between the end of the request and the beginning of result transmission. A smaller latency yields a higher comfort of operation.

- *Accuracy*: It is the measure of the distance separating the reported result from the ground truth. It is measured in the same units as the observed variables. All the applications seek a higher accuracy.

- *Resolution:* The tiniest perceivable change between two different samples, measured in the same units as the variables. A higher resolution leads to a smoother and more precise tracking, but also larger amounts of data.

- *Working volume*: It is the volume defined by the field of view regarding the tracker. The tracked entities cannot be detected outside of this space.

2.5 Virtual Model Design

In this section, an example of a virtual 3D model regarding the *HL5 Autonomous Robotics Lab* is illustrated. All the experiments in this thesis have been carried out in this indoor environment. The purpose of 3D design is the representation of the real arena with a faithful virtual scene.

An additional motion capture framework is required in order to compute the position and the aspect of the objects for a real-time interaction and a realistic perception. The more accurate the 3D representation and object tracking become, the less erroneous the interaction with reality will be [1].

As has been already shown in Chapter 1, the objective of this study is the improvement of the quality of real-time 6 DOF rigid-body tracking and 3D

registration in indoor environments. The indoor space is assumed to contain multiple ground and aerial vehicles, as well as some obstacles.

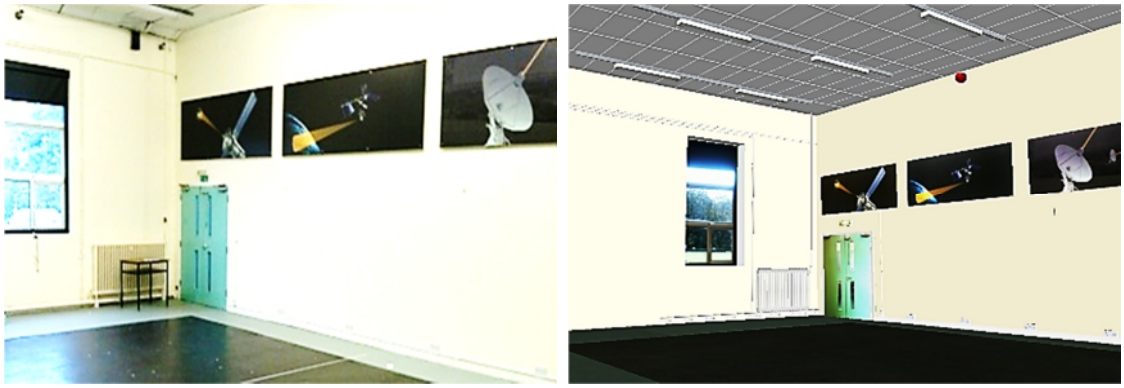
Figure 2.4 and Figure 2.5 depict some samples taken for the 3D model of the lab. The virtual representation is based on real measurements of the different components. The design was conceived and built with Google™ SketchUp[5].

The 3D lab is regarded as a static virtual environment (it does not change its geometric properties over time). This model can be populated with other 3D models of moving robots.

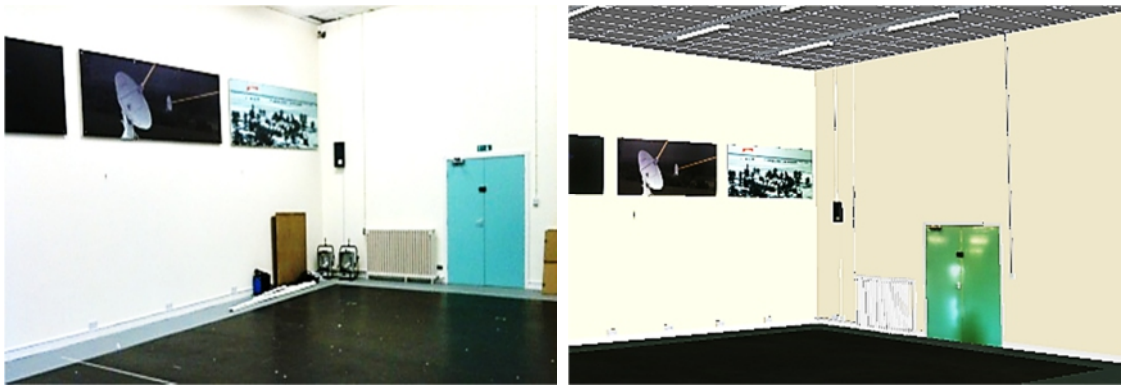
The process of 3D CAD modelling is done for every prospective item that may exist in the arena (robots or obstacles). These models are designed and stored in a database along with their necessary information such as the scale and the initial pose.

Figure 2.4 A perspective view of the 3D model for the lab

[5] http://www.sketchup.com/products/sketchup-pro. 2015



(a)



(b)

(c)

Figure 2.5 Real images on the left and their 3D correspondent on the right. **(a)** View 1. **(b)** View 2. **(c)** View 3

All the measurements of the items that co-exist in the same virtual environment, either static or dynamic, should be proportional to their real counterparts. Otherwise, a contradiction could occur and, therefore, lead to a mismatching between the virtual and the real views.

Only an indoor environment has been considered because of the limitations of the utilised sensors. However, the same methodologies and principles remain valid for outdoor scenarios. A good example of such an indoor/outdoor instance can be seen in the space exploration robot [26]. Nevertheless, some considerations should be taken into account to deal with the nature of the terrain and the opposing forces.

2.6 Image Registration Importance for MR Applications

Sight (of the five human senses) contributes 70% to the overall human sensing data [27]. For this reason, image data has a significant importance in MR applications. An entire scientific field has been dedicated to the research on how it would be possible to mimic the biological vision systems [28]. This field is *Computer Vision*, where researchers try to make computers understand their surrounding 3D real world through image data.

For mixed reality applications to work properly, the objects in the real and the virtual worlds must be properly aligned with respect to each other. Alternatively, the illusion that the two worlds coexist will be compromised (see Figure 2.6). For example, recall the medical application where the surgeon operates on the patient, Section 2.3.2. If the virtual tool (surgical needle) is not placed where the target area is located, the surgeon will miss the objective, and the entire operation would consequently fail. Such an erratic action may result in severe damage to the organ and the tissue. Therefore, without an accurate registration, MR concept would not be acceptable for any serious application.

On the other hand, Virtual Environments suffer less from erroneous alignment. This is due to the fact that the user sees only the virtual objects in a virtual environment. When the user is wearing a closed view HMD and hold up their real hand. The virtual hand should be rendered exactly where they expect it to be. If the virtual hand is wrongly placed by a few millimetres, the user may not notice unless actively looking for such an error. However, registration errors may cause seamlessness and conflicts between the different senses. As a consequence, the user can experience motion sickness, and the immersion becomes insupportable [29].

Another phenomenon known as Visual Capture [30] makes it even harder to detect such a registration error. The event occurs when the brain tends to believe what it sees rather than what it touches or hears. That is, the visual information overrides the remaining senses. However, if the errors are systematic, the users can even adapt to the new environment after a long exposure [30].

Figure 2.6 An example of an erratic mapping of texture on the 3D mesh of the terrain. Courtesy of *Google Earth*[6]

[6] http://www.google.co.uk/intl/en_uk/earth/. 2015

2.7 3D Registration Strategies

Feature correspondence is considered as the seminal basis for the 3D registration problem [31]. The latter can be either *sparse*, i.e. by just considering some key points or *dense* when all the points are involved.

2.7.1 Sparse Registration

The sparse or feature-based approaches (Figure 2.7) can be applied to the estimation of the rigid body motion of point clouds according to the following sequence:

- Extracting interest points using a 3D feature extractor. The result of this step (feature extraction) is a list of source/target key points $\boldsymbol{S_{kp}}, \boldsymbol{T_{kp}}$, respectively. These points have some special characteristics that enable them to be more informative than standard points.

- Computing the respective feature descriptors within their neighbourhood, these descriptors can have different strategies and sizes. For feature matching to work correctly, the descriptor should be similar to its counterparts captured at various viewpoints, lighting conditions or other sensors. On the other hand, the information embedded in a given descriptor should be sufficiently dissimilar to alternative descriptors'.

- Matching the descriptors in order to find a list of pairs of es $[\boldsymbol{S_{kp}}, \boldsymbol{T_{kp}}]$. The matching algorithm evaluates the distance between two descriptors. The latter can be matched in an easy and straightforward brute force manner (all against all; obviously time costly). Alternatively, KDtrees [32] can be used to restrict the search to just the neighbouring proximity. The best match corresponds to the minimal distance between two descriptors.

- The resulting pairs are used to minimise a cost function defined by an error metric that separates the source and the target sets of points. This error is written in the L_2 norm form [33].

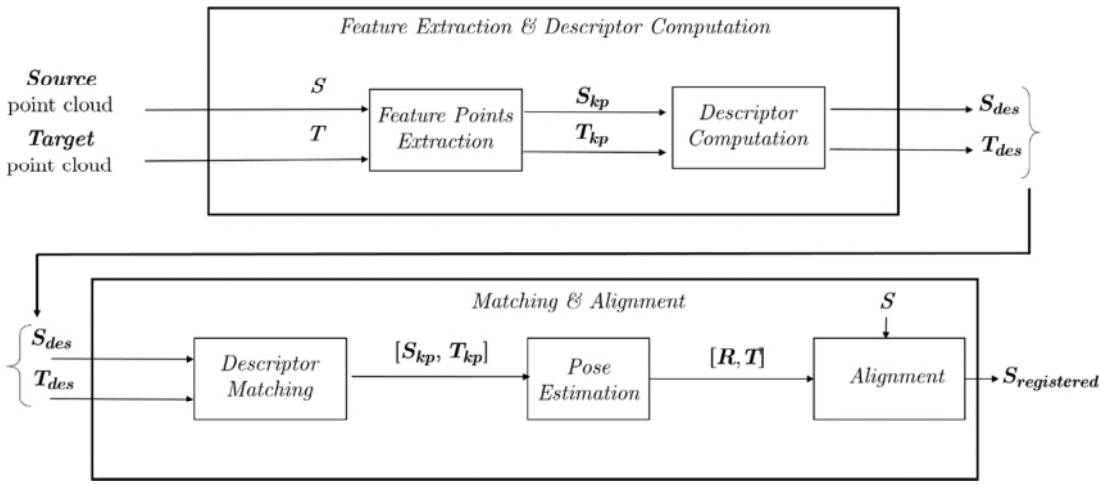


Figure 2.7 Sparse Registration

2.7.1.1 Advantages of Sparse Registration

The most plausible advantage of sparse registration is the relatively low computational burden compared to the dense one. Moreover, if features' positions are precisely determined, i.e. no outliers among the key points and no mismatches in the list of correspondences, the sparse registration outputs its best possible result.

Another advantage is the high likelihood of preventing local minima. Dense methods, on the other hand, suffer severely from such a problem.

2.7.1.2 Drawbacks

As was shown above, sparse registration can be solved optimally if the key points are outlier-free, and the correspondences between the source/target features are correctly determined. However, in practice, it is not possible to totally eliminate noise from the measured points. Hence, other measures to cope with the undesirable circumstances should be taken into account.

Another challenge for the sparse methods is the deficiency in good quality features in the poorly-textured 3D areas. The sparse registration can also be challenged with near-degenerate configurations due to the geometry of the points (collinear).

2.7.2 Dense Registration

Unlike the sparse, dense registration takes into account all the available data to compute the rigid body transformation. If the correspondences between points are known, a closed form solution is preferable. However, the determination of the correspondences itself is the real issue in the registration process.

2.7.2.1 Closed Form Solution

The closed-form approach has the advantage of being achieved in a single iteration. Unlike iterative methods, this approach does not require a good initial guess. It therefore spends just one iteration as shown in the diagram of Figure 2.8, to find an excellent result. The single iteration description can be confusing, however. It concerns just the computation of the eigenvectors of a 4×4 matrix. Nevertheless, prior to that task, the matrix must be built with the combination of the sums of products regarding the coordinates of the points. The complexity of this operation is linear to the number of points $O(N)$. In addition, a quartic equation needs to be solved to find the eigenvalues.

Horn presented a solution to this problem. The rotation matrix was initially represented by a unit quaternion vector [34], then an orthonormal matrix [35].

2.7.2.2 Iterative Solutions

When the correspondences between points are unknown, the iterative registration is the best qualified to align the views progressively. To this end, the method iterates over the following steps shown in Figure 2.8:

- At every iteration $\boldsymbol{k}$, the process of alignment starts with the estimation of a set of correspondences that are not necessarily correct. For every source point, the algorithm determines the closest neighbour in the target

cloud. The result of such a strategy is initially coarse, but it becomes gradually finer.

- Based on the estimated correspondences, an optimisation algorithm minimises the cost function, which is the L_2 distance between the sets of points. The result of this minimisation is the $\boldsymbol{k}$-th rigid body transformation $[\boldsymbol{R}, \boldsymbol{T}]_{\boldsymbol{k}}$.

- Using the estimated transformation, the source cloud is mapped to $\boldsymbol{S_{k+1}} = \boldsymbol{R_k S_k} + \boldsymbol{T_k}$.

- The convergence of the algorithm is based on checking whether the error (distance between the sets of points after registration) is below a certain threshold. Otherwise, the number of iterations should not exceed a predefined maximum value. Both parameters are fixed by the user. If one of the conditions is breached, the algorithm terminates immediately. The result would, therefore, be the last registered source. Alternatively, it reiterates from the beginning with the transformed source instead of the initial one.

2.7.2.3 Advantages

The iterative dense registration does not require any feature extraction, descriptor computation or matching. As a result, it is simpler and easier to implement. In addition, the closed form solution gives the lowest error when the correct correspondences between source and target data are available.

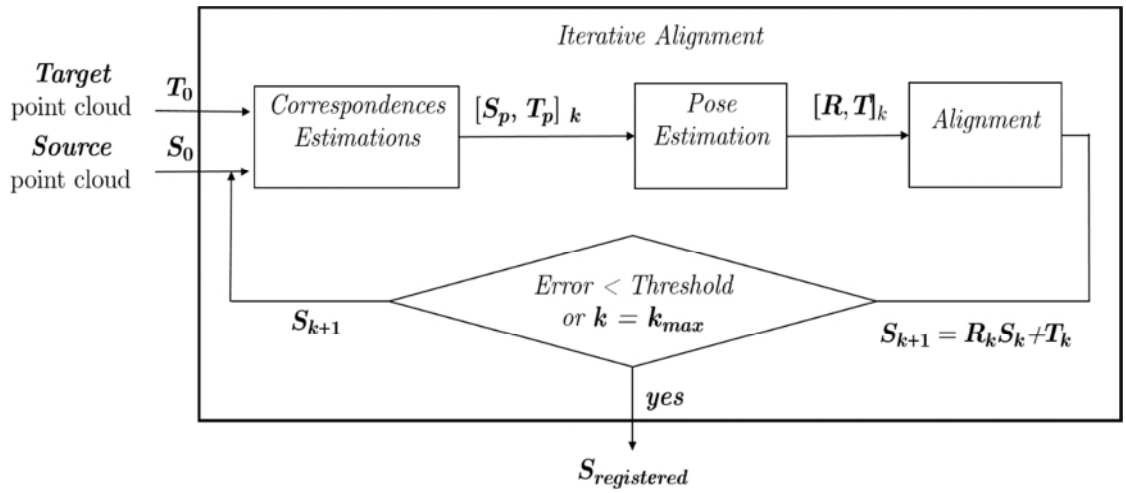


Figure 2.8 Dense registration

2.7.2.4 Drawbacks

The rough criterion in the association of the points into the same pair of correspondences increases the probability of the algorithm being trapped in a local minimum. Another shortcoming of dense methods is slow processing. The long time required to deliver the result of a relatively small point cloud (1000 points) renders the iterative choice unsuitable for real-time applications. Nevertheless, recent advancements have leveraged the computational power of the GPU to reduce the time elapsed in the search for correspondences [36].

2.7.3 A Compromise between Quality & Time

In practical real-time registration scenarios, such as those required for MR applications, it is always recommended to leverage both methods (sparse and dense) for a more optimised result and processing time. The sparse registration is good at preventing local minima and is time-efficient. However, it is worth considering an additional refinement step with the dense approaches to further tighten the quality of alignment. An iterative registration solution has the ability to achieve this extra smoothing.

The outcome of the sparse registration stage would serve as a good initial guess for the dense solution. As a result, the dense alignment would converge within a

few iterations to an even better result. This strategy helps in the reduction of cumulative alignment errors.

2.8 Rigid-Body Tracking

Accuracy of detection of the position and orientation of the real entity that interacts with the virtual data is crucial for a decent MR application. The objective expected from the tracker is the estimation of the 6 DOF regarding the objects. After that, it will be able to position them in the simulated world.

The available tracking systems can either provide the *absolute* (related to a single common reference frame) or a *relative* pose (linking one view to another). Here the pose consists of the position, responsible for describing the translations undergone by an object, and the rotations around the three axes.

2.8.1 Real-Time 6 DOF Tracking Methods

Despite the various means available for the determination of the pose of objects, vision-based tracking is still the most solicited and addressed in the community [37]. The dominance of this kind of implementation is due to the ubiquitous availability of cheap cameras. The term vision refers to the reliance on image data and computer vision algorithms to obtain the 6 DOF attribute. The coordinates $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ serve for the localisation, the aspect (orientation) is defined by the angles $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$, Figure 2.9.

Two main strategies can be followed to infer the relative position and orientation of a given object in the scene: The former is called *Marker-Based,* as its name indicates, it is based on several markers of known properties. Indeed, these markers can be easily distinguished from their surrounding [38]; the latter, on the other hand, is called *Markerless*. It does not assume the existence of any known markers.

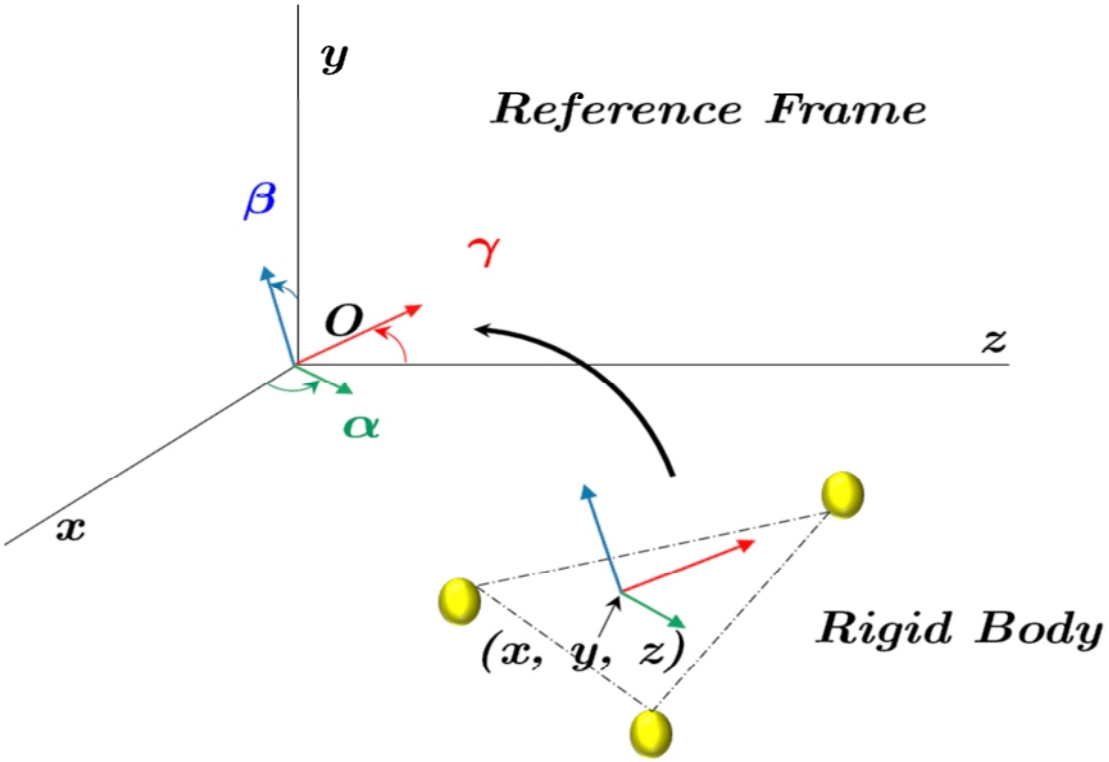


Figure 2.9 6 DOF of a rigid body

It rather detects the objects of interest based on their shape and appearance (texture) [39].

2.8.2 Marker-based Tracking

Marker-based tracking (Figure 2.10) uses artificial markers, i.e. they do not belong to the scene. These markers are attached to the body surface. They also have some specific properties that make them easy to distinguish from the remaining neighbourhood. The system, therefore, triangulates the 3D position of a marker between two or more calibrated views to provide the overlapping projections. The result is the (x, y, z) position. Nevertheless, at least three markers are required to further estimate the angles (α, β, γ) that constitute the orientation.

The markers can either be *passive* or *active*. The passive markers are coated with a retro-reflective paint to allow the IR cameras to detect them quickly. These cameras are endowed with a ring of IR-LEDs in order to illuminate the

markers, Figure 2.11 **(b)**. On the other hand, the active markers generate their light to allow the ordinary colour cameras to see them clearly.

Figure 2.10 depicts the flow of data starting from the acquisition, to the actual pose estimation. The input image $\boldsymbol{I}$ can either be visible or IR. In order to extract the markers from the image, several thresholding approaches can be used. The latter segment the areas where the markers reside from the remaining background. In professional tracking solutions, the thresholding operation can be accelerated with filters running in the camera itself. The result is a binary image $\boldsymbol{I_{thresh}}$ that contains the white spots representing the markers. Every marker in $\boldsymbol{I_{thresh}}$ is then represented with a Gaussian distribution whose mean is its centroid.

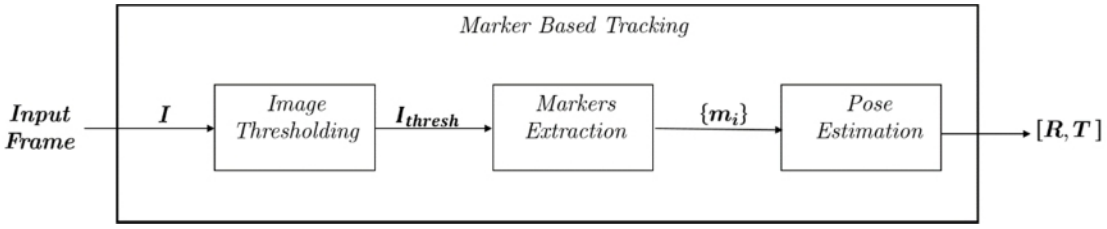


Figure 2.10 Marker-based tracking pipeline

The 3D position of the markers is computed from the triangulation over multiple cameras. To this end, each marker must be uniquely identifiable in order for its inter-camera matching to be achieved correctly.

2.8.2.1 Case Study: OptiTrack Tracking System

OptiTrack (Figure 2.11) is a motion capture system manufactured by NaturalPoint[7]. This motion capture system provides a high precision tracking for commercial, industrial, gaming as well as research applications. It has the ability to track rigid bodies, full body motion and face expressions.

[7] https://www.naturalpoint.com/. 2015

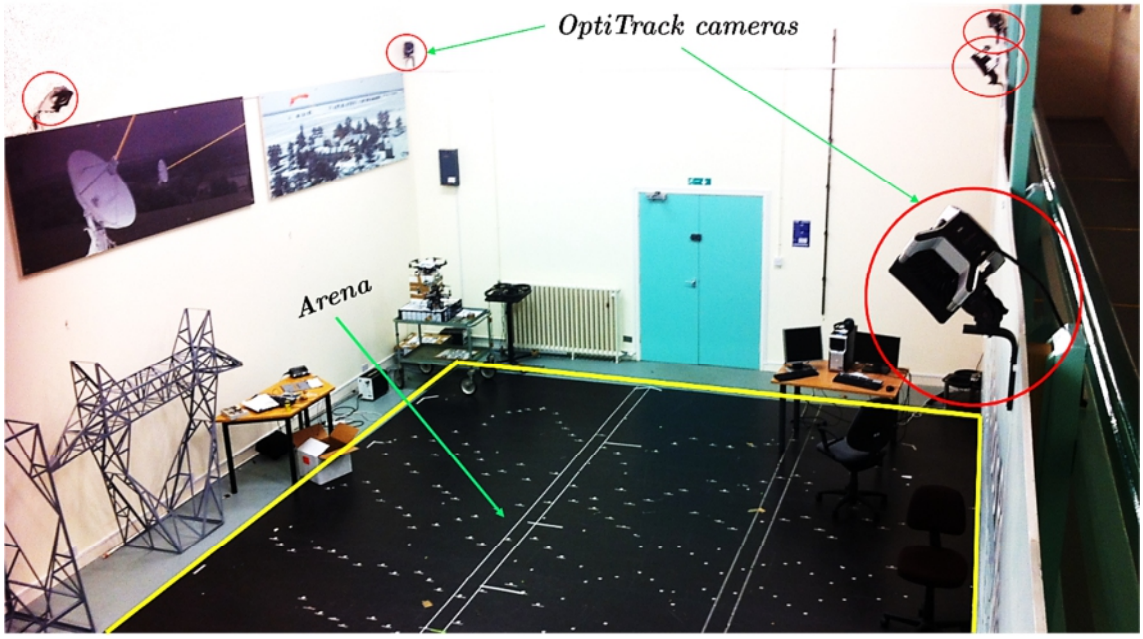
OptiTrack system contains several IR cameras placed at different viewpoints. The cameras allow the simultaneous recording of the movements of the markers attached to subjects. The system computes the three-dimensional position of every marker. The configuration for the *Autonomous Robotics Lab* encompasses six *Prime 17W* cameras. Each has a resolution of 1.7 MP (1664×1088 pixels), with a broad field of view 70°.

In order to track an object in the 3D space, the user must define a set of at least three markers for each rigid body. It is conventional for the markers to be spherical. Such a choice is motivated by the fact that the sphere conserves the same appearance after being rotated, translated or even scaled. This property makes its centre of mass stable and reliable.

When the number of markers used to define a rigid body exceeds three, the accuracy will increase. Nevertheless, the use of too many markers may cause redundancy. Their topology should be spacious enough. An inter-markers' distance of less than 6mm is very likely to result in an inaccurate estimation. The same problem happens when the extrinsic calibration of the cameras is poor. The quality of calibration directly affects the correctness of triangulation. In other words, the rays from the centre of each camera to the centroid of the marker do not meet at a single point.

An asymmetrical arrangement of the markers is also better to avoid the confusion that may arise because of the symmetry of shape. Moreover, it is worth considering different patterns for different rigid bodies to prevent the swapping and misidentification between them.

The arena where all the experiments of this thesis have been carried out has a volume of $8 \times 13 \times 3\ m^2$. The tracking system in the lab works at up to 360 FPS, which means a very low latency (2.8ms) and a smoother tracking.



(a)

(b)

Figure 2.11 OptiTrack tracking system in the Autonomous Robotic Lab. **(a)** Cameras setup. **(b)** *Prime 17W* camera.

The software that runs the tracking is called *Motive*. The system must be calibrated with the OptiWand and the Calibration Square before the first use. The Calibration Square is necessary to align the virtual axes of Motive with the physical volume. If any of the cameras is moved, the system must be recalibrated. A real-time access to the rigid body pose is available for third-party application through a C++ API.

2.8.3 Markerless Tracking

Unlike the previous strategy, this family of vision-based tracking algorithms does not necessitate any artificial markers. However, these methods rely on image features to detect the objects in the real world. At detection, the captured features are matched against the ones belonging to a known template. The latter can be either standard images taken for the object of interest or artificial CAD models.

The markerless tracking works as follows (see Figure 2.12):

At the acquisition of a new frame $\boldsymbol{I}$, the tracking process begins with the extraction of the key points from the image. The latter can either be 2D or 3D. The result of this step is a list of key points $\boldsymbol{S_{kp}}$. The descriptors that identify the features are then computed $\boldsymbol{S_{des}}$. The descriptors with regards to the template ($\boldsymbol{T_{des}}$) are assumed to have been calculated beforehand and stored in Templates' Database. Subsequently, the captured and the stored descriptors are matched to test whether a given template appears in the current frame. For instance, the feature extractor can be different among current image and the precomputed template. However, the respective descriptors ($\boldsymbol{S_{des}}$, $\boldsymbol{T_{des}}$) should have the same size (e.g. 64, 128, 256, 512 or 1024). This constraint is important to consider because the matching process is mathematically reduced to the evaluation of the distance between descriptors in the same dimension. The output of this stage is a list of correspondences $[\boldsymbol{S_{kp}}, \boldsymbol{T_{kp}}]$.

Based on corresponding features, an optimisation algorithm is launched to align the sets of the captured 3D points ($\boldsymbol{S_{kp}}$) on the templates' ($\boldsymbol{T_{kp}}$). Alternatively, in the 2D case, the features identifying an object are assumed to be resting in a same plane. As a result, a homography between the template and image is computed. This limitation is due to the missing information about the depth of feature points. If the last mentioned are captured with a depth sensor, their mutual three-dimensional coordinates must be obtainable. For this reason, the assumption of the common plane is not necessary for 3D key points. The latter

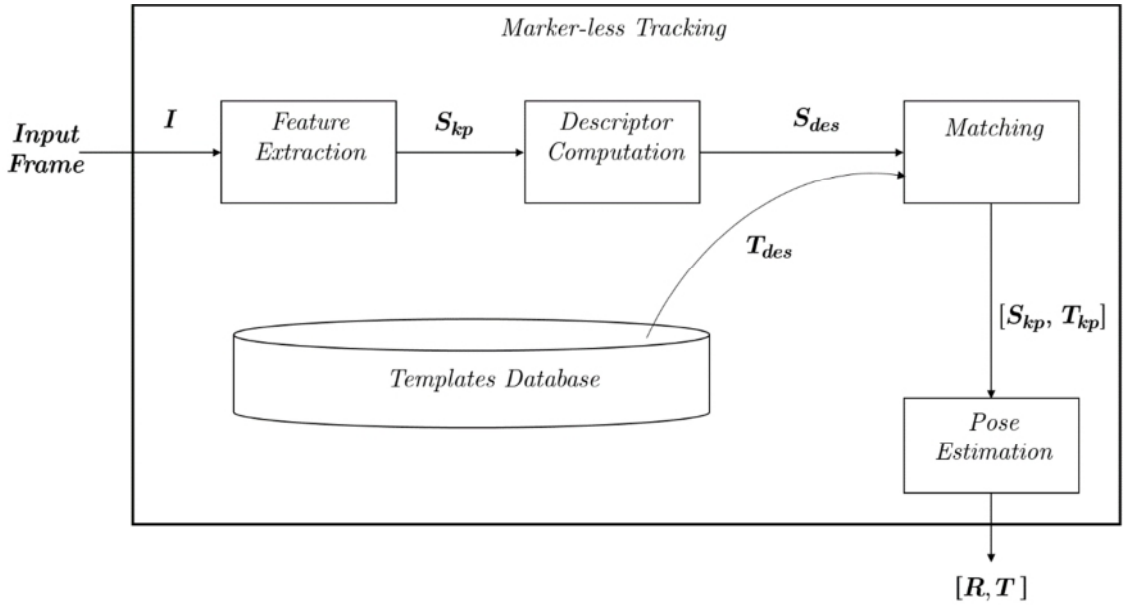


Figure 2.12 Markerless tracking

can be handled by the 3D registration algorithms as has been shown in Section 2.7. The result is a 6 DOF pose that identifies the current position and orientation of the target relative to the template.

As an alternative to the stored template, a frame-to-frame feature tracking is also possible. More sophisticated algorithms are required, however, to correct the drift due the cumulative incremental errors [40].

To sum up, it is possible to track real objects without a requirement for artificial markers. Such an alternative is ideal since it eliminates the hassle of placing and calibrating the markers in the scene. Nevertheless, the computational burden required to match an object in the captured data against all the templates may significantly challenge the real-time responsiveness. In addition, the current body of knowledge is not yet satisfactory to enable a performance equivalent to that of the marker-based tracking. More importantly, the preventing factor from using the markerless methods is their dependence on the captured image for feature extraction. The image data is naturally noisy, fluctuating and sometimes deficient in feature points. Such shortcomings render

the markerless methods currently inappropriate for accurate real-time applications.

2.9 Camera Model and Imaging Geometry

Theoretically, a camera model is a projective mapping from a three-dimensional to another $\boldsymbol{m}$-dimensional space, $\boldsymbol{m} \in \{\mathbf{2}, \mathbf{3}\}$:

$$\pi: R^3 \rightarrow R^m, m \in \{2,3\}\ ; (x,y,z) \overset{\pi}{\rightarrow} (v_1, \ldots, v_m) \tag{2.1}$$

For typical RGB cameras, the source space is the 3D real world, and the destination is the 2D camera plane. Such a mapping can utilise *perspective*, i.e. further objects are smaller than the closer ones. Such a property is due to the rays of light passing through the camera centre before reaching the imager. This projection can be seen in the human vision system as well as in most commercial cameras. Alternatively, the projective mapping can be *orthogonal*. The orthogonality originates from the incident rays of light being orthogonal to the image plane; i.e. they are not forced into the centre of the camera. In this thesis, a perspective (pinhole) camera model is adapted.

2.9.1 Pinhole Camera Model

The pinhole camera is the simplest camera that can mimic real ones. It is merely a cubical box with an extremely tiny hole in one of its faces. Its mathematical model fits most of the cameras in the market, with some considerations that should be taken into account.

The working principle of this camera is as follows:

Let us imagine ourselves in a large dark square room whose frontal wall contains a tiny hole in the middle. When a beam of light is shone from the scene towards the holed wall, a small upside down image of the scene appears on the opposite wall, Figure 2.13 (a).

Figure 2.13 (b) depicts a camera model whose centre of projection is $\boldsymbol{O}$ and the principal axis $\boldsymbol{z}$. The image plane is at focus, i.e. $\boldsymbol{z} = \boldsymbol{f}$. Conventionally, this is

done to avoid the negative sign in the subsequent computations. A 3D point $\boldsymbol{P}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ is projected on the image plane (imager) at $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{y})$. In the beginning, the camera calibration matrix $\boldsymbol{C}$ must be defined. The latter is used to determine the mapping relating the 3D point $\boldsymbol{P}$ to its correspondent on the imager $\boldsymbol{p}$.

From the illustration shown in Figure 2.14, with the similarity of the two triangles, blue and green Figure 2.14 (b), one can deduce:

$$x = f\frac{X}{Z} \tag{2.2}$$

$$y = f\frac{Y}{Z} \tag{2.3}$$

The representation of the point $\boldsymbol{p}$ in the homogeneous space associated to $\mathfrak{R}^{\mathbf{2}}$ is:

$$Z\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & \mathbf{0} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ \mathbf{1} \end{bmatrix} \tag{2.4}$$

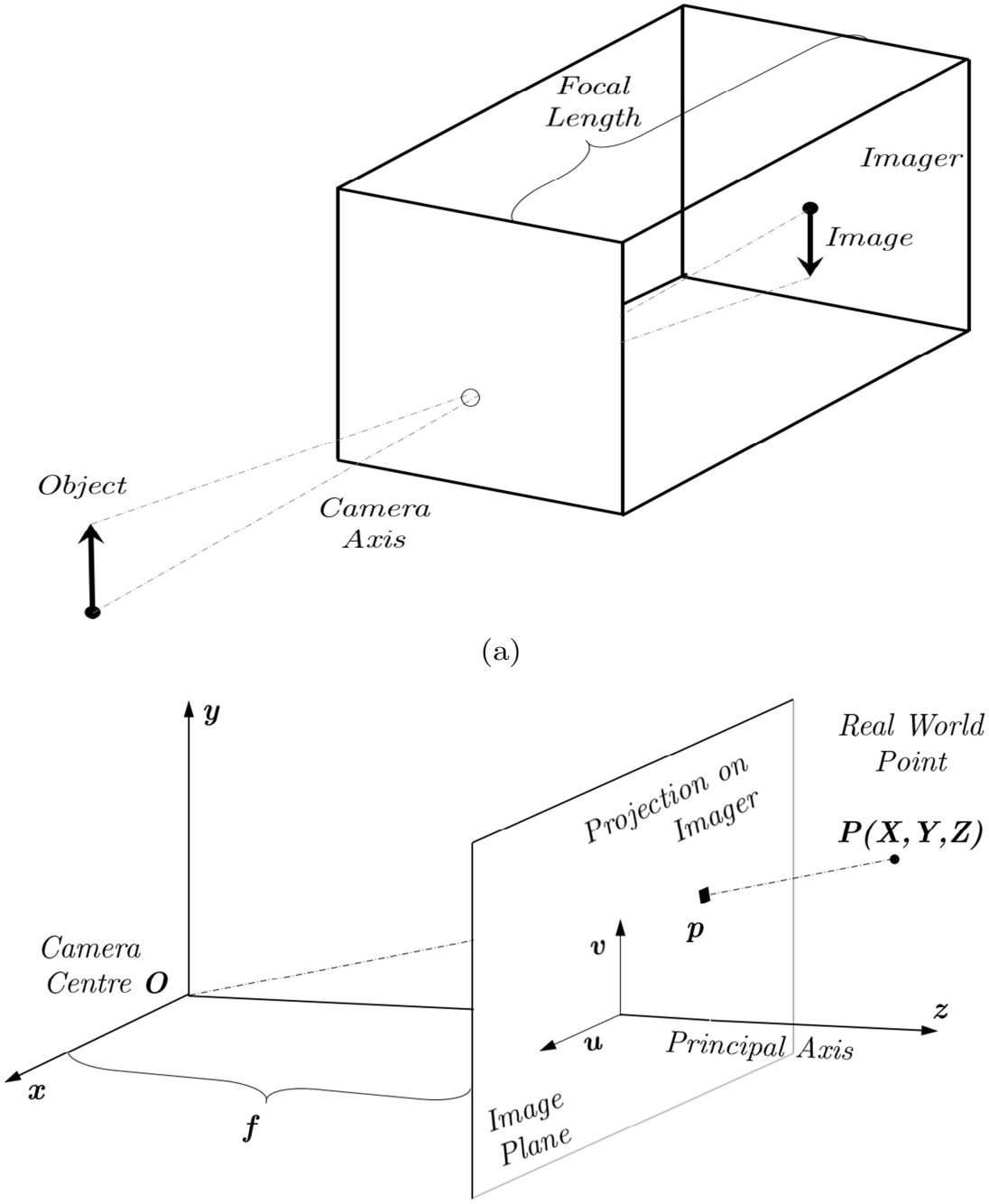


Figure 2.13 Pinhole camera model. **(a)** Model. **(b)** Geometry

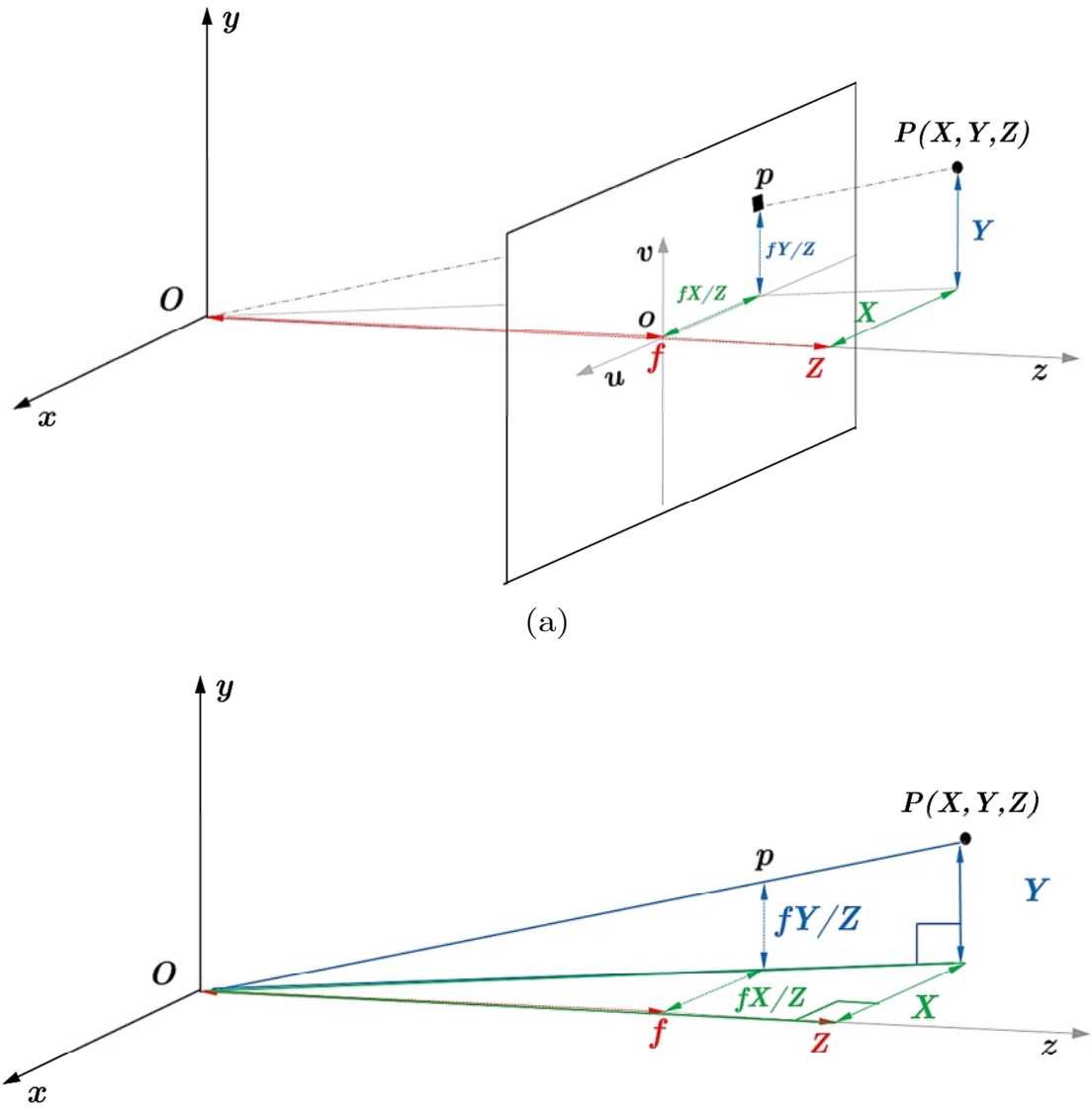


Figure 2.14 Pinhole Perspective Projection Geometry

If the centre of the imager $\boldsymbol{o}$ is different from the intersection of $\boldsymbol{z}$-axis with the plane $\boldsymbol{z} = \boldsymbol{f}$, then $\boldsymbol{p}$ must be translated with the vector $\boldsymbol{o}(\boldsymbol{c_x}, \boldsymbol{c_y})$. The translated point $\boldsymbol{p'}(\boldsymbol{x'}, \boldsymbol{y'})$ therefore becomes:

$$x' = \frac{fX}{Z} + c_x \tag{2.5}$$

$$y' = \frac{fY}{Z} + c_y \tag{2.6}$$

The matrix representation of the Equations (2.5), (2.6) is:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix} \tag{2.7}$$

In practice, the measurements obtained from the image are expressed in pixels. For this reason, it is necessary to map the coordinates of $\boldsymbol{p}$ from $\mathfrak{R}$ to the discrete pixel array. This mapping is a scaling along the axes $\boldsymbol{u}, \boldsymbol{v}$:

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{2.8}$$

Scaling factors $\boldsymbol{s_x}$ and $\boldsymbol{s_y}$ are defined from the shape of the pixel in the captured image.

The focal length $\boldsymbol{f}$ (expressed in metres) also needs to be scaled to pixel unit:

$$\begin{bmatrix} f_x \\ f_y \end{bmatrix} = f \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \tag{2.9}$$

The *Intrinsic* camera calibration matrix $\boldsymbol{K}$ becomes:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

$\boldsymbol{K}$ is an upper triangular 3×3 matrix. It holds the intrinsic parameters of the camera that consist in $\boldsymbol{f_x}, \boldsymbol{f_y}, \boldsymbol{c_x}, \boldsymbol{c_y}$.

If the camera does not have its centre of projection $\boldsymbol{O}$ at the origin of the world frame, or if its axes are not aligned, the rigid body transformation that aligns its frame on the world's frame must also be defined. The latter holds two transformations, i.e. camera translation to the origin of the world frame $\boldsymbol{T}$ $(\boldsymbol{t_x}, \boldsymbol{t_y}, \boldsymbol{t_z})$ and the rotation that aligns the principal axes, 3×3 matrix $\boldsymbol{R}$.

This rigid body transformation, applies a translation followed by a rotation. It is given by the 3×4 matrix $\boldsymbol{\mathcal{E}}$ called the *Extrinsic Parameters Matrix*:

$$\boldsymbol{\mathcal{E}} = [\ R|RT\ ] \tag{2.11}$$

The complete camera transformation can now be represented by:

$$\begin{aligned} K[\ R|RT\ ] &= [\ KR|KRT\ ] \\ &= KR\ [\ I|T\ ] \end{aligned} \tag{2.12}$$

Then $\boldsymbol{p}$, the 2D projection of $\boldsymbol{P}$, is given by:

$$\begin{aligned} p &= KR[\ \mathrm{I}|\mathrm{T}\ ]P \\ &= CP \end{aligned} \tag{2.13}$$

$\boldsymbol{C}$ is a 3×4 matrix called *the Complete Calibration Matrix*. The point $\boldsymbol{P}$ is defined in the homogeneous space associated with $\boldsymbol{\mathfrak{R}^3}$. Its 2D projection in the camera plane is also represented in the homogeneous space associated with $\boldsymbol{\mathfrak{R}^2}$.

To sum up, before using the camera, it is necessary to calibrate it, i.e. finding the matrix $\boldsymbol{C}$ or just $\boldsymbol{K}$ when camera frame is regarded as reference. For some cameras, it is also worth considering lens distortion. However, this parameter could be neglected for small fields of view.

To eliminate distortion from the captured images, Brown's [31] model can be adapted. The radial distortion is due to the shape of the lens. It creates the *fish-eye* effect in the image. The tangential distortion, on the other hand, is caused by the wrong parallelism between the imager and the lens.

The elimination of the distortion is based on the coefficients $\boldsymbol{k_1}, \boldsymbol{k_2}, \boldsymbol{k_3}$, for the radial and $\boldsymbol{p_1}, \boldsymbol{p_2}$ for the tangential distortion as follows:

$$\begin{cases} x^* = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y^* = y(1 + k_1 r^2 + k_2 r^4 + k_2 r^6) \end{cases} \tag{2.14}$$

$\boldsymbol{r}$ is the original radius. $\boldsymbol{x}^*, \boldsymbol{y}^*$ are the result after the elimination of the radial distortion.

The undistorted coordinates $\boldsymbol{x_{cor}}, \boldsymbol{y_{cor}}$ are then computed as follows:

$$\begin{cases} x_{cor} = x + \left(2p_1 x^* y^* + p_2(r^2 + 2x^{*2})\right) \\ y_{cor} = y + (p_1(r^2 + 2y^{*2}) + 2p_2 x^* y^*) \end{cases} \tag{2.15}$$

2.10 Stereo Imaging

The purpose of using two images instead of one is to recover the 3D structure lost during projection. The two cameras must be at different viewpoints, however. For instance, stereo images can be obtained either with two cameras or by a single, moving camera. Stereo imagery is regarded as a basis for the multiple view imaging. In other words, the problem of multiple cameras reconstruction can be reduced to the estimation of the poses between pairs of views [40], the resulting pose is then corrected with a global alignment approach.

Figure 2.15 depicts the different notions related to stereo vision geometry with $\boldsymbol{O_l}, \boldsymbol{O_r}$ being the centres of the left and the right cameras, respectively. In addition, $\boldsymbol{p_l}, \boldsymbol{p_r}$ are the 2D points where the object $\boldsymbol{P}$ is projected on the image planes. Here are some notions related to the stereo imagery:

- *Baseline*: the distance $\boldsymbol{O_l O_r}$ that separates the centres of the two cameras.
- *Epipolar plane*: the plane $\boldsymbol{O_l O_r P}$ defined by camera centres and the 3D point $\boldsymbol{P}$.

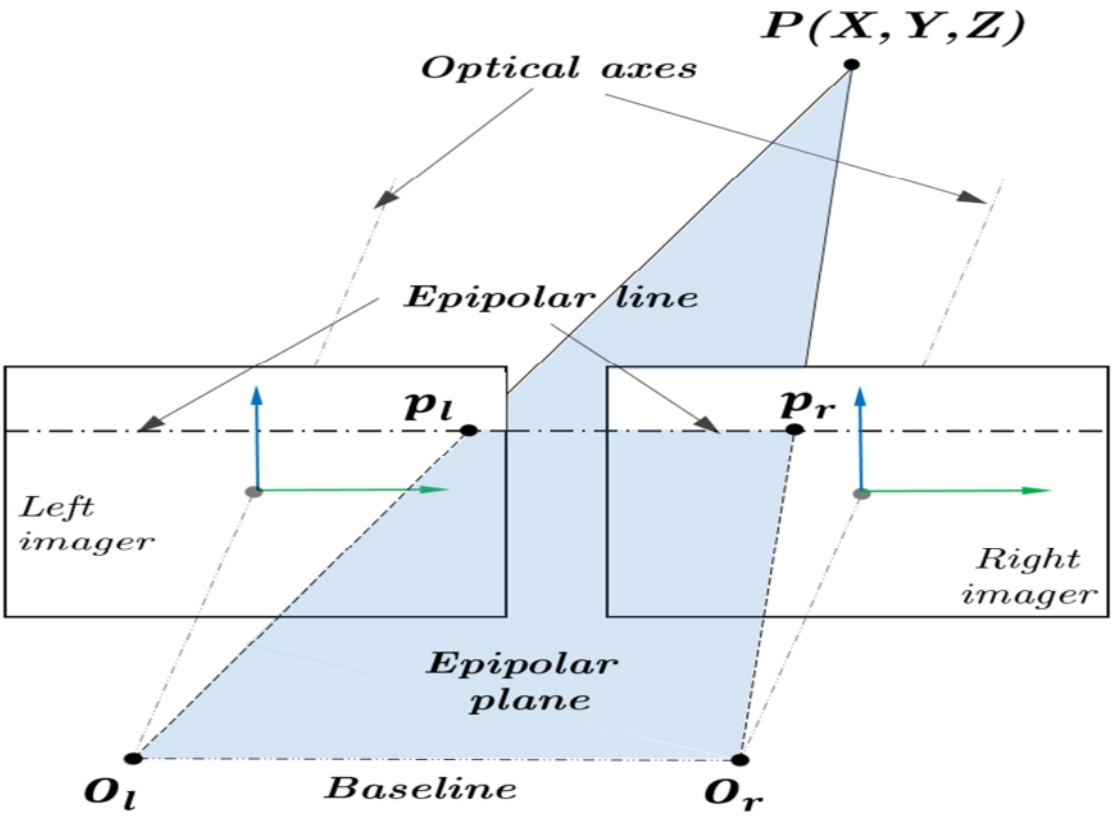


Figure 2.15 Stereo camera model

- *Epipolar line*: the intersection of the epipolar plane with the image plane. The imagers are assumed to rest in this plane.

- *Conjugate pair:* it is any point in the world that is visible to both cameras. Its representation of the stereo setup is a pair of pixels sharing the same epipolar plane.

- *Disparity:* the distance in pixel unit between two corresponding pixels. It is computed after overlaying the images.

- *Disparity map:* the 2D array of the same resolution as the images. Its entries are the disparity readings of all the points observable by both cameras (conjugate pairs).

2.10.1 Depth Recovery

The recovery of the distance between the epipolar plane and the scene (depth or range) is based on the triangulation principle [41]. The 3D location of any visible point in space is restricted to the straight line that passes through the centre of the camera and its projection on the image plane. The depth of the object is recovered from the intersection of the two lines $\boldsymbol{O_l P}$, $\boldsymbol{O_r P}$ passing through the centres of projection and the 2D points in each image, Figure 2.16.

The intrinsic camera calibration matrices are assumed to be known. $\boldsymbol{f_{x\_l}, f_{y\_l}, c_{x\_l}, c_{y\_l}}$ and $\boldsymbol{f_{x\_r}, f_{y\_r}, c_{x\_r}, c_{y\_r}}$ for the left and the right camera, respectively.

Assuming that the right camera is the reference, the mapping from the coordinate system centred at $\boldsymbol{O_l}$ to $\boldsymbol{O_r}$ can be achieved after the determination of the translation $\boldsymbol{T}$ and the rotation $\boldsymbol{R}$, Figure 2.16 (a).

Consider recovering the position of $\boldsymbol{P}$ from its projections $\boldsymbol{p_l}$ and $\boldsymbol{p_r}$, see Figure 2.16 (b):

$$x_l = f_{x_l} \frac{X_l}{Z_l} + c_{x_l} \tag{2.16}$$

$$x_r = f_{x_r} \frac{X_r}{Z_r} + c_{x_r} \tag{2.17}$$

The two cameras are related by the extrinsic parameters:

$$p_r = R\ (p_l + T) \tag{2.18}$$

It results from applying the rotation to the frame of the left image:

$$Z_r = Z_l = Z \tag{2.19}$$

$$X_r = X_l + T \tag{2.20}$$

By replacing in Equations (2.16), (2.17):

$$(x_r - c_{x_r})\frac{Z}{f_{x_r}} = (x_l - c_{x_l})\frac{Z}{f_{x_l}} + T \tag{2.21}$$

$$\left[\frac{(x_r - c_{x_r})}{f_{x_r}} - \frac{(x_l - c_{x_l})}{f_{x_l}} \right] Z = T \tag{2.22}$$

$$Z = T \left[\frac{(x_r - c_{x_r})}{f_{x_r}} - \frac{(x_l - c_{x_l})}{f_{x_l}} \right]^{-1} \tag{2.23}$$

Equation (2.23) outputs the actual depth of the point $\boldsymbol{P}$.

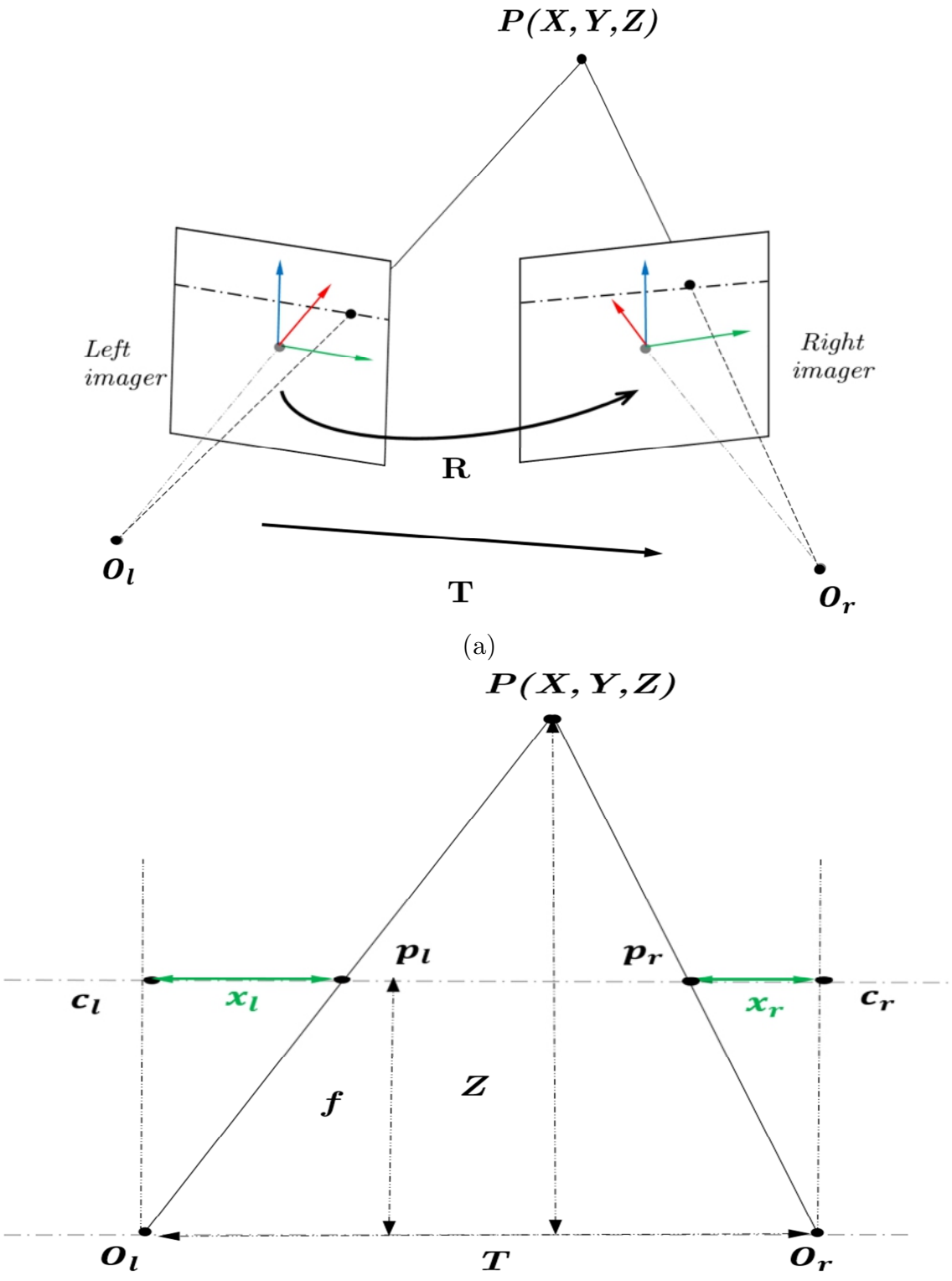


Figure 2.16 Stereo vision geometry. **(a)** Left-to-right alignment. **(b)** Detailed geometry

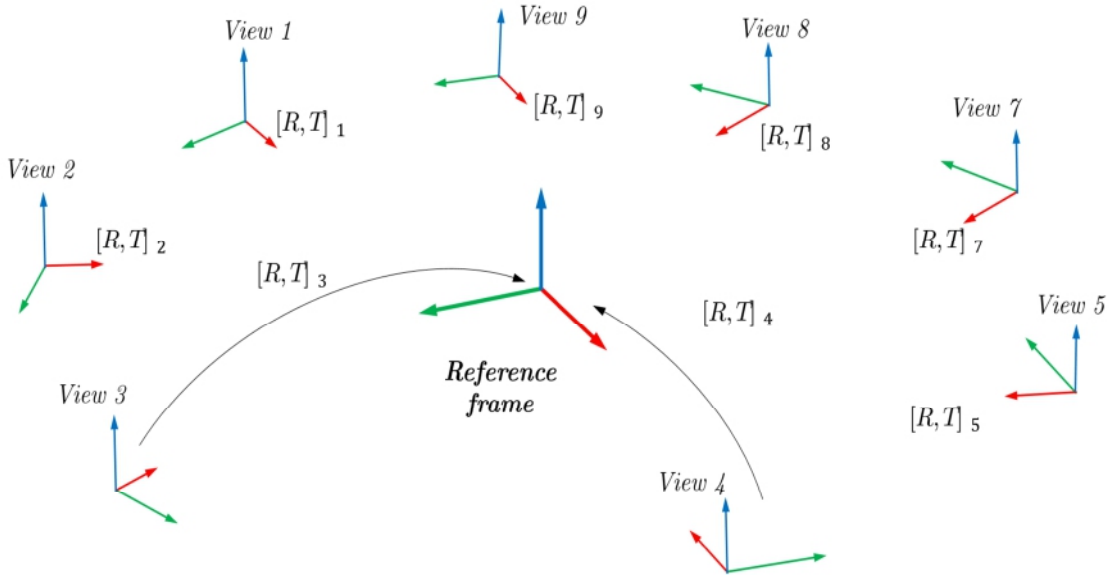
2.11 Multiview Imaging

Since it is not always possible to cover the whole scene with a single camera, accurate tracking requires additional sensors in order to provide a full view of the scene. In addition, occlusions are another issue that can be encountered due to one object obstructing another. Nevertheless, coordinate systems differ from one camera to another. It is therefore necessary to determine the relationships between viewpoints in order to exploit multiview imaging, Figure 2.17 (b).

Image-based 3D modelling does not assume knowledge of the geometry of the scene. The quality of reconstruction increases with the number of the available views [42]. Hence, a large number of views is necessary to accomplish a high-performance of rendering. Similarly, a vast amount of image data needs to be processed. On the other hand, if the number of views is small, the reconstructed scene will accordingly have a reduced quality.

Capturing multiple views from the scene can be achieved in two ways, either by the use of multiple cameras or a single, moving camera. The geometry of the cameras can be determined through an extrinsic calibration procedure. The latter is essential to register their respective frames to the common world frame. A set of planar patterns that hold known features is the most broadly used tool to calibrate the cameras. Alternatively, when the scene contains a sufficient number of key points observable by at least two cameras, the pose can be obtained without a calibration pattern. The procedure of 3D reconstruction from multiple images is called *Structure from Motion* [26].

(a)



(b)

Figure 2.17 Multiview imaging. **(a)** Multiview illustration, courtesy of *Computer Vision Group*[8]. **(b)** Local frames associated with each view

[8] http://vision.in.tum.de/. 2015

2.12 Case Study: RGBD Cameras

Kinect is an RGBD camera that was initially designed for the users of the Xbox[9] gaming console in order to enable them to interact with the game without a controller. The sensor captures the gesture and the speech of the players in real time.

This sensor consists of an (i) IR projector, (ii) an IR camera and (iii) an ordinary RGB camera. The projector projects a set of patterns onto the camera scene. The IR camera captures the projected patterns and sends them to a triangulation module. This module performs the triangulation[10] for the IR stereo setup to infer the depth map, Section 2.10. On the other hand, the colour camera is used to sense the texture of objects. The Kinect sensor is, therefore, an RGBD camera that is capable of simultaneously capturing the depth map and the colour image at a frame rate of 30 FPS.

The manufacturer has already released two generations of this sensor.

2.12.1 Kinect V1

2.12.1.1 Kinect V1 Specifications

This version of the sensor (Figure 2.18 (a)) is able to stream the depth and the colour images with a VGA resolution (640×480 pixels) at a frame rate of 30 FPS. The depth is encoded with **11** bits. The sensor, therefore, provides 2,048 different depth levels. The conventional minimum depth is 0.8m, and the maximum is 4.5m. The Kinect is also able to stream either the depth map or IR image with colour exclusively. Nevertheless, its hardware configuration does not allow the simultaneous streaming of the IR image and depth map.

The sensor can also deliver higher resolution images 1280×1024 at a lower frame rate (12 FPS). Both cameras (IR, RGB) have a field of view 57°

[9] http://www.xbox.com/. 2015
[10] http://wiki.ros.org/kinect_calibration/technical. 2015

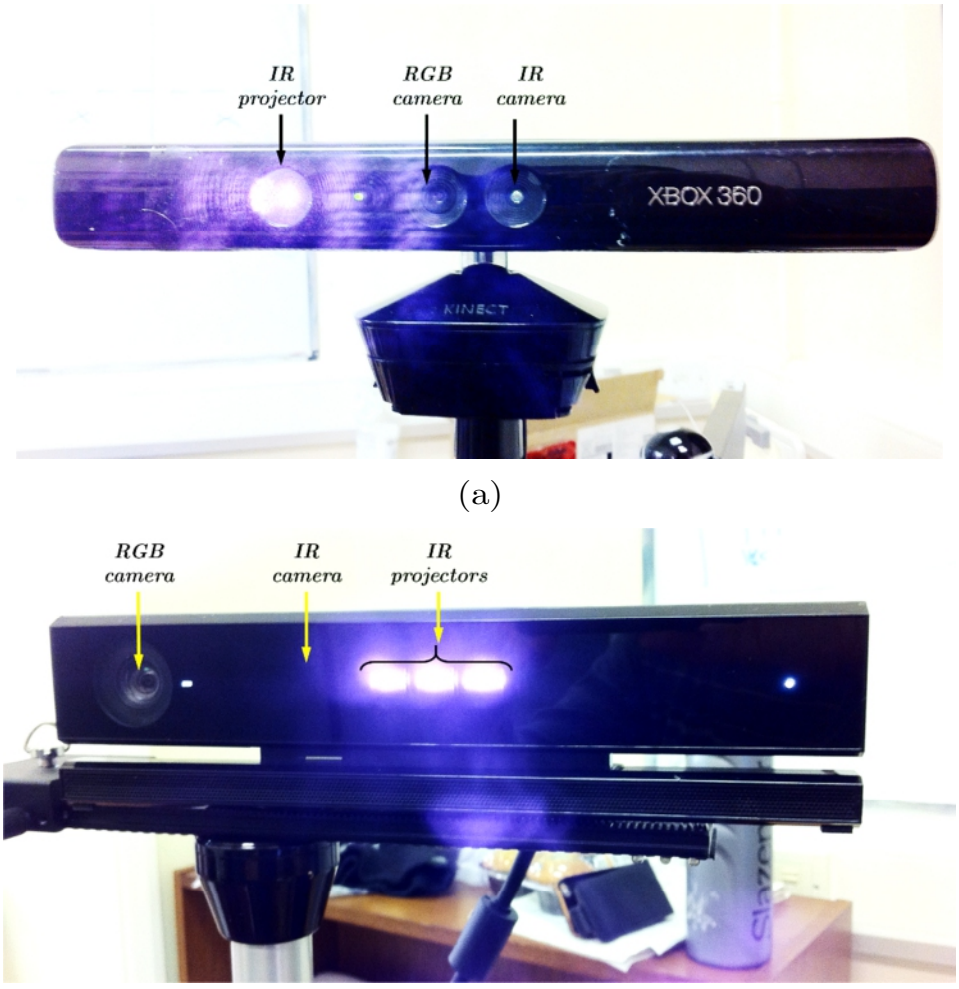
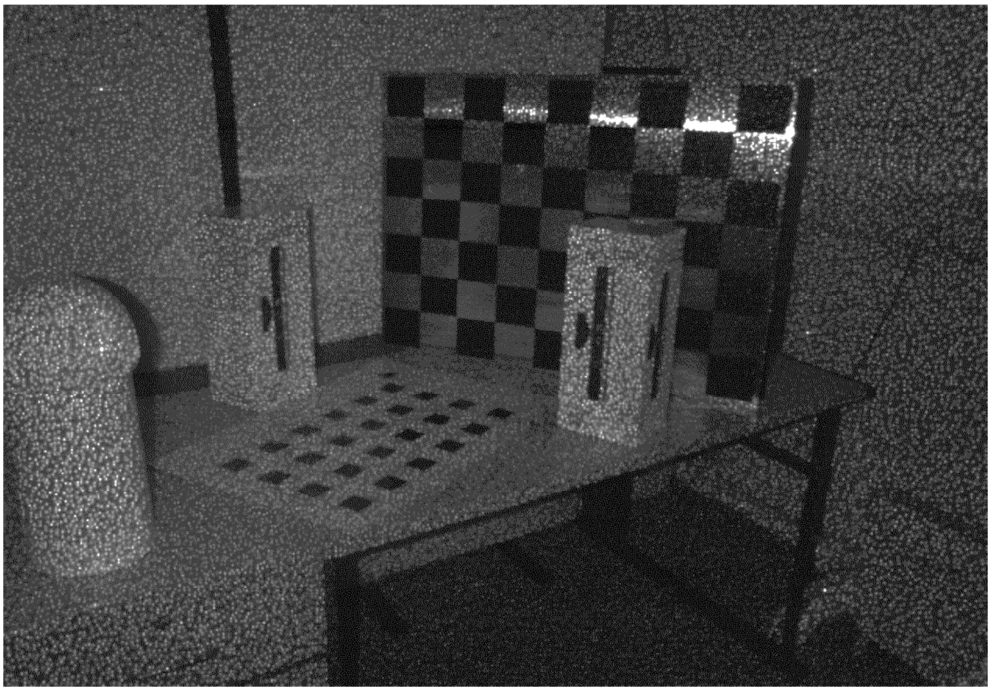


Figure 2.18 Kinect sensor in operation. **(a)** Kinect V1. **(b)** Kinect V2

horizontally and 43° vertically. Figure 2.19 depicts an image sample of the RGB, subfigure (a), and the IR, subfigure (b). The bright dots caused by the projected IR light can be clearly seen.

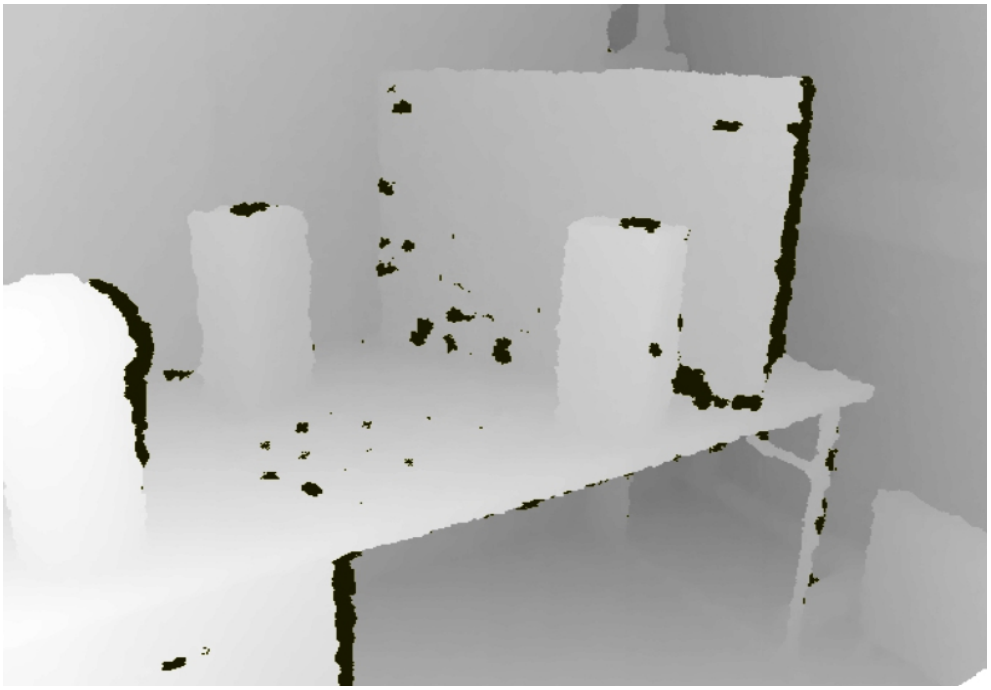
Figure 2.20 illustrates the depth map of the imaged scene. Subfigure (a) represents an intensity image for the depth map. The further away the object gets from the sensor, the darker its representation. On the other hand, Figure 2.20 (b) illustrates the 3D structure of the depth map after a surface reconstruction procedure. The resulting surface is bumpy and less representative of the real scene due to noise and the missing depth readings.

(a)

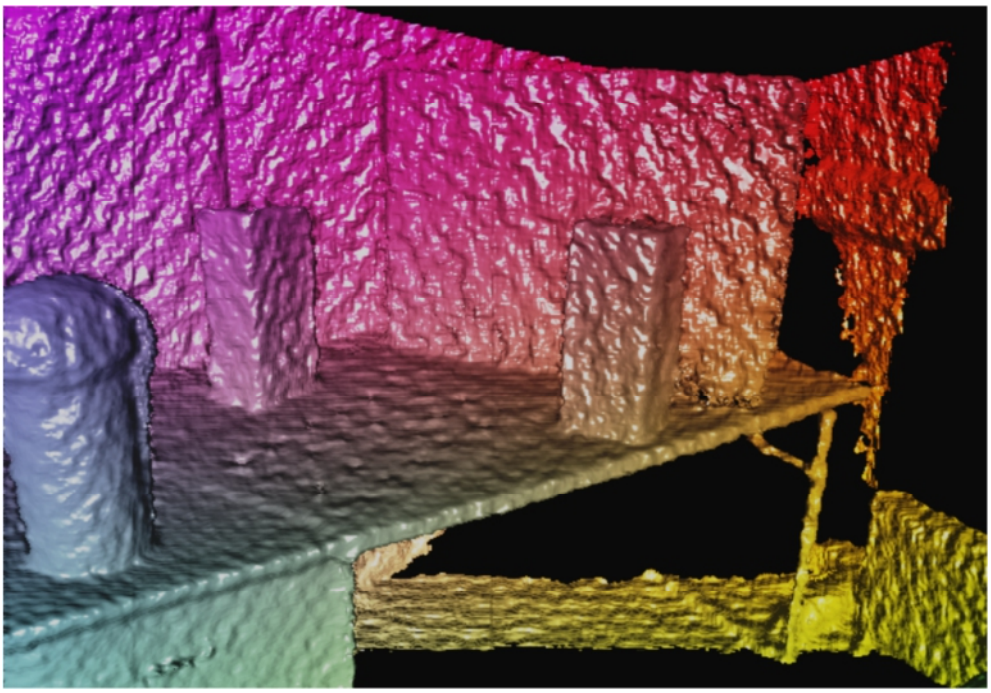


(b)

Figure 2.19 Raw outputs of Kinect V1 camera. **(a)** RGB. **(b)** IR



(a)



(b)

Figure 2.20 Kinect V1 depth output. **(a)** Intensity representation of the depth map (lighter colour for closer objects; black: undefined value). **(b)** Reconstructed surface

2.12.1.2 Kinect V1 Working Principle

Kinect V1 uses a structured light principle to compute depth [43]. The acquisition of the range is done by projecting known optical patterns onto the scene (they play the same role as features do). The deformation undergone by patterns when they strike the surface of the object is captured by the IR camera [44]. The shift between the obtained pattern and the reference stencil yields the actual value of disparity [45].

The resulting depth map does not perfectly represent the continuous detail of the scene due to limited resources [45]. It projects the captured depth map onto a set of discrete parallel planes, Figure 2.21. Consequently, some data, which should be positioned between the planes (according to their real-world $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ coordinates), is either lost or shifted to the closest available level. The accuracy of the sensor is largely affected by this quantisation operation.

This version of the sensor (Kinect V1) was used in the experimental section of Chapters 3, 4, 5. Hence, when we use the term *Kinect* in these chapters we refer to Kinect V1.

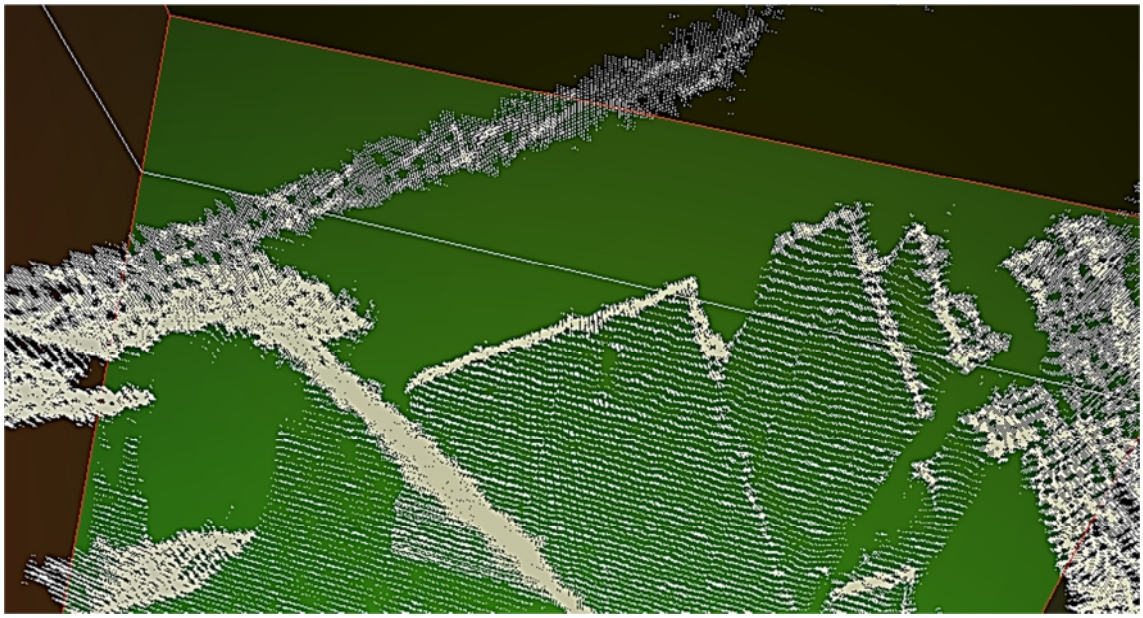


Figure 2.21 Kinect V1 points cloud structure

2.12.2 Kinect V2

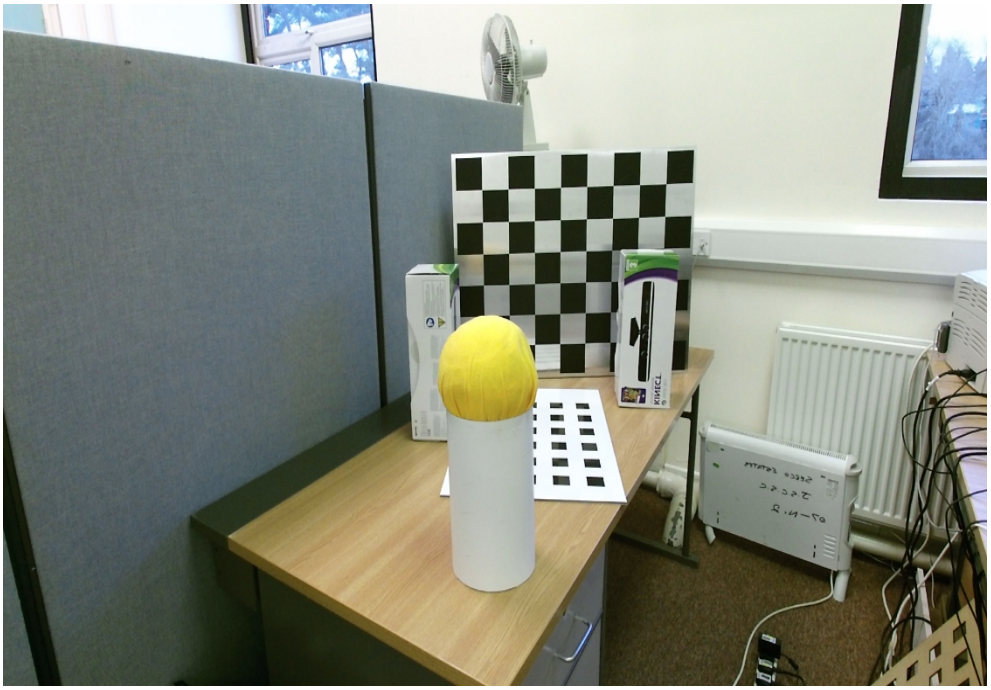
2.12.2.1 Kinect V2 Specifications

The recent version of the sensor (Figure 2.18 (b)) can stream the depth map and the colour image of an HD resolution (1920×1080 pixels) at a frame rate of 30 FPS. The minimum conventional depth is 0.8m, and the maximum is 4.5m. Kinect V2 uses USB 3.0 instead of USB 2.0; i.e. it has a ten times wider bandwidth than the older camera. This improvement enables the Kinect V2 to stream a data load of up to 2 Gigabyte/second.

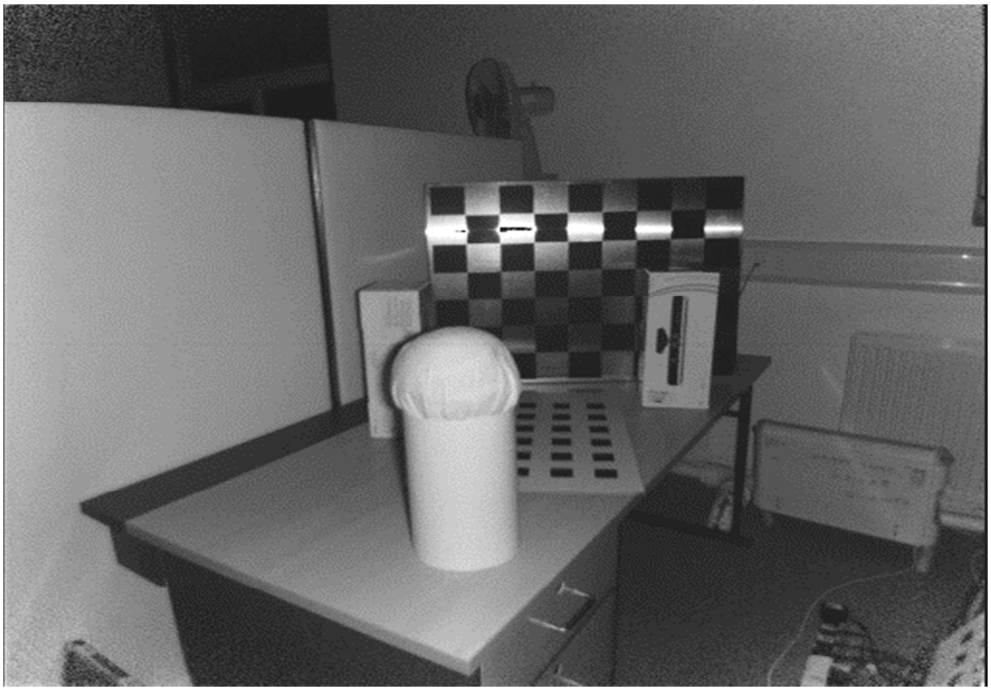
Figure 2.22 depicts a sample of the RGB (Figure 2.22 (a)) and IR (Figure 2.22 (b)) images. It is noticeable that the last mentioned does not contain any bright dots.

Figure 2.23 illustrates a depth map. Its intensity image is depicted in subfigure (a). On the other hand, subfigure (b) illustrates the 3D structure of the depth map after surface reconstruction. The resulting surface is smooth and accurate.

Unlike Kinect V1, the fine details on the edges of the table and the handle of the drawer are clearer. However, the checkerboard is wrongly reconstructed. The original board is flat, but the reconstructed one contains some cubical structures that are a corrupted representation of the flat squares. This phenomenon is due to the specular reflective nature of the metallic board.

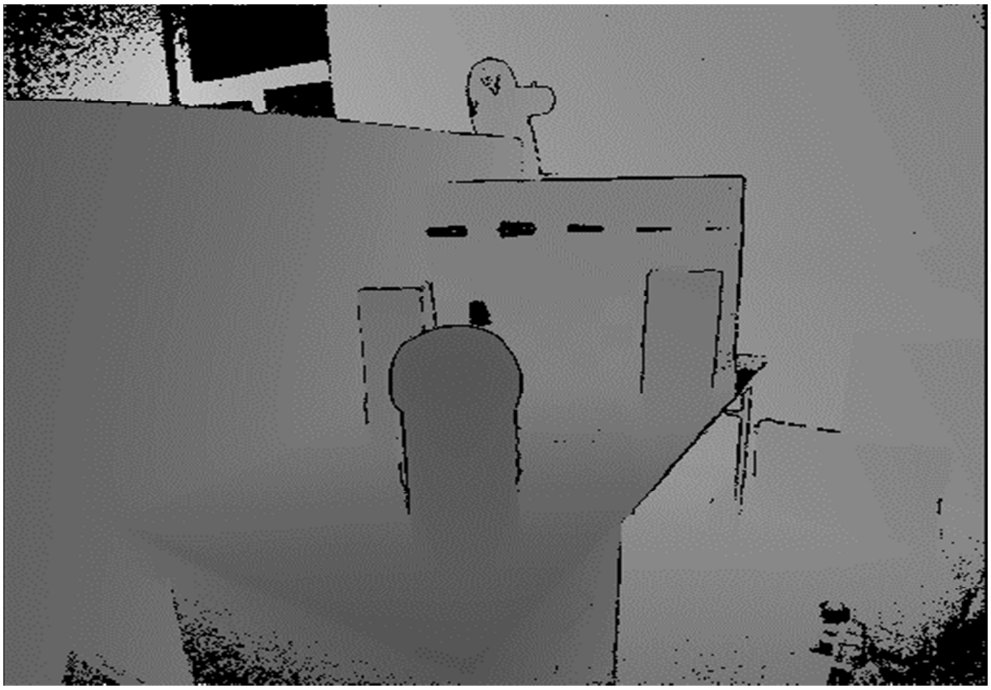


(a)

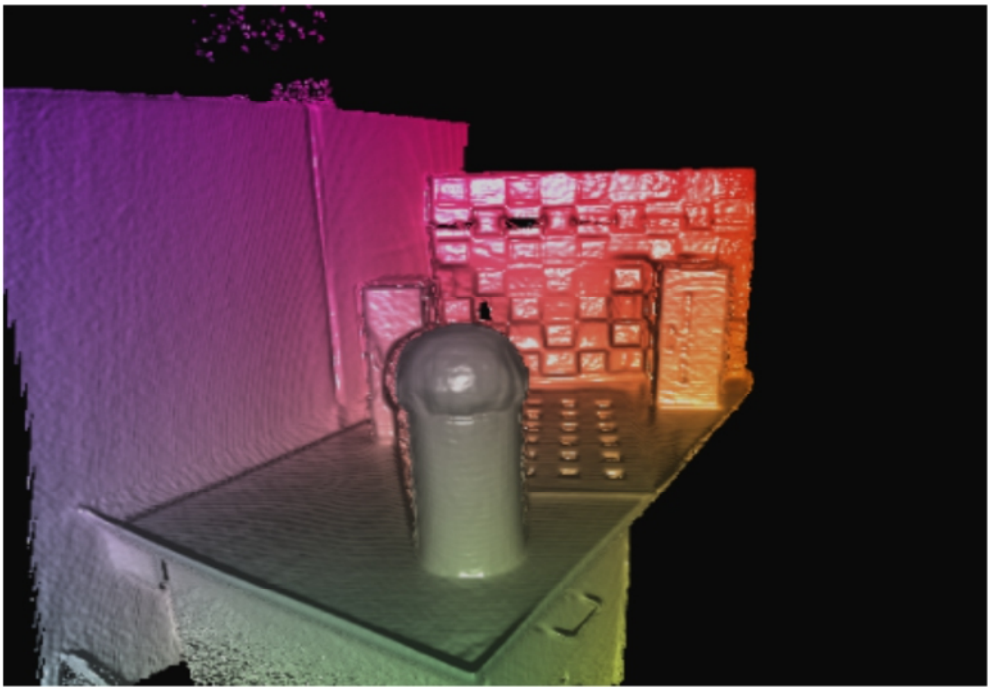


(b)

Figure 2.22 Raw outputs of Kinect V2 cameras. **(a)** RGB. **(b)** IR



(a)



(b)

Figure 2.23 Kinect V2 depth result. **(a)** Intensity representation of the depth map (lighter colour for further objects; black: undefined value). **(b)** Reconstructed surface

2.12.2.2 Kinect V2 Working Principle

Kinect V2 is based on a Time of Flight (TOF) principle to measure the depth of the scene. A TOF camera illuminates the scene with a modulated light signal. The phase shift between the emitted signal and the received one yields the range between the sensor and the scene. Every pixel in the resulting depth map holds a single depth reading.

Structured-light cameras have a higher spatial resolution, but they do not support interference with the alternative light sources. As a result, such a technology is better suited for indoor scenes. On the other hand, TOF technology is less sensitive to the lighting conditions, and it is more affordable. It also delivers a higher frame rate compared to structured light. As a result, the captured geometry is smoother, Figure 2.24.

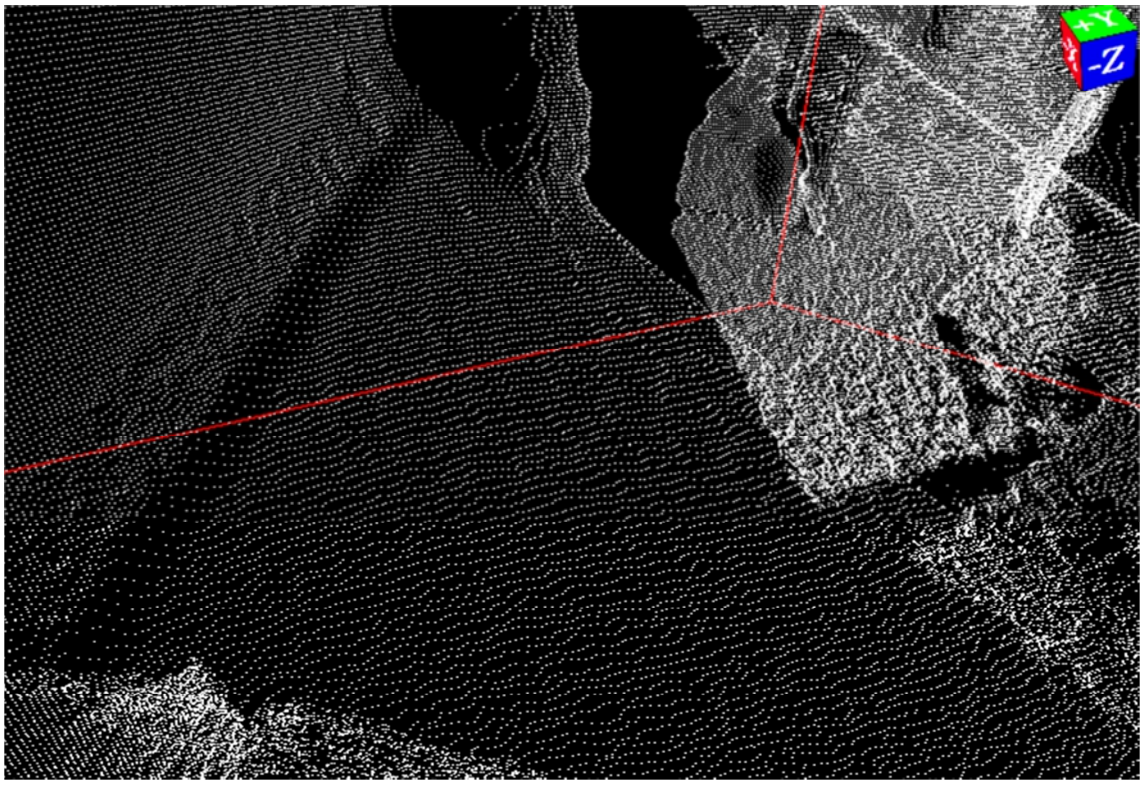


Figure 2.24 Kinect V2 points cloud structure

2.12.3 Kinect Calibration

Kinect sensor is distributed with its factory calibration parameters $f_x, f_y, c_x, c_y, T, R = I$. Hence, these parameters are assumed to be the same for all the cameras in the market. Since the old version of the sensor has a narrow field of view, it does not suffer from significant lens distortion. The newest camera, on the other hand, has a larger field of view. Thus, the distortion cannot be neglected.

A complete calibration is necessary for the camera to acquire reliable measurements [46]. First, both the RGB and IR cameras, are calibrated separately with the checkerboard of Figure 2.25. The purpose is to estimate the parameters regarding the projection matrix K of Equation (2.10). To this end, several images were captured for the calibration pattern at different poses. Indeed, it is recommended to take various samples from the entire operational space for a good result.

For the IR camera, if the projector is left uncovered, the IR speckles yield a salt and pepper noise. This problem can be prevented by covering the projector's aperture with an opaque tape. Table 2.1 summarises the results of the calibration. The overall accuracy of point marking residuals in image space was 0.115 pixels for the IR and 0.078 pixels for the RGB camera.

Parameter	**Kinect V1**		**Kinect V2**	
	RGB	**IR**	**RGB**	**IR**
f_x **(pixels)**	528.22	594.21	1060.70	367.53
f_y **(pixels)**	526.62	591.04	956.35	244.49
c_x **(pixels)**	329.02	339.30	1058.60	366.59
c_y **(pixels)**	268.08	242.73	518.97	207.83
k_1	0.259	-0.263	0.0544	0.010
k_2	-0.840	-0.999	-0.067	-0.289
k_3	0.912	-1.305	-0.019	0.111
p_1	-0.002	-0.001	0.0003	-0.001
p_2	0.001	0.001	-0.002	0.001
T **(cm)**	$\begin{bmatrix} 1.989 \\ 0.001 \\ 0.003 \end{bmatrix}$		$\begin{bmatrix} 5.192 \\ 0.0004 \\ 0.0001 \end{bmatrix}$	
R	$\begin{bmatrix} 0.995 & 0.001 & -0.0174 \\ -0.001 & 0.997 & -0.0122 \\ 0.017 & 0.012 & 0.9934 \end{bmatrix}$		$\begin{bmatrix} 0.9967 & -0.0126 & -0.0072 \\ 0.0162 & 0.999 & 0.0074 \\ 0.0078 & -0.0078 & 0.994 \end{bmatrix}$	

Table 2.1 Calibration results for the two versions of Kinect sensor

Figure 2.25 Calibration checkerboard

2.12.4 Multi Kinects Calibration

The data captured from different Kinects looking at the same scene are mapped to different coordinate frames. It is, therefore, necessary for 3D reconstruction applications to transform the sensor-wise outputs into a standard coordinate system. To this end, a global external calibration of the three cameras in Figure 2.26 was carried out. The calibration procedure that has been implemented in this thesis is a combination of the works [47] [48] using just the colour images as input. This cue is more stable and simpler for the extraction of features. Either the colour or the IR image is sufficient because they both belong to the same sensor that is regarded as a rigid body.

The relationship between the different views is deduced from their feature correspondences. To facilitate the acquisition of feature data, a checkerboard of a known geometry has been used. The whole procedure is described below:

- Several RGB frames are simultaneously taken for the checkerboard by the three Kinects, Figure 2.27 (a), (b), (c). All the cameras must see the same calibration pattern at every frame. This pattern must also be large and positioned at different poses in the working volume.

- The features (corners) of the checkerboard are then extracted, identified and sorted in the conventional order. As a result, a list of correspondences between all the views can be directly obtained.

- Based on these correspondences between each pair of views, the pose relating a couple of sensors is computed. For the setup of Figure 2.26, three different stereo parameters $[\boldsymbol{R}, \boldsymbol{T}]$ were computed, i.e. $(\boldsymbol{k_1}, \boldsymbol{k_2}), (\boldsymbol{k_2}, \boldsymbol{k_3}), (\boldsymbol{k_1}, \boldsymbol{k_3})$.

- Eventually, a loop closure between the elementary stereo results is further applied to reduce the cumulative pairwise error. A bundle adjustment algorithm [44] was run on the stereo and mono camera parameters and the entire image data to refine features' re-projection error [44].

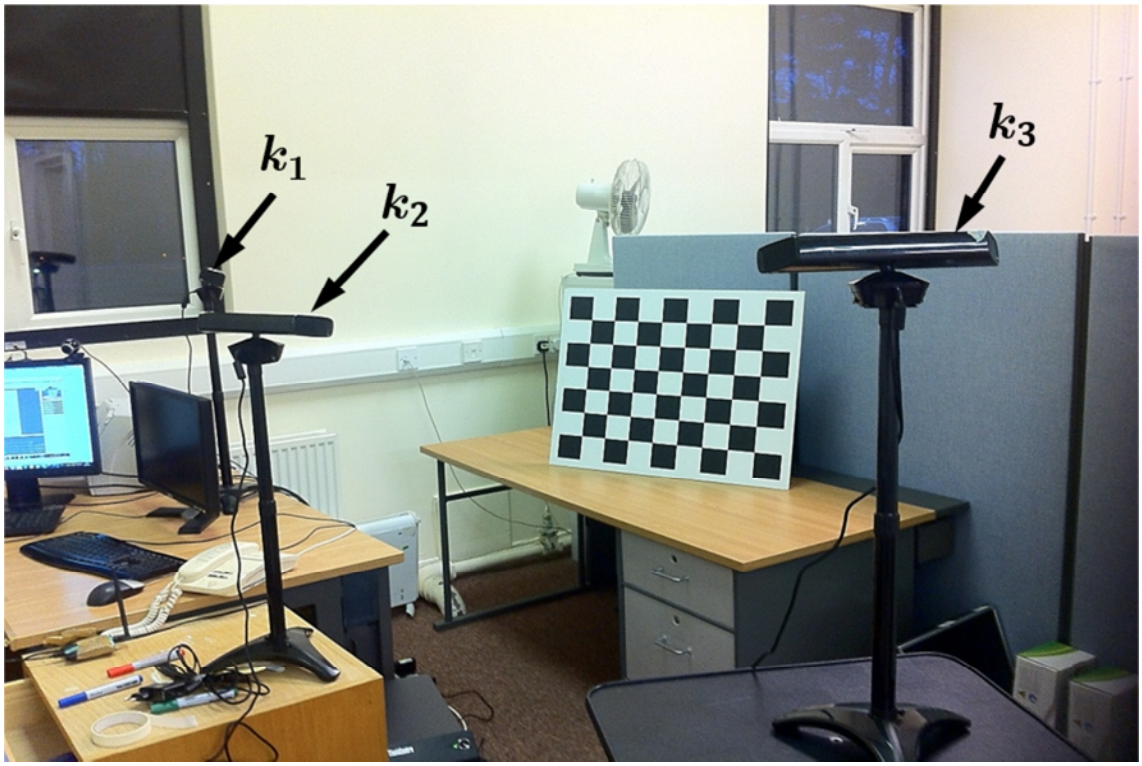
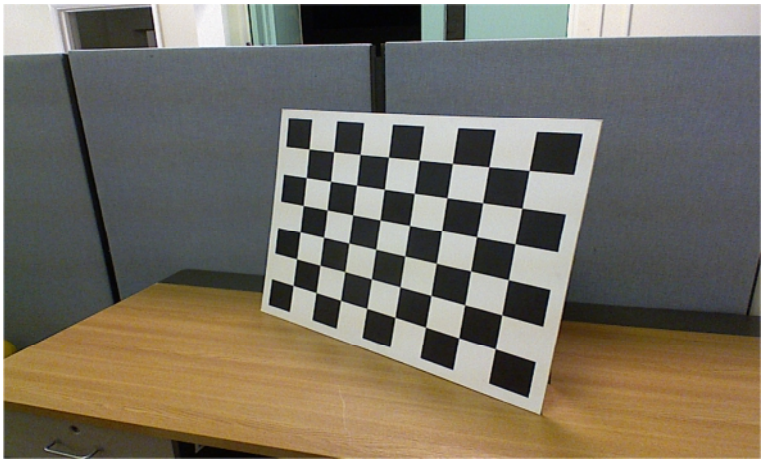
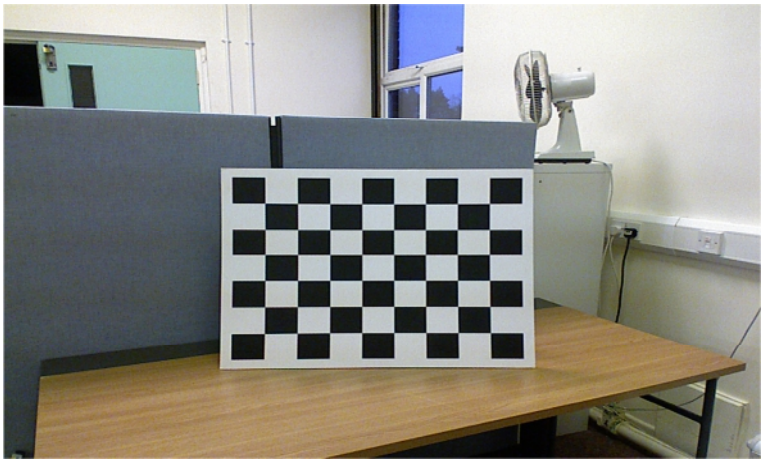


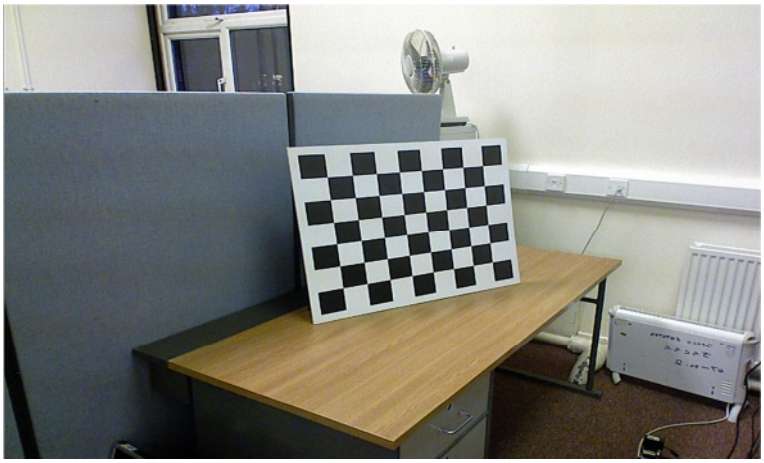
Figure 2.26 Multiple Kinect calibration



(a)



(b)



(c)

Figure 2.27 Multiview Calibration. **(a)** View ***k1***. **(b)** View ***k2***. **(c)** View ***k3***

2.12.5 Interference between Kinect Cameras

When more than one Kinect is looking at the same target, the projected IR beams mutually interfere. This phenomenon manifests in a confusion that misleads the stereo matching, as a single pixel may be attributed multiple depth readings. Consequently, it receives a null disparity. The correspondent of a null disparity is a hole in the captured image, Figure 2.28.

In order to reduce interference when multiple RGBD cameras are used simultaneously, Maimone et al. [49] used a vibration motor. Their idea lies in minimally vibrating a given unit (sensor) to produce an artificial blur to alternative concurrent units. Such a blurring is due to the inability of the latter to observe correctly structured light patterns emitted by the vibrated sensor. As a result, interference among cameras is eliminated. Whereas, the IR projector/camera of the vibrated unit move in harmony. Its depth sensing, therefore, works normally. This solution was not used in our work because the interference was not significant.

2.13 3D Feature Points

Key points are important to find the correspondences between two sets of 3D data. In this thesis, four key point extractors have been tested: TOMASI [50], SIFT3D [51], THRIFT [52] and HARRIS3D [53]. Feature detectors are used to determine the pose between two views. The processing load regarding the registration based on sparse key points is not as substantial as the dense iterative registration's. Among all key points that have been tested so far in the extent of this thesis, HARRIS3D and THRIFT have output the best results for RGBD data. The two methods are based on local property description. In other words, they find in which directions the gradient or the normals change significantly. HARRIS3D uses the gradient of intensity, whereas, THRIFT uses the normals and the geometric curvature.

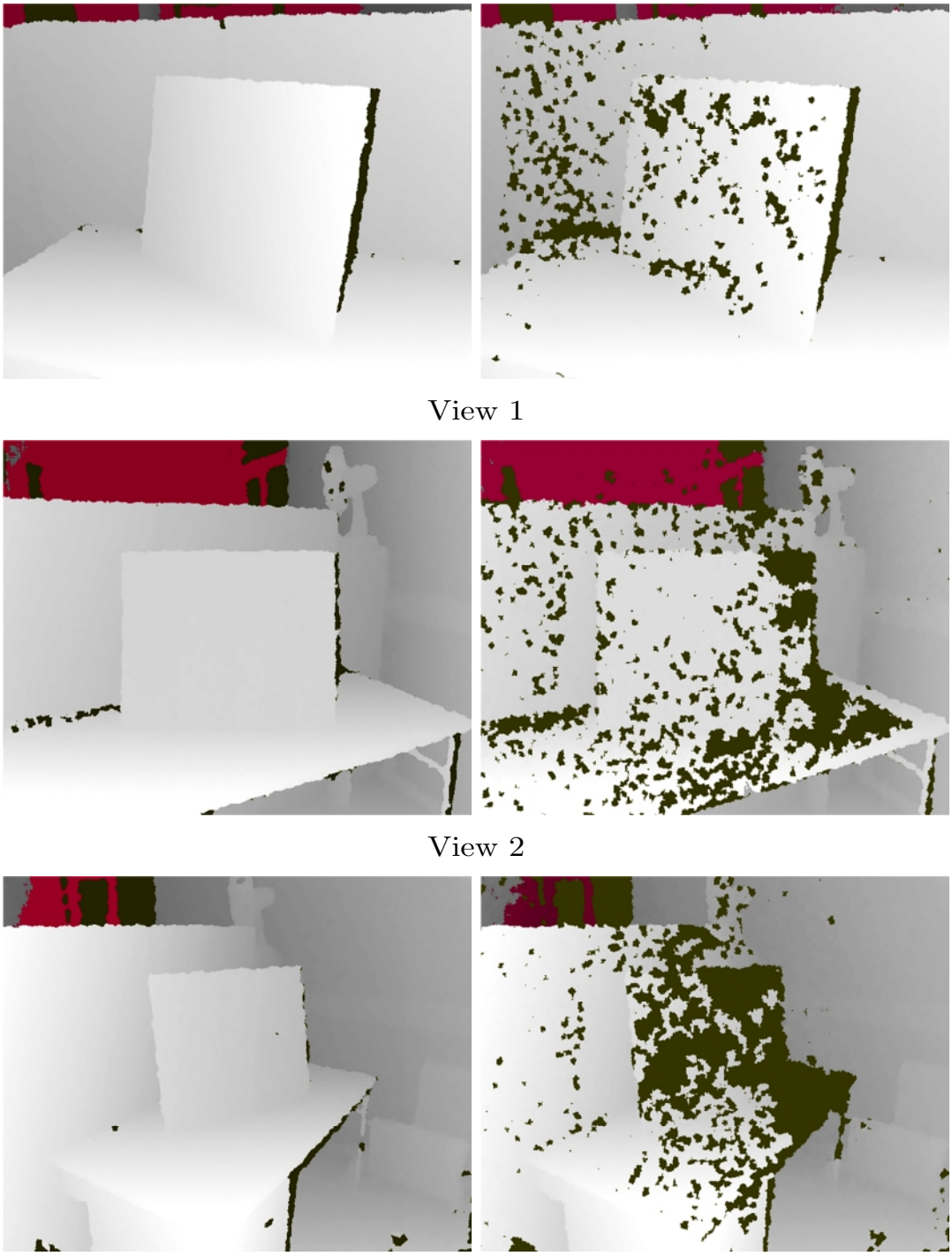


Figure 2.28 Kinect interference problem. On the left, the depth outputs with only one camera operating. On the right, the output when all three cameras are working

2.13.1 3D Key Points' Properties

A good 3D key point must hold all the following properties:

- *Distinctiveness*: The neighbourhood of a key point should have a distinctive geometry and appearance that allow the descriptor to be correctly associated with its counterparts in alternative views.

- *Sparseness*: The number of key points must be small compared to the size of the entire point cloud where they belong.

- *Repeatability*: The key points should also be repeatedly detectable from different viewpoints and in various conditions.

Despite the partial knowledge that the sparse registration methods have on the scene, they are as effective as the dense ones because of the superior distinctiveness of the key points compared to the ordinary points. In addition, the exclusion of the non-key points reduces error during the matching process. This is due to the fact that the presence of non-descriptive points will result in uncertain correspondences. The key points identify a small number of locations where computing feature descriptors have maximum effect.

3D key points should determine stable regions in 3D point clouds. They must remain detectable and less affected by the different transformations that can alter the data they represent [54]. In other words, after applying a rotation, translation or a variation of sampling density, the entries of the descriptors should not change significantly. Similarly, they must also be robust to the mild noise that can corrupt useful data [55].

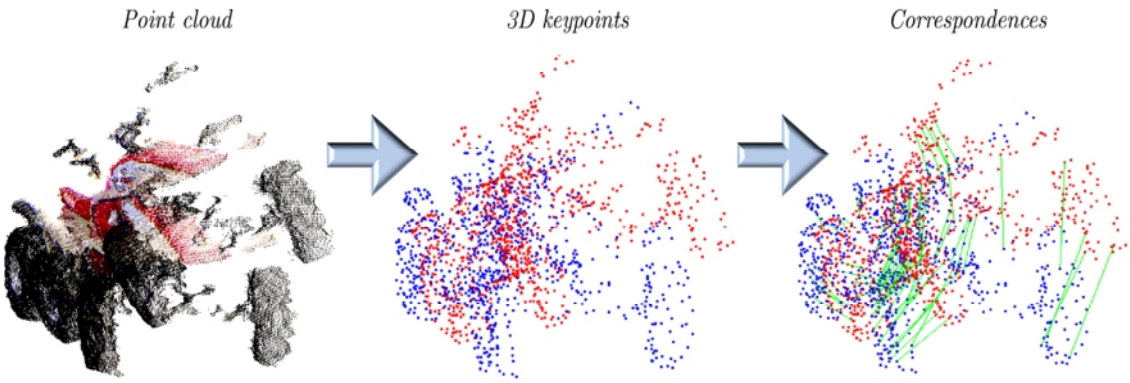


Figure 2.29 Key points extraction and descriptor matching process

Despite the attractive computational asset of geometric key points, their descriptors fail to provide a reliable distinctiveness. This shortcoming is due to the sparseness and the irregularity that characterise point clouds. The addition of colour information, has been proven to be very useful to complete the geometric information. In addition, the colour enables the detectors to leverage the mature solutions already available in the 2D imagery to tackle 3D problems.

2.13.1.1 HARRIS3D Key Points

HARRIS3D feature detector (Figure 2.30 (a)) uses an auto-correlation matrix based on image intensity. This intensity is the value taken by a pixel. It represents a physical entity such as light or the range.

If we consider a patch $\boldsymbol{w}$ in the image $\boldsymbol{I}$ being shifted with a small amount $(\Delta\boldsymbol{x}, \Delta\boldsymbol{y})$. The sum-squared difference at $\boldsymbol{x}$ is:

$$E(x,y) = \sum_{x,y} w(x,y)[I(x,y) - I(x+\Delta x, y+\Delta y)]^2 \tag{2.24}$$

Taylor's first order approximation of $\boldsymbol{I}(\boldsymbol{x}+\Delta\boldsymbol{x}, \boldsymbol{y}+\Delta\boldsymbol{y})$ yields:

$$I(x+\Delta x, y+\Delta y) = I(x,y) + \Delta x\ I_x + \Delta y\ I_y \tag{2.25}$$

$\boldsymbol{I_x}, \boldsymbol{I_y}$ are the gradient in $\boldsymbol{x}, \boldsymbol{y}$ directions respectively.

The matrix form of Equations (2.24) becomes:

$$[\Delta x \quad \Delta y]\, W \underbrace{\begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x\, I_y \\ \sum_{x,y} I_y\, I_x & \sum_{x,y} I_y^2 \end{bmatrix}}_{\boldsymbol{A}} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{2.26}$$

The matrix $\mathbf{A}$ is a cross-correlation matrix. The response $\boldsymbol{R}(\boldsymbol{x})$ of Harris detector is therefore given by:

$$R(x, y) = det\, A - \alpha\, (tr(A))^2 \tag{2.27}$$

$\boldsymbol{\alpha}$ is a positive real number. This parameter plays the role of a lower bound for the ratio between the magnitude of the weakest and the strongest edges. There are two ways to define a HARRIS key point:

- *Significant response*: the locations $(\boldsymbol{x}, \boldsymbol{y})$ with $\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{y})$ greater than a certain threshold.

- *Local maximum*: The locations $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{y})$ is the greatest within the neighbourhood.

In the case of 3D data, for a given point in space, the authors of [53] associate multiple rings composed of neighbouring points centred at the subject key point (centroid). This pattern is then translated to the origin of the 3D coordinate system. The best fitting plane is computed by Principal Component Analysis (PCA). Its normal vector is associated with the lowest eigenvalue.

Afterwards, the set of points is rotated in a way that the normal of the fitting plane is overlaid on the $\boldsymbol{z}$-axis. The least principal component is chosen as a normal to the plane. The authors claimed that the transformed points show a good spread in the $\boldsymbol{xy}$-plane after rotation. Therefore, it becomes possible to work only in the $\boldsymbol{xy}$-plane to compute the derivatives. For the partial derivation in the direction of $\boldsymbol{x}$ and $\boldsymbol{y}$, the authors fit a quadratic surface $\boldsymbol{f}$ to the set of the resulting points. The partial derivatives of this function would replace the gradients of Equation (2.25).

$$z = f\ (x, y) = \frac{P_1}{2}\ x^2 + P_2 xy + \frac{P_3}{2}\ y^2 + P_4 x + P_5 y + P_6 \qquad (2.28)$$

2.13.1.2 HARRIS3D in Practice

The same formulation of the original Harris detector can be followed to extend the definition to 3D space. To this end, the image gradient is replaced by the surface normal. It becomes possible to compute a 3×3 covariance matrix in the neighbourhood of a given 3D point. The response at the 3D point $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, whose normal is $(\boldsymbol{n_x}, \boldsymbol{n_y}, \boldsymbol{n_z})$ can be defined by:

$$R(x, y, z) = det\left(cov(n_x, n_y, n_z)\right) - \alpha(tr(cov\ (n_x, n_y, n_z)))^2 \qquad (2.29)$$

In the Point Cloud Library (PCL) library[11], two implementations of HARRIS3D, dubbed *Lowe* and *Noble,* are available. The difference between them resides in the response function. For Noble, the response is given by:

$$R(x, y, z) = \frac{det(cov(n_x, n_y, n_z))}{tr(cov(n_x, n_y, n_z))} \qquad (2.30)$$

And for Lowe it is given by:

$$R(x, y, z) = \frac{det(cov(n_x, n_y, n_z))}{tr(cov(n_x, n_y, n_z))^2} \qquad (2.31)$$

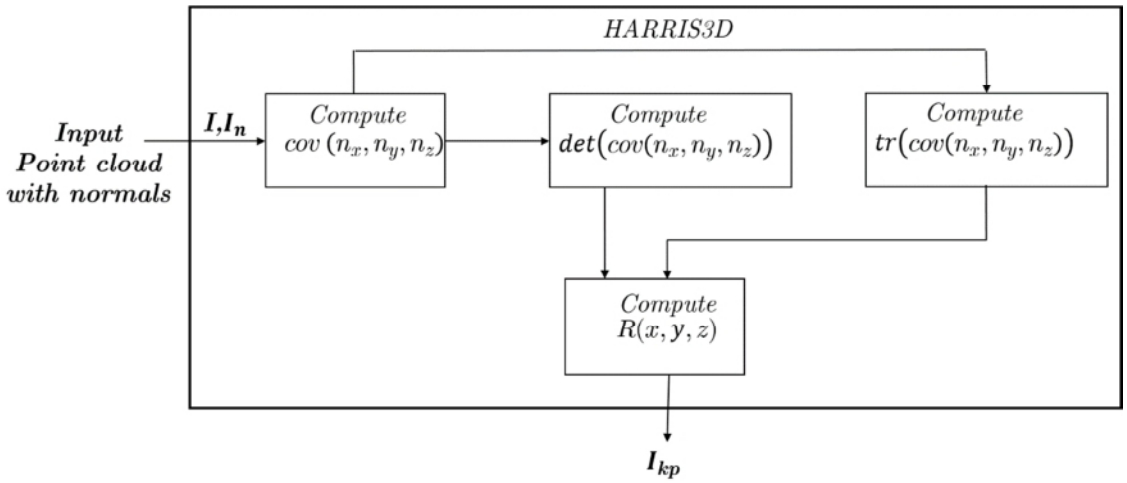
Unlike the previous formulation, these variants deliver a reasonable accuracy, at a small computation burden.

2.13.1.3 THRIFT Key Points

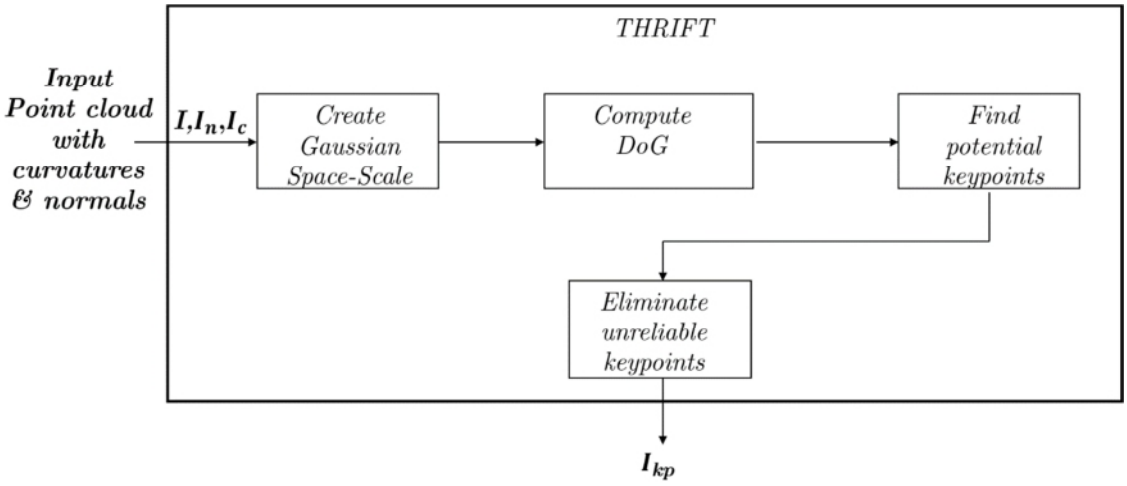
HARRIS3D detector is invariant to rigid body motion. However, it lacks the ability to handle the scale difference between point clouds. In practice, the scale factor tends to vary largely between views. The THRIFT detector (Figure 2.30 (b)) was introduced by Flint et al. [56]. It is an extension to the Scale Invariant Feature Transform (SIFT) [57] and the Speed Up Robust Features SURF [58]. It benefits from the advantages of both for repeated 3D features extraction and

[11] http://pointclouds.org/. 2015

invariance towards scale. SIFT is the original idea that has been proposed in order to cope with the scale change. The same strategy was later extended to deal with the 3D space context [59].



(a)



(b)

Figure 2.30 Key points Extraction Process. **(a)** HARRIS3D. **(b)** THRIFT

Unlike HARRIS3D, the authors replaced the colour intensity or the normals at a given point by the direction of surface normal at its level.

The THRIFT extractor uses the following steps to extract interest points:

- It utilises a 3D version of the *Hessian* matrix to select reliable features. The input is a cloud of points $\boldsymbol{I}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ that will be convolved with the

Gaussian kernels $G(x, y, z, \sigma_i), i = 1, \ldots, n$ of different standard deviations $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$; with $\sigma_{i+1} = \sqrt{2}\,\sigma_i$. Typically $n = 5$, as recommended by the author of the algorithm [57]. These standard deviations act as scaling coefficients.

- Adjacent Gaussians are then subtracted $D(x, y, z, \sigma_j) = G(x, y, z, \sigma_{j+1}) - G(x, y, z, \sigma_j)$. The resulting $D(x, y, z, \sigma_j)$ is the Difference of Gaussians (DOG) between the adjacent clouds.

- The previous two steps are repeated over the entire scale levels.

- The preliminary features are the local extrema of the resulting DOG. Each point in the DOG is compared to its eight neighbours at the same scale and nine in every DOG within the adjacent scales (above and below).

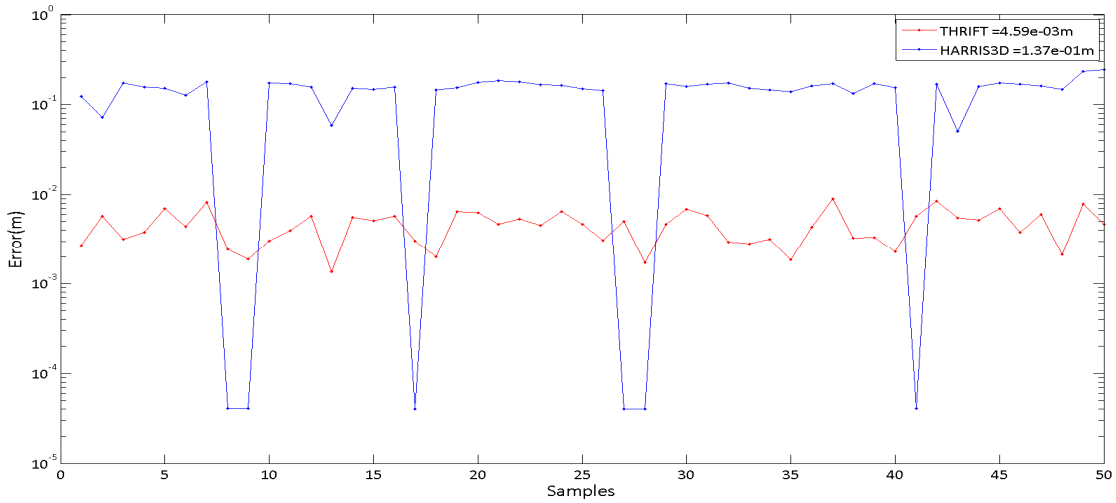
- The point is designated as a *potential feature* when its value is extremal (minimal or maximal). Afterwards, the candidate features are tested for possible elimination.

- A key point will be discarded if its principal curvatures are greater than a specified threshold.

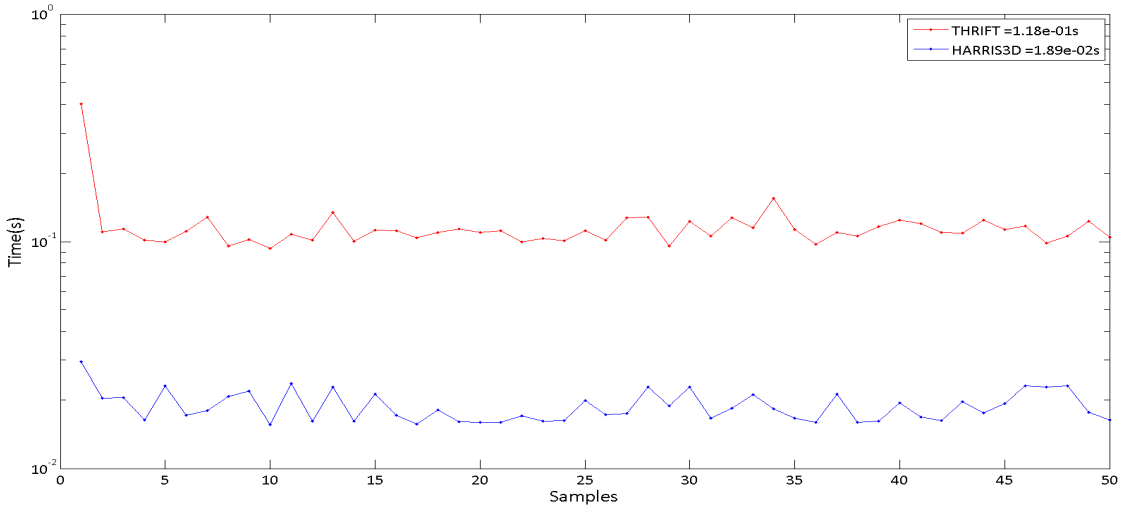
SIFT and SURF both use the image gradient as the basis for describing image patches. Both extractors are robust to changes in viewpoint and partial illumination [53]. On the other hand, THRIFT uses the orientation as a basis for its descriptor. In the case of range data, the dominant orientation at a given point is the direction of surface normal. In this sense, the surface normal is a direct generalisation of the pixel orientation that has been used in SIFT [53].

Figure 2.31 depicts a comparison between THRIFT and HARRIS3D in terms of registration and computation time. The test was run on 50 point clouds of a similar size (50000 points). Typically, the extraction of key points results in an average of 300 features for THRIFT and 176 for HARRIS3D. From the

alignment error, Figure 2.31 (a), as well as the time taken for the extraction, Figure 2.31 (b), it is plausible that THRIFT has a higher performance. On the other hand, HARRIS3D spends less time for an accuracy comparable to THRIFT's.

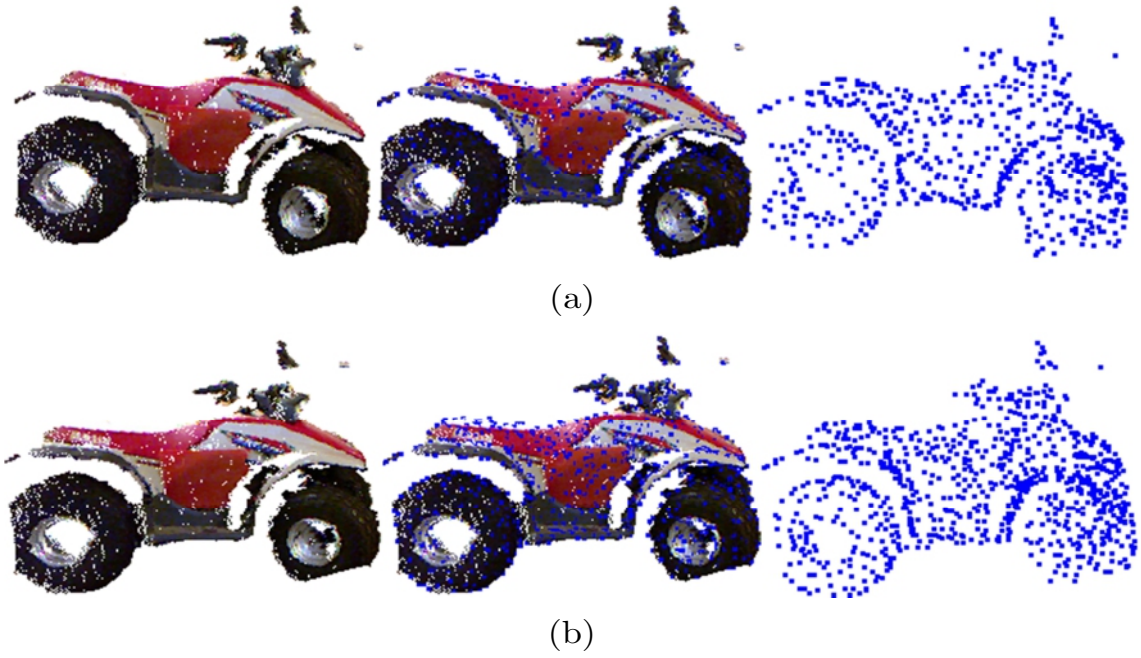


(a)



(b)

Figure 2.31 Comparison between HARRIS3D & THRIFT. **(a)** Error in registration. **(b)** Computation time



(a)

(b)

Figure 2.32 3D key point Extraction. **(a)** HARRIS3D. **(b)** THRIFT

2.13.2 Key Point Descriptors

The quality of feature correspondences has a direct impact on the outcome of the sparse registration. Moreover, the computation of the correspondences itself relies on the quality of the input feature descriptors. Good feature data are, therefore, essential for a correct registration. The latter, should not lead to any confusion during the matching process.

The role of the descriptor is to encode a point's $\boldsymbol{k}$-neighbourhood geometrical and textural properties in a higher dimension space (64, 128, 256, 512 and 1024). The greater the dimension, the more distinctive the descriptors. Nevertheless, the processing of larger data is clearly time consuming.

The similarity between two matched descriptors respective to a pair of key points means that these points represent the same world's feature. A reduced dimension in the descriptor can lead to large errors in the matching process (inconsistent correspondences).

2.13.2.1 CSHOT Descriptor

An evaluation of most of the existing key point descriptors claimed that the definition of a single, unambiguous and stable local reference frame (coordinate system) at each key point is the most prohibiting issue in the description of key points [60]. The authors further proposed a novel descriptor based on the definition of a local Reference Frame (RF). The latter was called Signature of Histograms of Orientations (SHOT). The algorithm of description works as follows, see Figure 2.33:

- It computes a local RF at a given key point $\boldsymbol{p}$. To this end, the coordinates of $\boldsymbol{n}$ of its neighbouring points $\boldsymbol{p_i}; \boldsymbol{1 < i < n}$ are used to calculate a weighted covariance matrix $\boldsymbol{C}$:

$$C = \frac{1}{n}\sum_{1}^{n}(r - \|p_i - p\|).\,(p_i - p)(p_i - p)^T \tag{2.32}$$

  $\boldsymbol{r}$ is the radius of neighbourhood's volume.

- The eigenvalue decomposition of $\boldsymbol{C}$ outputs three orthogonal eigenvectors. The latter are used to define the local RF at the point $\boldsymbol{p}$. The decreasing order of the corresponding eigenvalues associates to each eigenvector the local $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ axes respectively.

- The RF is used to partition the isotropic spherical grid centred at $\boldsymbol{p}$. For every point $\boldsymbol{p_i}$ in a given cell within the grid, the angle $\boldsymbol{\xi_i}$ that separates the normals at $\boldsymbol{p_i}$ and $\boldsymbol{p}$ is computed. The local distribution of angles is therefore described by a single histogram associated with each cell.

- If the grid contains $\boldsymbol{k}$ cells whose histograms comprise $\boldsymbol{b}$ bins, the global histogram related to the feature point $\boldsymbol{p}$ would cover at most $\boldsymbol{kb}$ values. The latter are normalised to sum up to one in order to adapt to the varying density of points across the views.

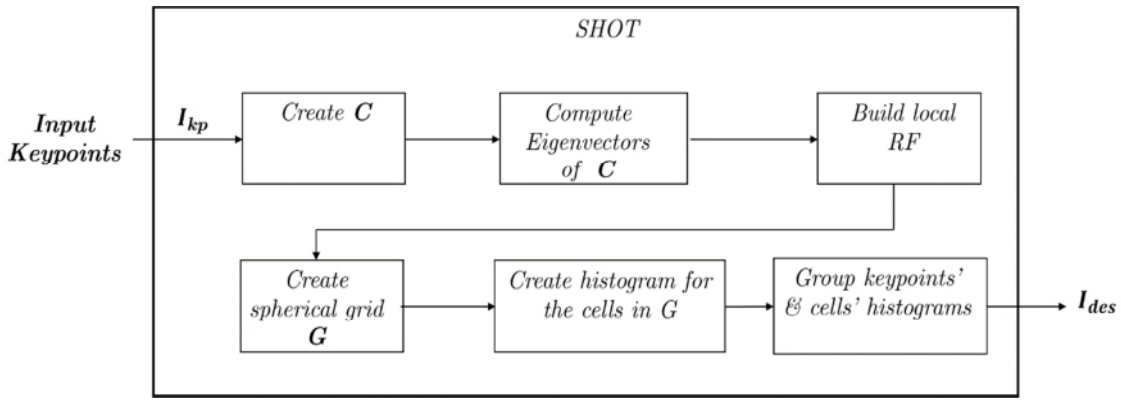


Figure 2.33 SHOT algorithm

SHOT is classified among Signature of Histogram descriptors [61]. It promotes computation efficiency as well as distinctiveness and the robustness against noise. It also allows encoding of multiple cues such as shape and colour simultaneously. Initially, the descriptor was not designed to take advantage of the colour of 3D points. The authors later added this cue to the original solution that has finally become Colour-SHOT or CSHOT [62]. This improved version is used in the thesis to leverage the colour information for a higher performance. The latter is naturally available in the RGBD data. After testing several descriptors [7], CSHOT was proven to be the best qualified for RGBD key points description [7].

2.14 GPU Acceleration

Graphical Processing Units (GPU) are powerful parallel processing devices dedicated to 3D image synthesis. Indeed, gaming and multimedia industries are the principal consumers of this technology. However, the array structure of an image fits very smoothly into the parallel design of the GPUs. This convenience is due to the simultaneous processing that can be launched for every pixel separately. Most algorithms in the literature were developed in a sequential manner that suits CPU. The transition from sequential to parallel processing is not always straightforward since several algorithms proved hard to reshape into a parallel structure [63].

Throughout the past decade, parallel computing platforms and strategies have made significant progress. Such a progress is mostly owed to the escalation of clock frequency and the multiplication of cores within a single unit. Nevertheless, only the multiplication of cores has remained increasing. Such a trend is driven by the cessation of raising clocks frequency because of the heat sink problem [64].

Nowadays, multithreaded programming is primarily related to conventional multicore CPUs [64]. However, the General Purpose GPU (GPGPU) programming model is progressively invading the market for data processing[12]. The demand for a highly realistic rendering has triggered a rapid improvement of GPU technologies. The computational capability of the GPU exceeds considerably that of traditional processors for certain algorithmic structures. As a result, attention has shifted towards them to overcome various computational bottlenecks.

Among the few companies manufacturing GPUs, NVidia™ has played a significant role in affordable accessibility to GPGPU after releasing CUDA™. The latter is a programming language dedicated to writing heterogeneous programmes able to run in both the CPU and the GPU[13].

The general architecture of the GPU is shown in Figure 2.34. This device encompasses $\boldsymbol{n}$ *Multiprocessors* along with their memory spaces. Each of which has a local *Shared Memory* as well as a set of $\boldsymbol{m}$ *Scalar Processors* managed by a local *Instruction Unit*. This unit is responsible for handling the multithreaded execution on the scalar processors. Unlike the CPU, the GPU sustains a higher number of threads executed at lower speed. These threads are grouped into one block called *Thread Block*. Each of which runs entirely on a single *Multiprocessor*. The large number of threads enables a one-to-one association between the processing unit and a datum. Table 2.2 illustrates a comparison

[12] http://www.gpgpu.org. 2015
[13] http://www.nvidia.com/object/cuda_home_new.html. 2015

between the GPU and the CPU for some basic image processing operators. An extensive comparison can be found in [63].

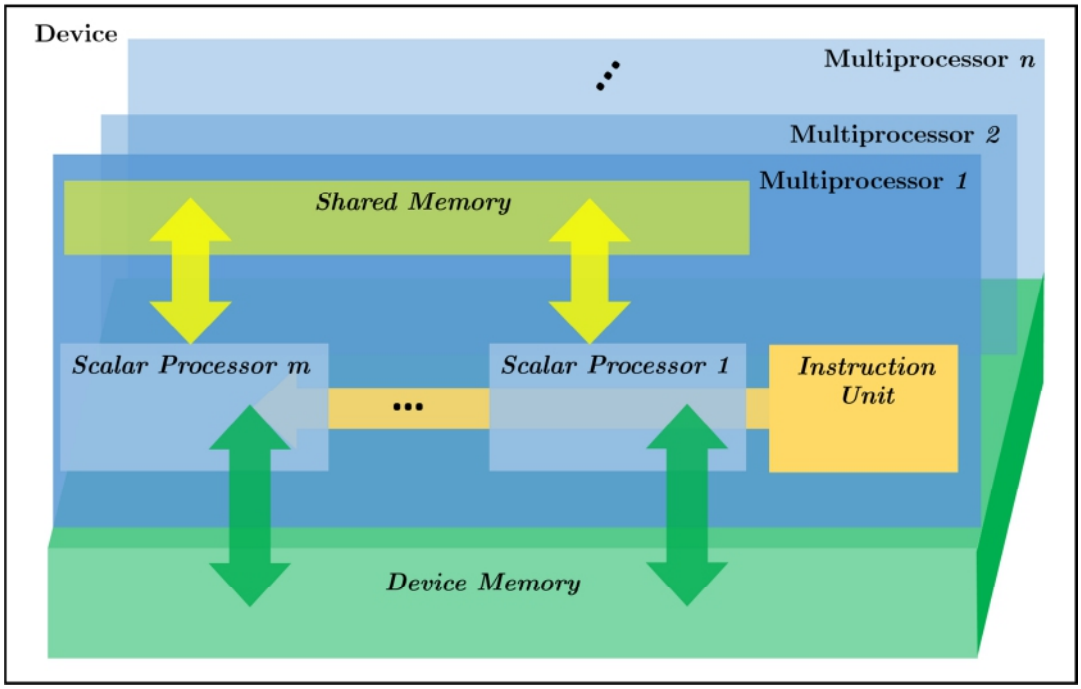


Figure 2.34 GPU (Device) general architecture

Operator	**Erode/Dilate**		**Sobel**	
Image size in pixels	**CPU**	**GPU**	**CPU**	**GPU**
2048×2048	27.35 ms	820 μs	33 ms	962 μs
1024×1024	5.52 ms	153 μs	8.54 ms	234 μs
512×512	1.7 ms	89 μs	1.42 ms	95.4 μs
256×256	750 ms	75 μs	715 μs	74.2 μs
128×128	122μs	53 μs	153 μs	58.49 μs

Table 2.2 CPU/GPU comparison based on some basic image processing operations (binary erosion/dilatation and Sobel edge detector). The hardware includes an Intel i7 CPU, 3.20 GHz. NVidea GeForce 2GB GTX 680

2.15 Conclusion

In this chapter, the seminal theoretical and practical basis of the concepts related to the thesis was presented. The notions of VR and AR, which can be merged to form MR, were studied. Several examples available in the market were given to motivate the practical utilisation of these technologies (AR, VR and MR).

The communication between the real and the virtual requires an accurate registration and tracking of the physical entities to mimic their behaviours in the virtual environment. Both concepts, i.e. registration and tracking, were thoroughly presented.

The alignment of 3D data requires the rigid body motion that relates two point clouds. The latter can be computed either with a *sparse* or a *dense* strategy. The former selects just some reliable key points; whereas, the latter considers all points together.

On the other hand, the estimation of the motion undertaken by real objects is necessary to update the state of their virtual counterparts. This task can also be approached in two different alternatives: The first one utilises several artificial markers, OptiTrack system was presented as a case study. The second is markerless, it relies only on the natural landmarks that can be obtained from the scene. The advantages and drawbacks of each were discussed.

Since the data that feeds the MR applications is mainly visual, another section was dedicated to defining the different schemes related to the camera model. In other words, the *Pinhole*, *Stereo* and *Multiview* imaging principles were all debated along with their properties and mathematical formulations. A case study about RGBD sensors used to capture datasets for the validation of contributions was considered with the necessary calibration procedures.

The 3D interest points used by the registration were also discussed. If the scale different is not significant, HARRIS3D is better. Otherwise, THRIFT can cope

with the scale for an extra load in computation. Furthermore, the CSHOT descriptor was presented along with its algorithm.

Lastly, the GPU architecture was described. This device is used in the thesis to ensure real-time responsiveness of otherwise bottlenecked algorithms.

3 GPU-Based Real-Time RGBD Data Filtering

Commodity RGBD cameras such as the Kinect sensor have recently been a large success in many indoor robotics and computer vision applications. Nevertheless, professional applications cannot rely on their raw outputs because of their low accuracy. Indeed, these consumer cameras can produce precise depth measurements within a small range, but, they do suffer from potential noise when the target is further away than the permitted distance. In this chapter, an innovative adaptation of the Kalman filtering scheme is proposed to improve the precision of Kinect as a real-time RGBD capture device. The Kalman filter's adaptation to any Kinect-like camera is demonstrated and justified by real experiments. A GPU implementation of the filter with different coding optimisations is also described.

3.1 Overview

The notion of mixed reality has been progressively gaining importance in many civilian and defence applications. This trend is mainly due to the easiness of importing 3D data from the scene into virtual environments [1]. Tools to acquire the 3D information of objects are required, however. Laser scanning devices can produce very high-quality scans [65], but they are expensive and require a level of proficiency to manipulate them correctly [2]. Along with laser scanners, ultrasonic and radar scanners [66] are available for use as well. The existing multi-view methods [5] can produce acceptable models after the registration of two or more views acquired for the same object at different viewpoints [4]. Nevertheless, this solution for 3D reconstruction is computationally greedy; therefore, not suitable for real-time applications [6]. Besides, the multiview reconstruction technology presents other drawbacks such as sparse textures or complex occlusions among different perspectives [4].

Two other classes of range sensors namely, Time of Flight (TOF) and structured light can be considered as well [67]. The former captures reflected light and computes the distance between the sensor and the scene from the time elapsed between emission and reception [68]. The latter uses an IR projector that fires light patterns onto the scene [69]. The same patterns are then captured back by an IR camera then the sensor produces a disparity map in order to determine the actual depth of the scene [70]. The purpose of this chapter is commodity range cameras capabilities enhancement with the integration of an innovative filtering stage in order to infer accurate and trustworthy 3D scans. Nevertheless, the precision on its own is not sufficient for a realistic 3D emersion. Real-time performance should also be considered in the architecture that is aimed to be designed. In other words, filtering algorithms should run at the same frame rate as image acquisition without introducing any latency in the system. Consequently, the user benefits as much as possible from the stream of data delivered by the sensor.

The device used is the Microsoft Kinect[1] V1. Additionally, a parallel design is proposed and implemented in the GPU for a shorter response time. At this level, it would become possible to embed the entire GPU-based filtering algorithm in dedicated cards for a self-contained capture/filtering solution. The resulting configuration could serve as a pre-processing layer for any pipeline using RGBD data to feed the subsequent processing levels with clean inputs.

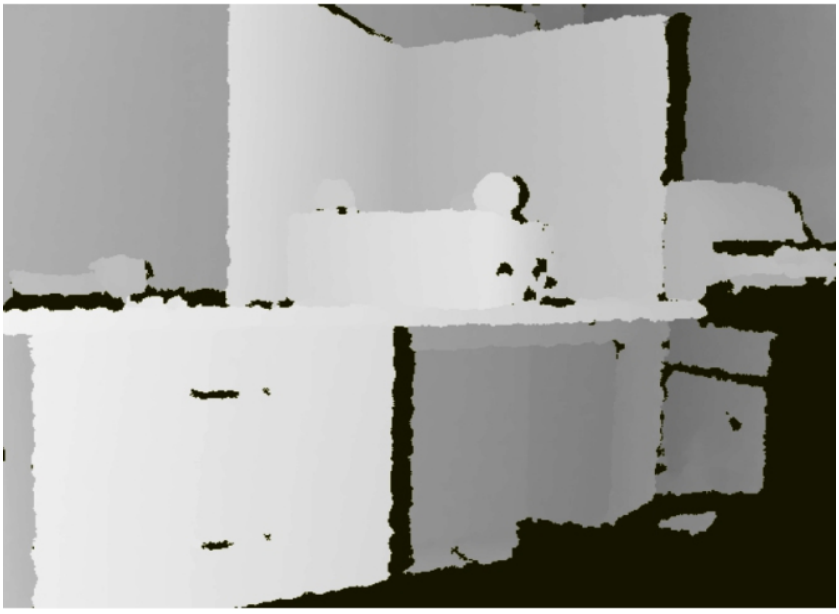
This chapter is organised as follows: In Section 3.2, the related state of the art research in RGBD data filtering is debated. The general architecture of the system is then presented in Section 3.3. In Section 3.4, Kalman filter is presented. Then, its utilisation to improve the depth quantisation accuracy is investigated in Section 3.5. In addition, the different components of the sensor, its driver, and the higher data smoothing algorithms are discussed in detail. In the same section, the issue of accuracy is raised. In addition, the Mathematical formulation of the problem and the properties of the camera are exposed in detail. The effect of the proposed filtering scheme is validated with two experiments:

The former (Section 3.6) is a moving vehicle tracking scenario where it has been confirmed that cheap RGBD sensors can be used as an accurate real-time tracking device. The results are obtained from a single depth camera for moving robot localisation. The purpose of this type of applications is the evaluation of filter's precision.

The latter (Section 3.7) demonstrates the performance of a Kalman filter in a depth image registration pipeline. Here, the visual quality of the geometry built upon filtered 3D data is assessed. 3D registration plays a fundamental role in many applications, such as simultaneous localisation and mapping (SLAM). The potential of the proposed filtering scheme is easily noticeable after testing it against equivalent methods in the literature in Sections 3.8, 3.9.

[1] http://www.microsoft.com/en-us/kinectforwindows/. 2015

(a)



(b)

Figure 3.1 Kinect depth and colour data. **(a)** RGB. **(b)** Depth

3.2 Related Works

By their very nature, depth data are rough and noisy. RGBD sensors, in general, are sensitive to noise because of their active nature, see Figure 3.1. Filtering approaches aim to remove the noisy data (outliers), clean the useful regions (inliers) and preserve the edges. In the presence of specular reflective or light emitting objects, holes appear in the captured depth data. Hole-filling can be a useful tool to recover the lost data, but it is a challenging task due to the missing depth values. The recovery of the lost information is constrained by some assumptions about the neighbourhood where disparity data is available.

Despite the non-negligible quantisation noise (Chapter 4), most research papers commonly use raw Kinect data without any pre-processing or filtering. Hence, arises the motivation to address the limitation of the covered space and the maximum reachable distance. Some alternative works in the literature have already alluded to Kinect data de-noising or proposed a pre-processing stage in their applications. Menna et al. [71] presented a detailed study regarding precision of the Kinect's depth map. Although no particular approach to depth map accuracy improvement is proposed, they applied a filtering approach based on the Spatio-temporal median computed from motion vector. On the other hand, Camplani et al. [72] used an adaptive joint bilateral filter that combines depth and colour images by analysing an edge-uncertainty map and foreground regions to improve the quality of Kinect data.

Kalman filter is an optimal state estimation tool that can produce statistically optimal estimates from a sequence of noisy measurements observed over time [73]. This filter is well-known amongst navigation, guidance, communication and control researchers. It is appreciated by the community because it helps enormously in predicting and correcting the context of noisy measurements.

The Kalman de-noising algorithm was implemented to clean Kinect data in a few works in the literature. Ling et al. [74] applied the extended Kalman filter in a real-time 3D mapping framework on Kinect RGBD data. The authors

proposed a repetitive linearization of the nonlinear measurement model to provide a running estimate of camera motion. Likewise, Thibault et al. [75] applied nonlinear-Kalman filtering to generate accurate 3D maps. Sangheon et al. [76] also proposed a 3D hand tracking method based on the Kinect along with a Kalman filtering strategy.

In all the previously cited works, the Kalman filter was customised to fit the target application (3D mapping or tracking). However, the novelty of this contribution originates from the fact that some interesting characteristic properties of the Kinect sensor and the behaviour of its outputs over time were uncovered. The depth measurements can be optimally filtered to feed several applications without any supplementary parameter tuning. Such a modelling is useful for the users of the Kinect camera in particular, and RGBD sensors in general.

3.3 System Architecture

As illustrated in Figure 3.2, the Kinect outputs three different streams of data[2]. Among all others, the depth stream is of particular interest in the system to be designed. The Kinect sensor has the advantage of working in real-time at a frequency of 30 FPS. Whenever a further processing load is included in the line of processing, a significant frame rate drop can occur. Consequently, to conserve the real-time nature of the solution, an optimal hardware/software combination that best fits the requirements has to be found.

The GPU has provided many advantages when the CPU has been proven incapable of coping with substantial data. As a consequence, a series of algorithms embedded in the graphics processor has been designed. These algorithms allow full advantage of the maximum frame rate delivered by the camera.

[2] http://www.microsoft.com/en-us/kinectforwindows/develop/learn.aspx. 2015

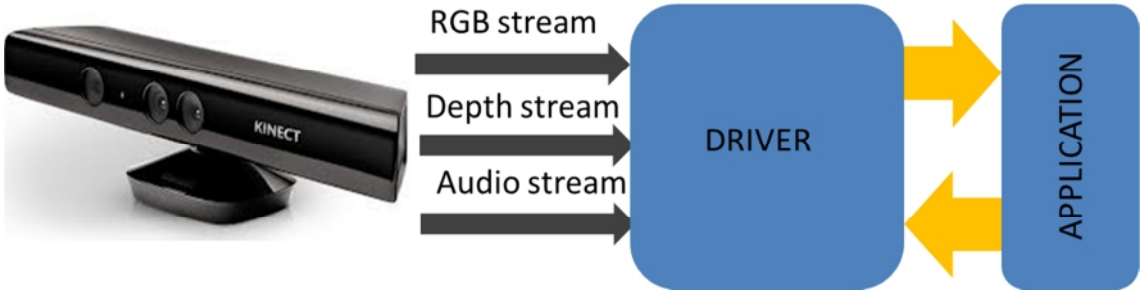


Figure 3.2 Kinect data streams

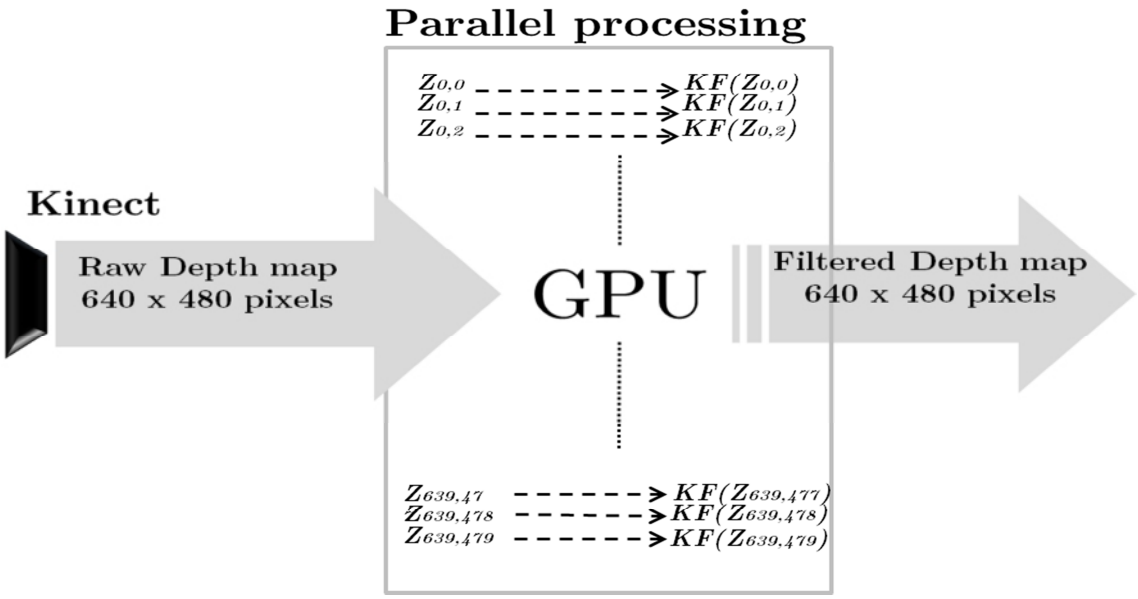


Figure 3.3 Real-time RGBD data filtering architecture

Such a performance would be otherwise impossible to reach with the classic CPU implementations. Figure 3.3 shows the stream regarding depth data flowing from the camera to the parallel filtering stage, to arrive finally at the application level. This architecture could be easily integrated as an independent data enhancement module into the driver of any RGBD camera. The core algorithm requires only some initialisation with the appropriate calibration parameters.

From the software side, a recursive state-transition filter for Kinect's depth data was adapted. However, this was not a straightforward task, since some preliminary conditions needed to be fulfilled. In addition, this kind of adaptation has not been previously discussed in the literature. Thus, a proper mathematical formulation was developed to fit the problem adequately into the filter's model.

3.4 Kalman Filter

Since the Kinect was designed for computer gaming, where the user is relatively close to the sensor all the time (Chapter 2), professional applications using Kinect's data can only be accurate within a small range. Indeed, for depth values greater than 3.50m the error can reach $\pm$20cm. This low accuracy is unacceptable by most market and research applications.

Many filters in the literature can be applied to improve the limited quality of data. Nevertheless, none was entirely adapted to work on RGBD sensors. Here, a Kalman filter was used carefully to stabilise the capture of Kinect depth data over time. Sensory capabilities of covering a larger and deeper view were also enhanced. As stated earlier, this contribution is motivated by some properties discovered in the depth data. These features allow fitting of the depth data smoothing problem into a Kalman filter stabilisation framework. The working principle of this filter is based on a recursive *prediction* of the next state followed by its optimal *correction*. The Kalman filter is distinguished by its ability to run in real-time, using only the recent measurements as input and the previously estimated state. Thus, no additional anterior knowledge about the behaviour of the system is required [77]. In addition, the filter needs the statistical characteristics of the inherent process and measurement noise models. To ensure the optimality of estimation, some conditions require to be initially satisfied, however. The mathematical rationale is based on the assumption of Gaussianity resulting from system and measurements' noise processes; as well as the linearity of the frame to frame relationship between successive states. As will be shown, both conditions are verified for RGBD sensors.

The general form of state-transition filters to predict or estimate the state of a dynamic system from a series of incomplete or noisy measurements is defined by the following equations:

Prediction

$$X_t = A_t \widehat{X}_{t-1} + B_t U_t + w_t \tag{3.1}$$

$$Y_t = z_t \tag{3.2}$$

$$P_t = A_t \hat{P}_{t-1} A_t^T + Q_t \tag{3.3}$$

Correction

$$K_t = P_t H_t^T (H_t P_t H_t^T + R_t)^{-1} \tag{3.4}$$

$$\widehat{X}_t = X_t + K_t (Y_t - H_t X_t) \tag{3.5}$$

$$\hat{P}_t = (I - K_t H_t) P_t \tag{3.6}$$

Where for each discrete time-step $\boldsymbol{t}$:

$\boldsymbol{X_t}$ is the a priori state estimate; $\widehat{\boldsymbol{X}}_{\boldsymbol{t}}$ is the a posteriori state estimate; $\boldsymbol{Y_t} = \boldsymbol{z_t}$ is the measurement; $\boldsymbol{P_t}$ is the a priori state-error covariance estimation; $\hat{\boldsymbol{P}}_{\boldsymbol{t}}$ is the a posteriori state-error covariance estimation; $\boldsymbol{K_t}$ is the Kalman filter's gain; $\boldsymbol{A_t}$ is the state-transition model; $\boldsymbol{B_t}$ is the control-input model; $\boldsymbol{H_t}$ is the observation model; $\boldsymbol{Q_t}$ is the covariance matrix of process noise; $\boldsymbol{R_t}$ is the covariance of measurement noise. $\boldsymbol{w_t}$ is a random variable representing process noise, $\boldsymbol{v_t}$ is another random variable representing measurement noise; $\boldsymbol{u_t}$ is a control signal. Process and observation noise models should be independent,

white and follow a normal distribution $\boldsymbol{w_t} \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \boldsymbol{Q_t})$, $\boldsymbol{v_t} \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \boldsymbol{R_t})$, respectively.

To adapt the Kalman filter to this particular RGBD data filtering problem, one should first demonstrate that the sensor's model and its data satisfy the requirement to fit into Equations (3.1) through to (3.6). During the experiments, it was found that Kinect RGBD pixels lie in parallel planes towards the positive z-axis direction. The depth values are limited to a known discrete range. An adaptation of depth data structure to the Kalman filter was contributed in order to improve the precision of the sensor without any extra hardware. The latter associates each pixel with an optimal depth value within a few frames.

3.5 Kalman Filter on Kinect's Data

3.5.1 Z-Resolution

To study the nature of depth resolution regarding the camera, we pointed the sensor parallel to a large flat wall as shown in Figure 3.4 (a). This setup allows us to get a cloud of points from the whole operational field of view and to determine the parameters regarding the filter.

As shown in Figure 3.4 (b), depth resolution is inversely proportional to the distance from the sensor. In addition, the points within the snapshot (taken from the same frame) are distributed over independent clusters that have been defined as Z-levels, Figure 3.4 (c). For this reason, the Kinect's data is formulated as a finite set of points lying in parallel planes. Every plane constitutes a partition of the whole cloud of points. The mathematical definition is as follows:

- $\boldsymbol{K}$: Set of ordered indices ranking the parallel planes.
- $\boldsymbol{I}$: Set of indices indexing the points resting in the same plane.

- $\boldsymbol{C}$: Set corresponding to the whole point cloud Figure 3.4 (b),
- $\boldsymbol{Z_k}$, $\boldsymbol{k} \in \boldsymbol{K}$: Plane in $\boldsymbol{C}$ Figure 3.4 (c),
- $\boldsymbol{P_i}(\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{z_i}), \boldsymbol{i} \in \boldsymbol{I}$: Point in the RGBD space, lying in a given Z-Level; $\boldsymbol{Z_k} = \boldsymbol{z_i}$, $\boldsymbol{k} \in \boldsymbol{K}$, Figure 3.4 (c).

Every point cloud $\boldsymbol{C}$ satisfies the properties:

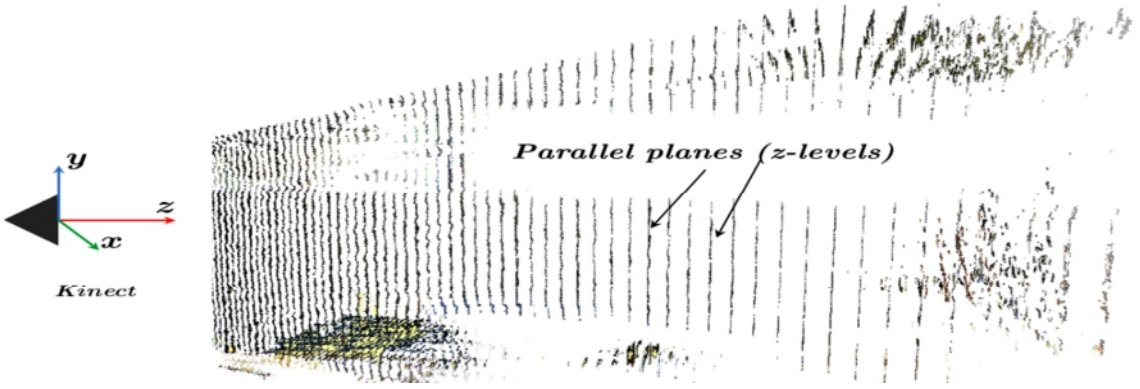
$$C = \bigcup_k Z_k, k \in K \tag{3.7}$$

$$\forall\ Z_{k1}, Z_{k2} \in C, k_1, k_2 \in K, k_1 \neq k_2, Z_{k1} \cap Z_{k2} = \emptyset \tag{3.8}$$

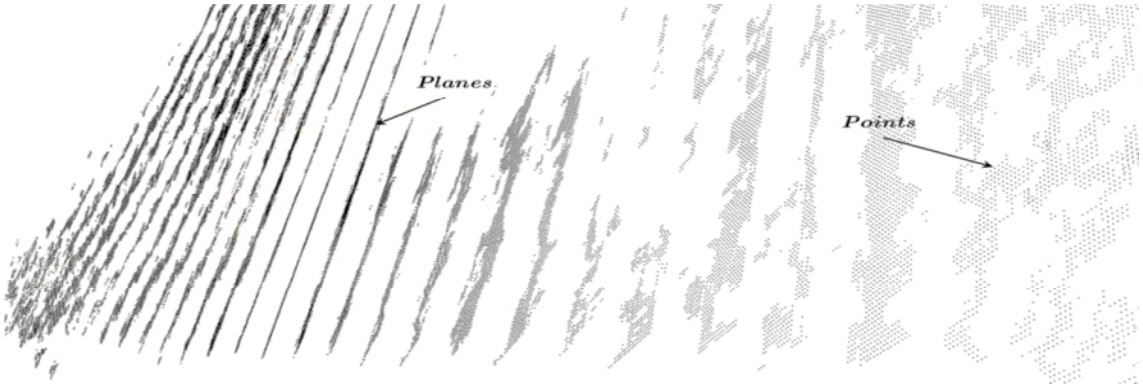
$$\forall\ p_i \in C, i \in I, k \in K, \exists!\ Z_k, p_i \in Z_k \tag{3.9}$$

$$\forall\ Z_k \in C, k \in K, Z_k \perp Zaxis \tag{3.10}$$

(a)

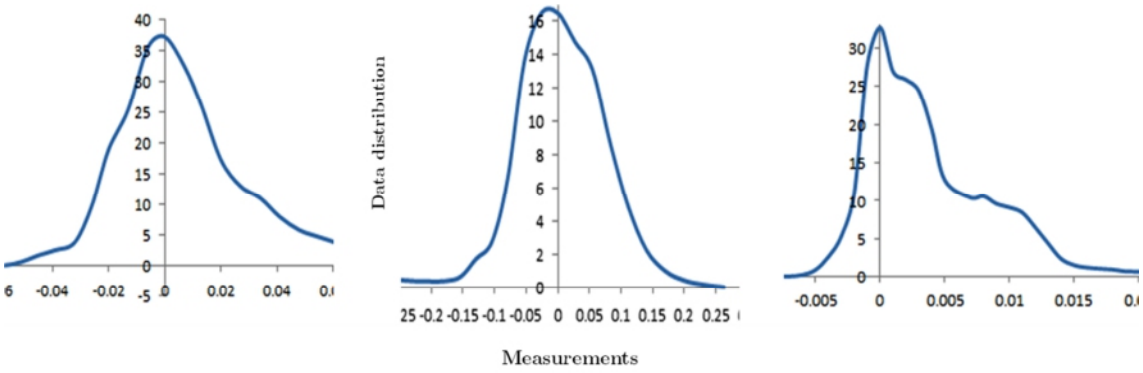


(b)

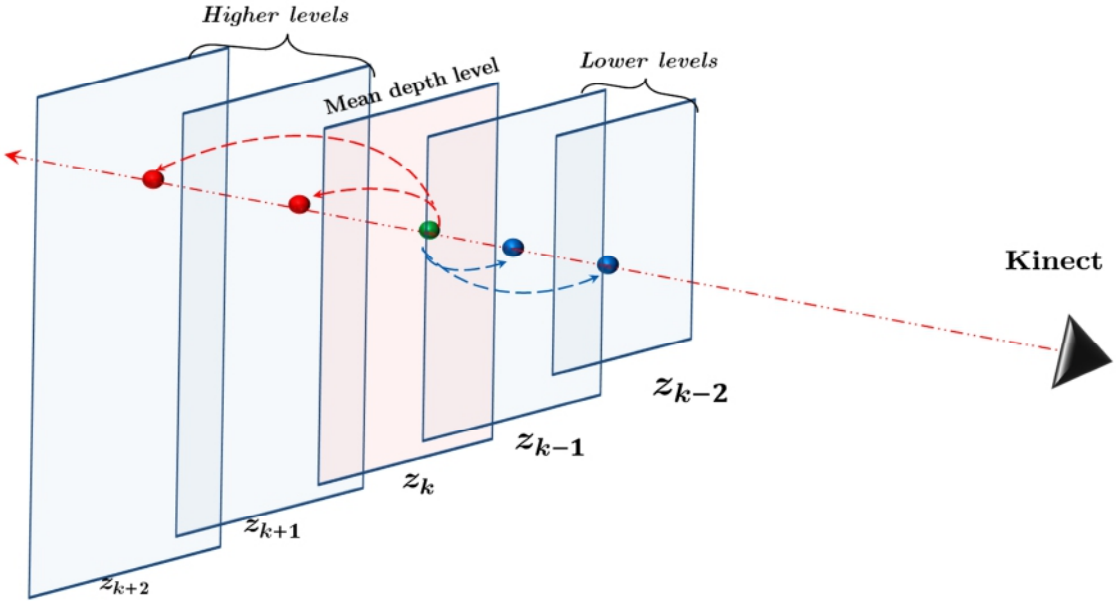


(c)

Figure 3.4 Kinect's point cloud structure. **(a)** Colour image. **(b)** Depth image. **(c)** Point cloud components



(a)



(b)

Figure 3.5 Depth data quantisation noise. **(a)** Statistics of depth data at 1.51m, 3.40m and 0.60m, respectively. **(b)** Displacement of a single depth measurement

3.5.2 Depth Noise Statistics

The Kinect, as an electronic device, has an inherent hardware related noise. The latter is due to reference template inaccuracy, the calibration process or lighting conditions and the objects' surface properties [46]. The errors in the imaged data increase proportionally to the depth of the scene. This behaviour is due to the decrease in depth resolution, see Figure 3.4 (b). We carried out a study in order to determine the nature of noise affecting depth measurements. We found, after sampling and fitting the distribution of data with different probability models of different parameters, that depth outputs are more likely to follow a normal distribution. This distribution has the average of Z-levels as mean and $\boldsymbol{\sigma_k}$, of Equation (3.11), as standard deviation. More importantly, the more samples we consider, the more Gaussian depth data distribution becomes.

Figure 3.5 (a) shows some samples that were captured at 1.51m, 3.40m, and 0.60m, respectively. Based on the graphs, the corresponding standard deviations are respectively 0.032m, 0.075m and 0.0025m.

When the sampled points were re-projected back to their original depth map, it was found that the standard deviations $\boldsymbol{\sigma_k}$ could be formulated by this equation:

$$\sigma_k = (\ Z_{k+h} - \ \ Z_{k-h})\ /\ 2, \forall k\ \in\ K, h \in \mathbb{N}^+ \qquad (3.11)$$

Where $\boldsymbol{\sigma_k}$ is the average distance between the two extremities of the $\boldsymbol{2h+1}$ Z-levels and the central one $\boldsymbol{z_k}$ that contains the sampled point, see Figure 3.5 (b). As a result, at every level $\boldsymbol{z_k}$, Kinect noise remains Gaussian and $\boldsymbol{\sigma_k}$ defined in Equation (3.11) is its standard deviation. Empirically, the best results are reached when $\boldsymbol{h = 3}$. This property allows one to prove the Gaussian nature of the quantisation noise affecting the depth data. It also means that the first condition required to apply a Kalman filter is satisfied. For instance, a Kalman filter still works fine for non-Gaussian noise; but estimation optimality is not guaranteed [77].

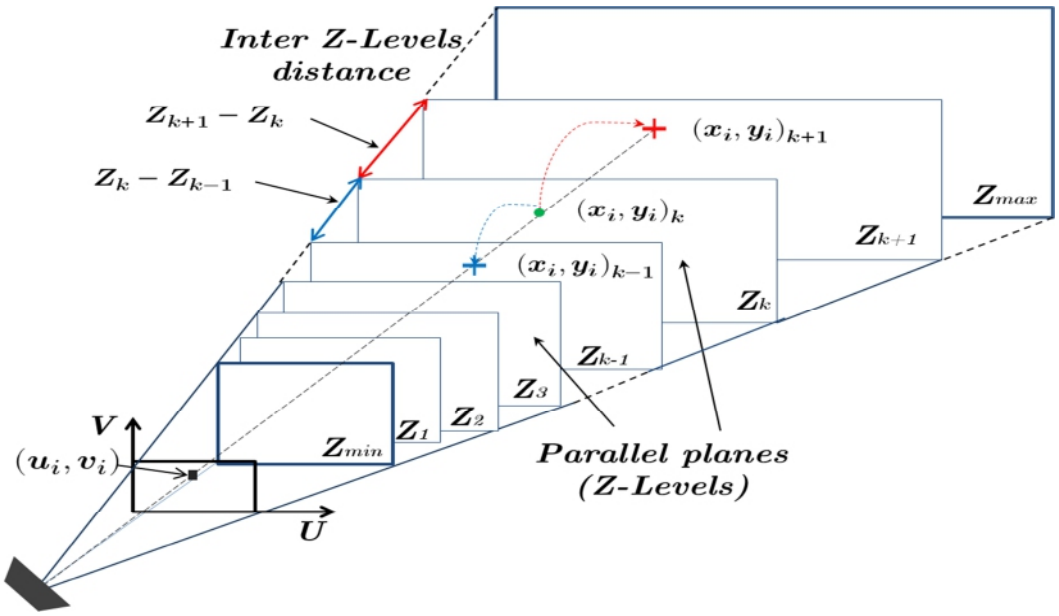


Figure 3.6 The behaviour of $\boldsymbol{\sigma_k}$ for every Z-Level

3.5.3 IR Pixel States

When the sensor is pointed towards a static scene, and the depth map is observed over time (Figure 3.6), fluctuations in almost 90% of all map's elements (range readings) are noticed. The sensor and the scene are assumed to remain steady during the whole period of acquisition.

The depth value taken by a given pixel $(\boldsymbol{u_i}, \boldsymbol{v_i})$ in the depth image tends to vary within a limited range over time. This variation is due to the fact that: for every frame, the discrete imaged point $(\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{Z_k})$ associated with a given point in the observed scene moves to a neighbouring Z-level in the range $[\boldsymbol{Z_{k-h}}, \boldsymbol{Z_{k+h}}]$, as can be seen in Figure 3.5 (b). In addition, for every capture in any scene, there is a finite set of depth values. In other words, the possible discrete depth values that may be encountered in the output data can be predicted. As explained above, the Kinect sensor works in a discrete set of depth elements Z-levels. Every level constitutes a partition of the ensemble of points within a frame, Equations (3.7), (3.8), (3.9), and has the property of being entirely independent of the neighbouring levels and orthogonal to the $\boldsymbol{z}$-axis, Equation (3.10). As a result, a point cannot be found out of these parallel

planes. This is what really appears in all the scanned data if the scene is rotated over $\boldsymbol{x}$ or $\boldsymbol{y}$ axes; i.e. the points lie in fronto-parallel planes. The importance of such findings for Kinect based applications is that the relationship between depth measurements over different frames can be studied. That is, if it is found that two successive depth measurements are related with a linear mapping, i.e., the second condition required to adapt Kalman filter would have been fulfilled.

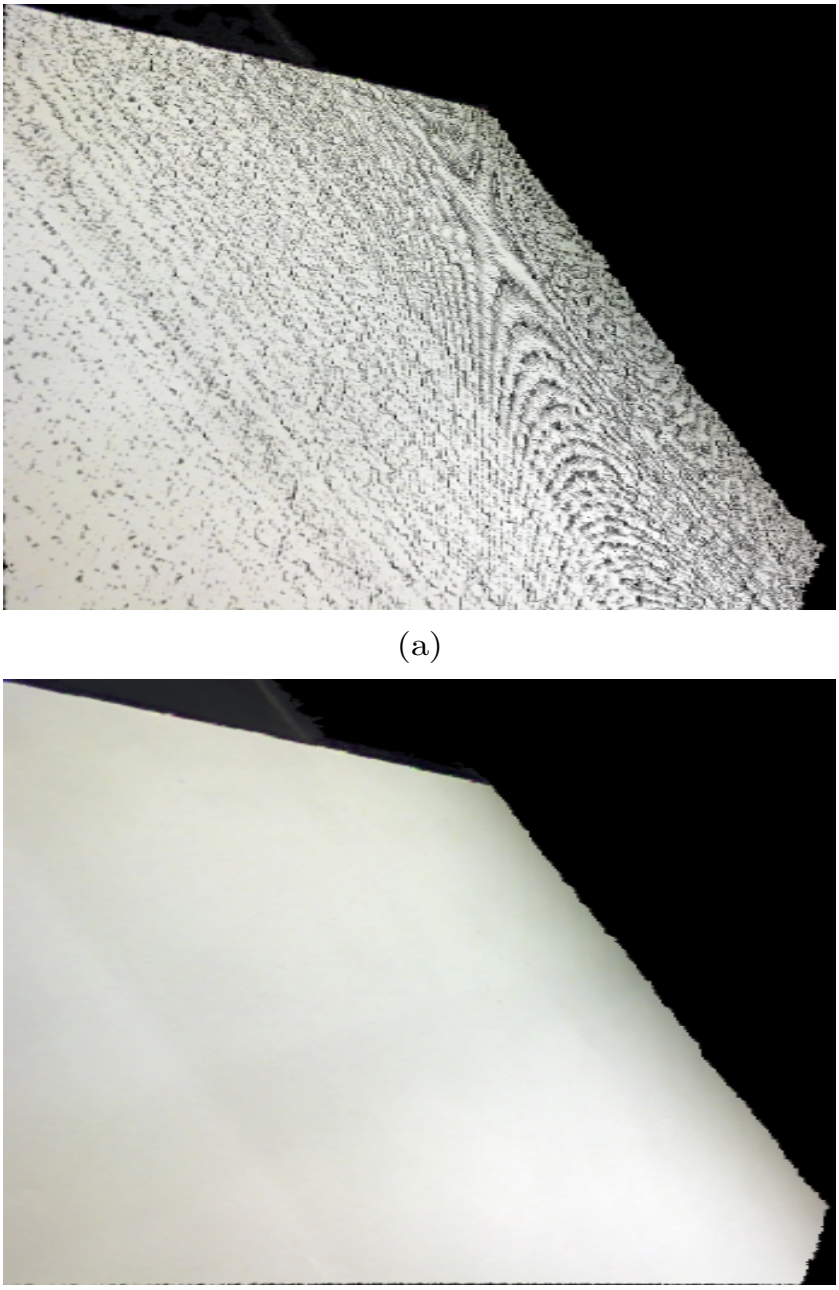
When the sensor is stationary, the depth map keeps changing because of points jumping from one Z-Level to another, Figure 3.6. The destination level is not necessarily adjacent but within a limited radius, Equation (3.11).

Even when the points change their depth level, the 2D $(\boldsymbol{u_i}, \boldsymbol{v_i})$ image coordinates on the screen remain identical. Nevertheless, their world counterparts $(\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{z_i})$ vary. This is true because every point $\boldsymbol{P_i}$ $(\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{z_i})$ in the 3D world lies on a line originating from the centre of the camera passing through the pixel $(\boldsymbol{u_i}, \boldsymbol{v_i})$ on the screen towards the scene, Figure 3.6, following the direction of the perspective frustum [78]. Using this information, the relationship between two successively measured 3D coordinates can be inferred. From the intrinsic parameters of the camera $(\boldsymbol{f_x}, \boldsymbol{f_y}; \boldsymbol{c_x}, \boldsymbol{c_y})$, two equations can be obtained:

$$\begin{cases} u_i = (f_x / z_i) x_i + c_x \\ v_i = (f_y / z_i) y_i + c_y \end{cases} \qquad z_i \neq 0 \tag{3.12}$$

As pixel coordinates in two successive frames remain the same, from Equation (3.12) with $\boldsymbol{z_i} \neq \boldsymbol{0}, \boldsymbol{z_i^+} \neq \boldsymbol{0}$; $(+)$ means the new coordinates in the following frame are:

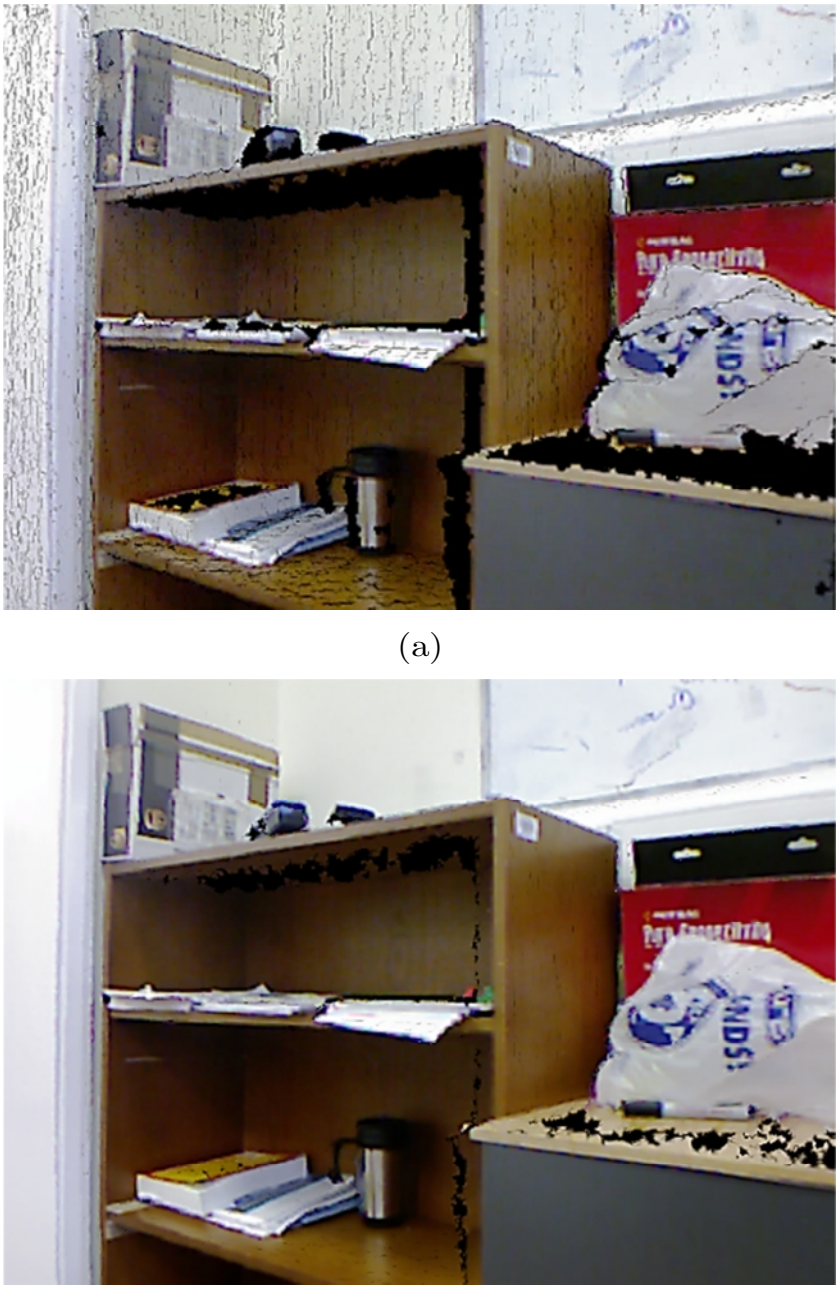
$$\begin{cases} u_i^+ = u_i \xrightarrow{yields} (f_x / z_i^+)\ x_i^+ + c_x = (f_x / z_i) x_i + c_x \\ v_i^+ = v_i \xrightarrow{yields} (f_y / z_i^+)\ y_i^+ + c_y = (f_y / z_i) y_i + c_y \end{cases} \tag{3.13}$$



(a)

(b)

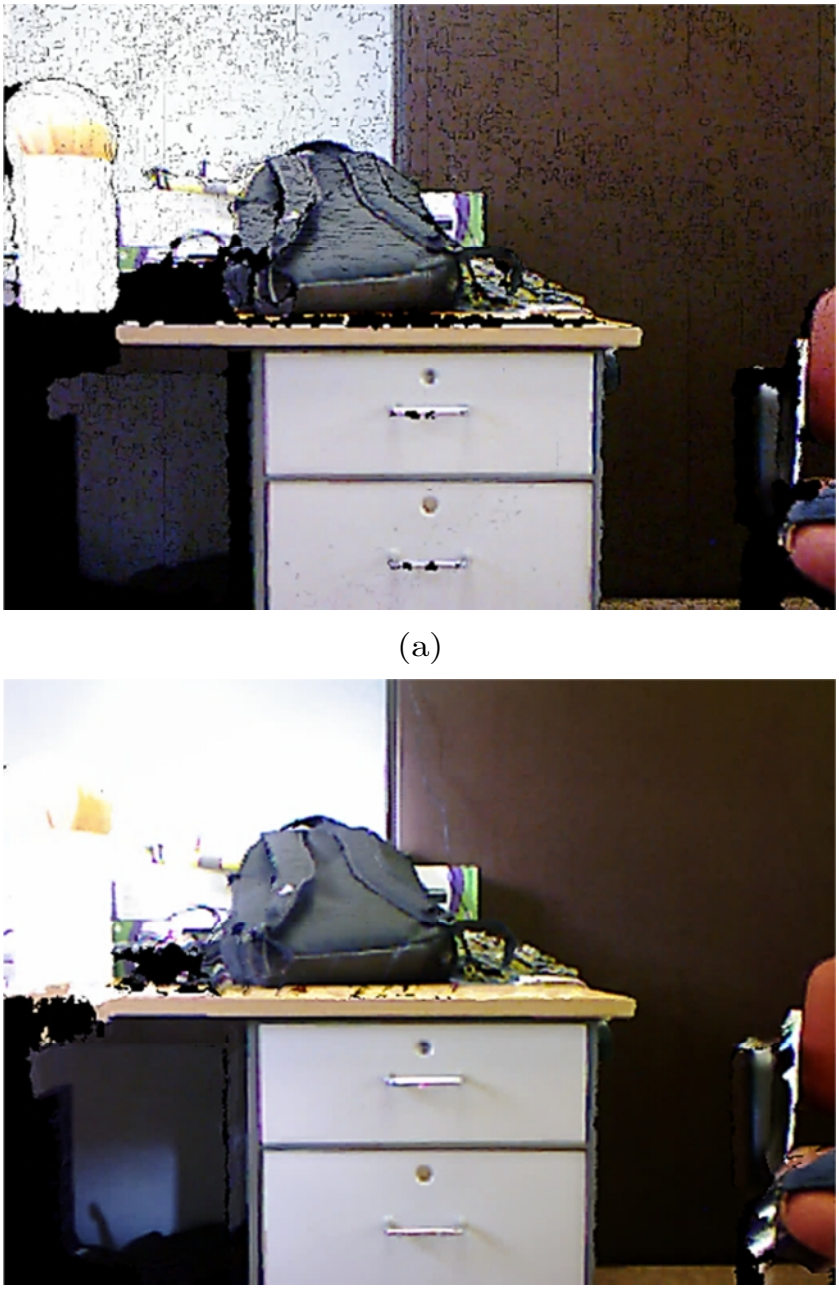
Figure 3.7 The smoothing effect of Kalman filter on the "flat panel" scene. **(a)** Before. **(b)** After



(a)

(b)

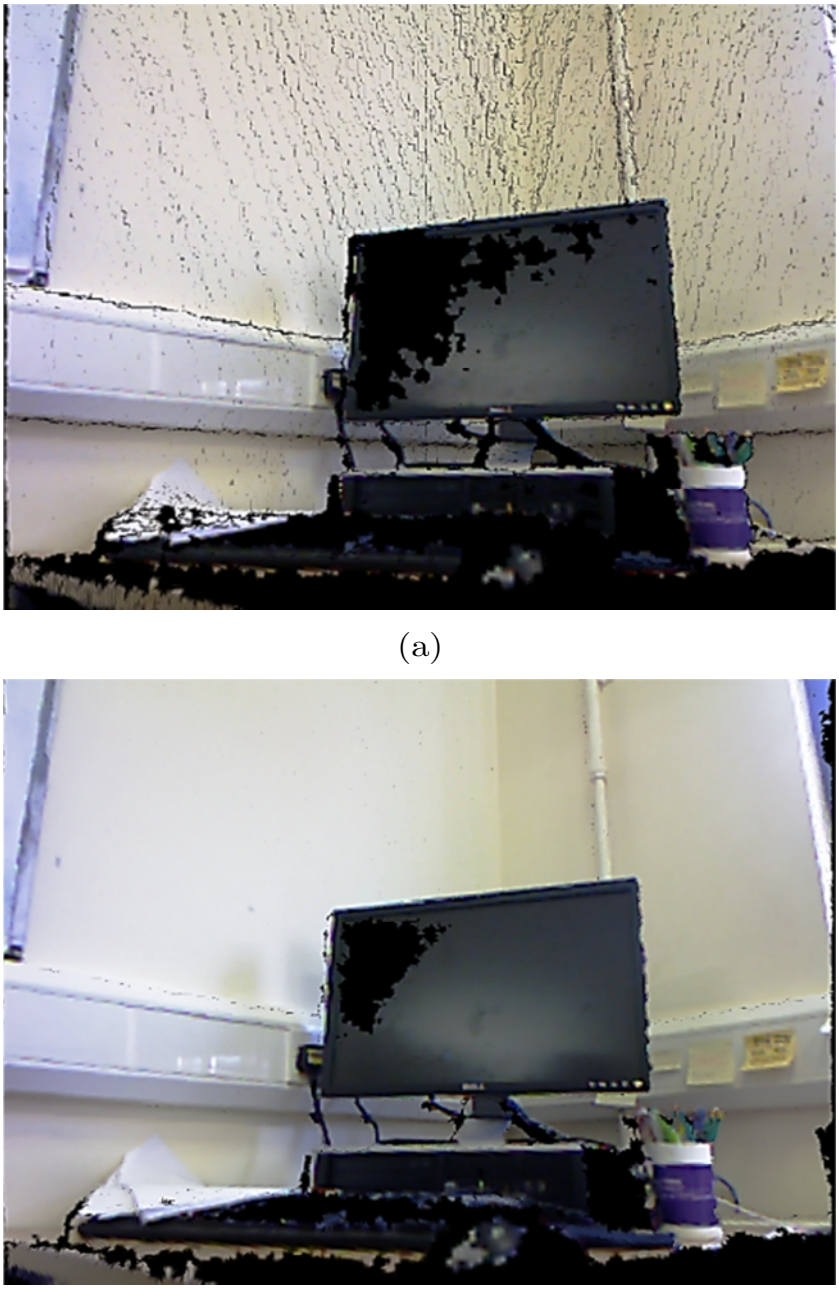
Figure 3.8 The smoothing effect of Kalman filter on the "Shelves" scene. **(a)** Before. **(b)** After



(a)

(b)

Figure 3.9 The smoothing effect of Kalman filter on the "Desk" scene. **(a)** Before. **(b)** After



(a)

(b)

Figure 3.10 The smoothing effect of Kalman filter on the "screen" scene. **(a)** Before. **(b)** After

Finally, after simplification of Equation (3.13) it can be seen that:

$$\begin{cases} x_i^+ = \left(\frac{z_i^+}{z_i}\right) & x_i \\ y_i^+ = \left(\frac{z_i^+}{z_i}\right) & y_i \end{cases} \tag{3.14}$$

Equation (3.14) proves the linearity between points projected at the same pixel on the screen over different frames. Adding this to the Gaussian nature of noise, the Kalman filter can be safely adapted as a real-time filtering tool for RGBD sensors. The effect of the filter can clearly be seen in Figure 3.7 through to Figure 3.10.

3.5.4 Kalman Filter Adaptation to Kinect Sensor

For a given pixel within the frame at time step $\boldsymbol{t}$, $\boldsymbol{X_t}$ is the state estimate; $\boldsymbol{z_t}$ is the measured depth; $\boldsymbol{P_t}$ is the a priori estimate-error covariance and $\boldsymbol{K_t}$ is the Kalman gain. There is one to one scalar correspondence between state-measurements, so $\boldsymbol{H = 1}$. Moreover, $\boldsymbol{A = 1}$ as the depth should not change beyond the magnitude of noise ($\boldsymbol{\sigma_k}$) between two successive frames. $\boldsymbol{B = 0}$ for a fixed sensor. As the system is modelled accurately, we assume a small process noise whose covariance $\boldsymbol{Q = \varepsilon I}$, ε is a small positive number that was set to $\boldsymbol{10^{-8}}$ in our experiments. $\boldsymbol{R_{t,k} = \sigma_k{}^2}$ covariance of observation noise ($\boldsymbol{\sigma_k}$ is the standard deviation describing the magnitude of noise around the expected $\boldsymbol{z_t}$, and differs from one Z-level to another proportionally to the distance from the sensor). For a static sensor/dynamic scene setup, Kalman Equations (3.1) to (3.6) become:

Prediction

$$X_t = \widehat{X}_{t-1} \tag{3.15}$$

$$Y_t = z_t \tag{3.16}$$

$$P_t = \widehat{P}_{t-1} \tag{3.17}$$

Correction

$$K_t = P_t(P_t + R_{t,k})^{-1} \tag{3.18}$$

$$\widehat{X}_t = X_t + K_t(Y_t - X_t) \tag{3.19}$$

$$\widehat{P}_t = (1 - K_t)P_t \tag{3.20}$$

From the equations above, one may think that at a given pixel, $\boldsymbol{z_t}$ does not change over time. Although important variations in depth occur if the observed object moves backward or forward, the filter optimises the depth estimates under the assumption that the dynamics of the scene do not abruptly change between two successive frames (the change is not sudden, i.e. $\Delta \boldsymbol{z_t} \leq \boldsymbol{\alpha} \times \boldsymbol{\sigma_k}$ within 33ms. For instance 33ms = 1/30 Hz). $\boldsymbol{\alpha}$ has been determined empirically. After experimental tuning, we found that $\boldsymbol{\alpha = 2.5}$ leads to the smallest error in depth estimation. In practice, this customisation of Kalman filter is generally verified because the scene contains physical objects that move gradually and continually.

The elementary displacements of these entities are small given the high frame rate of capture that is being ensured by implementing the filter on the GPU. After taking this into account, whenever the depth reading changes its levels, the filter updates the pixel's workspace ($\boldsymbol{X_t}, \boldsymbol{R_k}$, $\boldsymbol{P_t}$, $\boldsymbol{K_t}$). However, when the difference between two successive $\boldsymbol{z}$ values becomes greater than $\boldsymbol{2.5 \times \sigma_k}$, the same workspace is reinitialised according to the current value of depth. In other words, the object has jumped from Z-level $\boldsymbol{k}$ to Z-Level $\boldsymbol{k'}$ ($\boldsymbol{X_{t+1} = z_{t+1}}$, $\boldsymbol{R_{k'} = 2.5 \times \sigma_{k'}}$, $\boldsymbol{P_{t+1} = R_{k'}}$, $\boldsymbol{K_{t+1} = P_{t+1}(P_{t+1} + R_{k'})^{-1}}$). As a result, the steadier the scene, the better the filter performs. In real scenarios, most of the pixels within the depth image do not jump beyond the threshold at every frame. This fact helps the filter to operate smoothly in an indoor environment where the scene does not tend to change all the time.

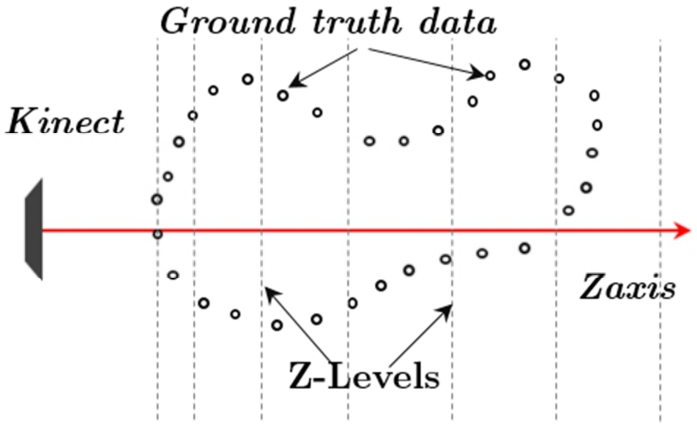
To validate these findings, the literature was checked for the most potential scenarios where RGBD data is used. As a result, it was found that there are mainly two classes of possible applications relying on depth sensors: The former uses the sensor as a depth measuring device, which is widely encountered in robotics and object tracking domains; the latter uses the camera as a scanning device, which is widely regarded in mixed reality and computer graphics applications. For this reason, the filter was tested on both classes of scenarios to assess its added-value to their innate performance.

3.6 Kalman Filter Effect on RGBD Data for Moving Vehicles Tracking

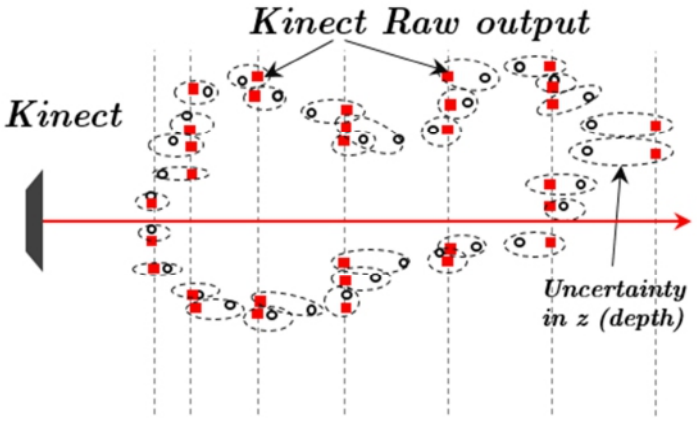
Tracking applications are very sensitive to position accuracy of the tracked entity. However, Kinect raw data is not accurate enough to precisely localise an object within its neighbourhood. When the sensor acquires a point cloud, the 3D data is automatically distributed over the discrete Z-Levels, Figure 3.11 (a). Original point data comes from the continuous real world. The corresponding images in Kinect's space lie in the sensor's parallel planes, Figure 3.11 (b). The error in measurement is therefore proportional to the gaps between Z-Levels where the 3D points are projected.

The Kalman filter takes these noisy raw data as input, optimises them to approach as closely as possible their real world positions, see Figure 3.11 (c). The performance of the filter can be clearly seen in Figure 3.12. The latter depicts the raw and the filtered trajectories for a moving robot tracked by the same Kinect camera. The blue points in Figure 3.12 (a) represent the measured positions taken by the robot. If the deepest points are carefully observed (greater z_i), it is noticed that the gaps between the parallel Z-Levels are larger. Indeed, this is due to the drop in resolution as one gets far away from the camera.

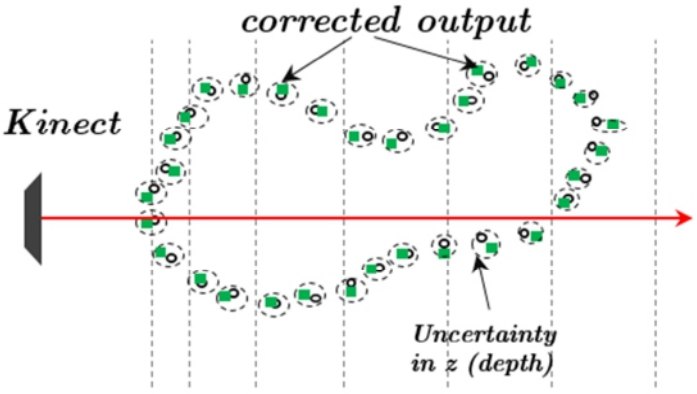
However, Kalman filter's smoothing effect, as seen in Figure 3.12 (b), optimally condenses the sparse and discrete points around their relevant real world true



(a)



(b)



(c)

Figure 3.11 Kalman effect on Kinect's data for object tracking applications. **(a)** Ground truth trajectory. **(b)** Raw Kinect trajectory. **(c)** Filtered trajectory

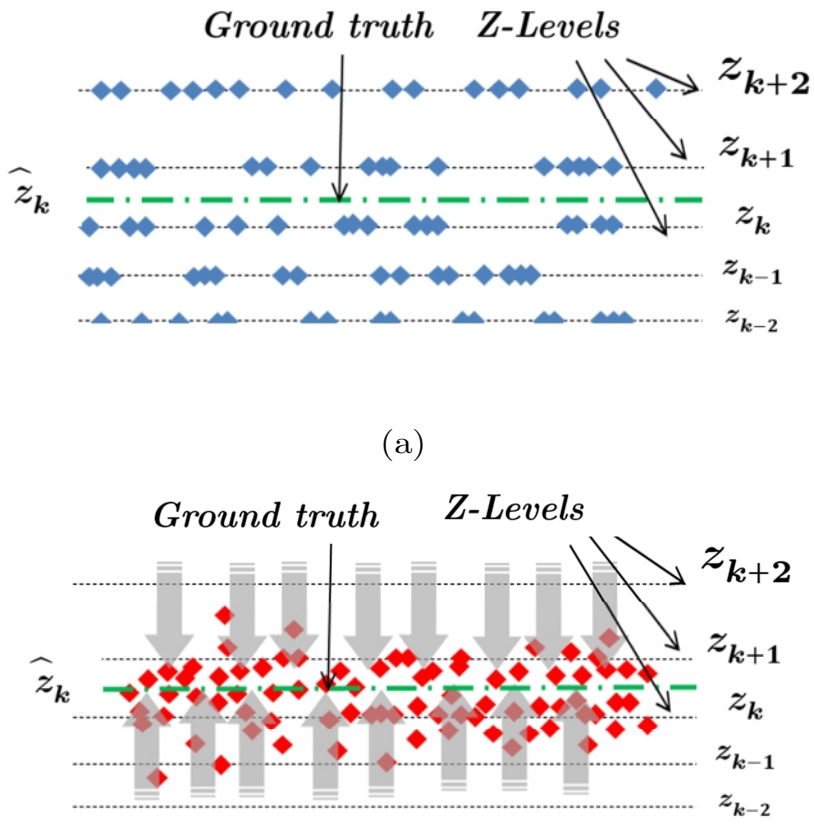
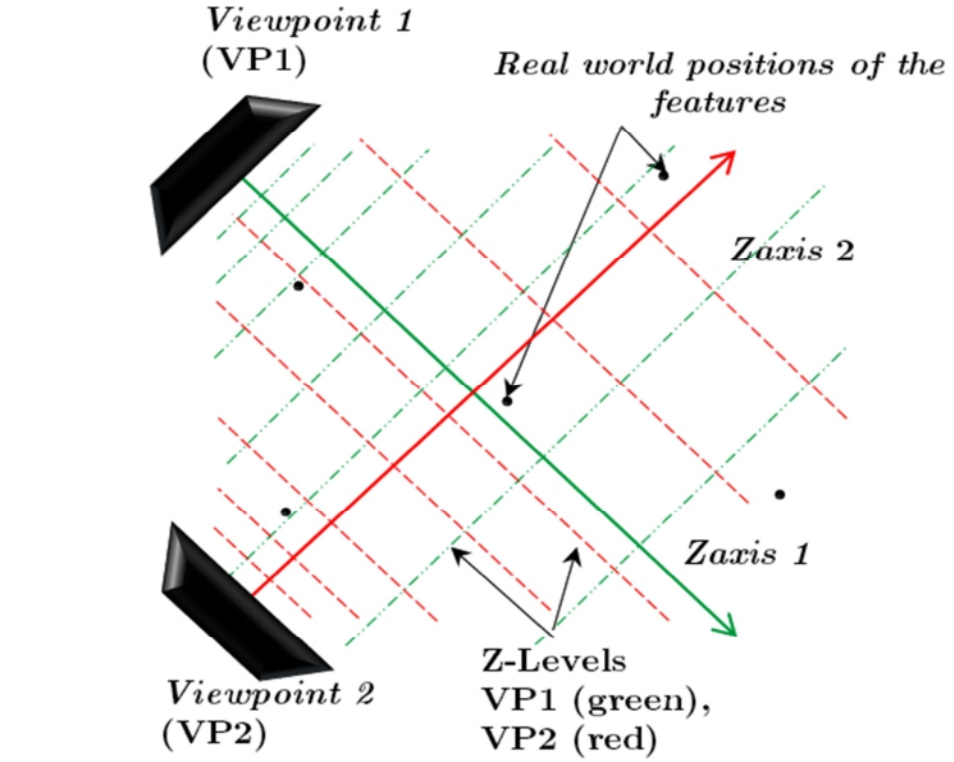


Figure 3.12 Kalman effect on position data. **(a)** Raw points. **(b)** Filtered points

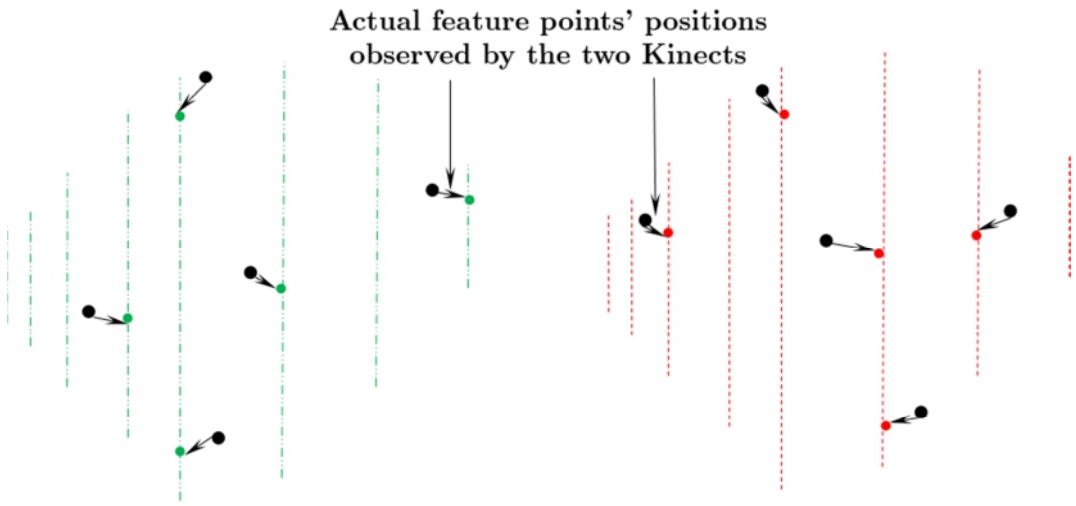
measurements. This behaviour reduces the error in the 3D point cloud. As a result, the position of the tracked object becomes more precise and reliable.

3.7 Kalman Filter Effect on RGBD Data for Depth Image Registration

Image registration is necessary to reconstruct 3D models of real objects for simulation, virtual and augmented reality applications. Feature extraction and matching are the essential tools to find the respective correspondences between two different images before alignment takes place, see Figure 3.13 (a).



(a)



(b)

Figure 3.13 Kinect sensing of 3D feature points. **(a)** Our setup with world positions of the features. **(b)** Features projected on Z-Levels. Here, the two viewpoints refer to the same camera looking at the scene from two different angles

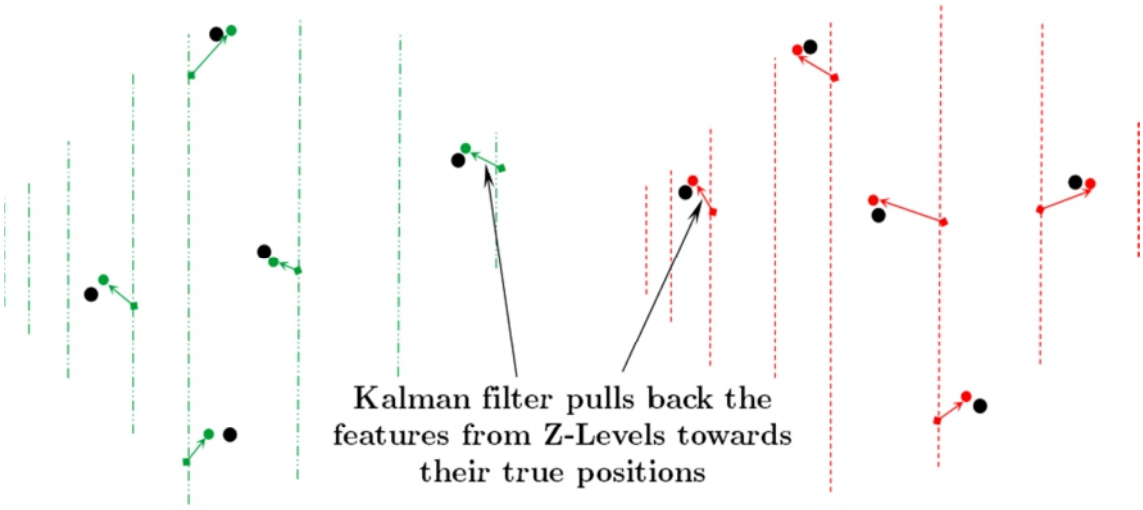


Figure 3.14 Kalman filter influence on the captured points

3D reconstruction applications using Kinect can only be accurate at a close range. This inaccuracy is unacceptable by most registration applications, as it widely exceeds the alignment error. To deal with this low-quality data, many filters exist in the literature, but up to now, none was entirely adapted to work on Kinect.

Figure 3.13 (b) illustrates the fact that the captured points are distributed on parallel planes (as explained earlier). When the same real world feature is detected within multiple point clouds, its respective projections in the different views rest in the discrete Z-levels, Figure 3.14. The 3D data of the scene are discretised because of the nature of the limited resolution regarding the sensor. These levels do not necessarily correspond to the correct locations. The gaps between the scanned positions and the actual ones increase exponentially as one gets further away from the sensors. As a result, similar features within different point clouds will be wrongly matched, and alignment error accordingly grows, see Figure 3.15 (a).

On the other hand, the application of Kalman filter optimises the positions of the tracked features and produces the best possible result given hardware limitations. Kalman filter optimally places the discrete points in the continuous real space (off the Z-Levels), see Figure 3.14. For this reason, the features

obtained from different viewpoints show closer 3D geometric properties, and the subsequent registration is achieved with less error, see Figure 3.15 (b). The Kalman filter is more useful at greater ranges because the Kinect accurately measures the depth at small ranges (below 1.5m).

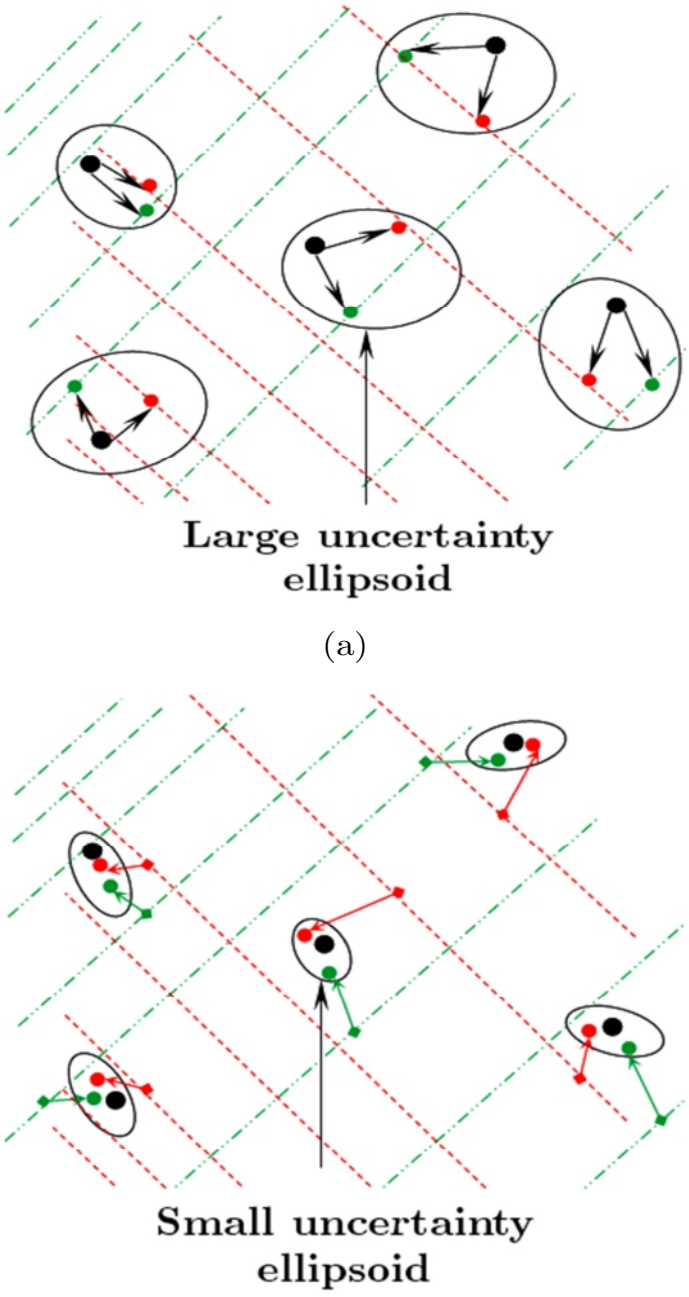


Figure 3.15 Registration error. **(a)** Raw Kinect data. **(b)** Filtered data

3.8 Results & Discussions

The discussions below are based on the following hardware configuration: an Intel i7 3930K CPU with six physical cores (two logical cores per physical) running at 3.20 GHz. 16.0 GB of RAM along with an NVidea GeForce 2GB GTX 680 GPU.

Our results were compared with the Moving Average Filtering [79]. This method is designed to smooth a series of noisy or incomplete data, just as the Kalman filter does. However, the optimality is not guaranteed even with Gaussian noise. The results show that our technique outperforms the moving average filter. Such an advantage is due to the optimality ensured by the Kalman filter when correctly adapted to the problem.

Image data is more naturally organised to fit GPU thread blocks. Every element in the block (Thread) processes a single pixel at a time [80]. Figure 3.16 illustrates how the depth map delivered by the camera is divided into image blocks of a constant size (16×16 pixels. Thus, 256 threads is the size in of the block in this implementation). The pixels of the same image block are processed simultaneously in the same GPU thread block. As a result, a thread in the GPU is attributed to every pixel in the depth map. The latter runs the actual filtering (KF) on a single depth pixel (range reading) and saves the necessary data for the next frame ($\boldsymbol{X_t}$, $\boldsymbol{P_t}$). This scheme is straightforward because there are no constraints amid the pixels and the order in which they should be processed. Otherwise, more specific techniques should be applied to benefit from the parallel computing ability of GPUs. Processing complexity is reduced to that of the algorithm running in the thread (Kalman filter), which is indeed constant.

3.8.1 GPU Implementation of Kalman Filter for Depth Map Filtering

After establishing the theoretical feasibility of the main ideas, it has been found that the usual implementation on the CPU generates latency. This problem induces a decrease in the native frame rate of the sensor. When the filter was

first run on a regular CPU, the maximum reachable frame rate was 17 FPS. The need to implement the solution on the GPU has therefore emerged. The GPU was used instead because of its adequacy of fitting image processing problems and efficiency of preserving the real-time property of the system. As a result, the filter is eventually capable of processing VGA resolution (640×480 pixels) depth images at the same frame rate as the camera. The enhanced results allow the following applications to exploit the frame rate offered by the sensor (30 FPS) entirely.

Other optimisations should be addressed to profit fully from the utilisation of all the available hardware capability. The design of heterogeneous algorithms aims at a higher occupancy of the processors, as well as an extensive usage of the bandwidth when exchanging data between the central memory (RAM) and the global memory of the GPU (GMGPU)[81]. To this end, two optimisation aspects have been focused on:

Running asynchronous transfers: When the GPU is processing the current frame, the bus linking it to the central memory is entirely free. This idle state can be exploited to exchange data. In other words, the following frame ($\boldsymbol{f_{t+1}}$) is sent from the RAM to the GMGPU and the already available result ($\boldsymbol{f'_{t-1}}$) is sent back to the RAM. Simultaneously, the current frame ($\boldsymbol{f_t}$) is being processed on the device (GPU), see Figure 3.17 (a).

Memory coalescing: The GPU automatically loads the content of adjacent memory cells because its internal design assumes that it is very likely for neighbouring data within the same area to be soon requested as well [82]. Memory coalescing is another optimisation measure that significantly helps increasing the probability of threads in the same warp (a group of 32 threads from the same thread block running simultaneously) to access the memory together. The purpose of coalescing is to ensure that the threads access the same memory segment to only pay a single memory transaction. However, if

they request sparse locations, it would cost the GPU as many transaction as the number of sparse addresses.

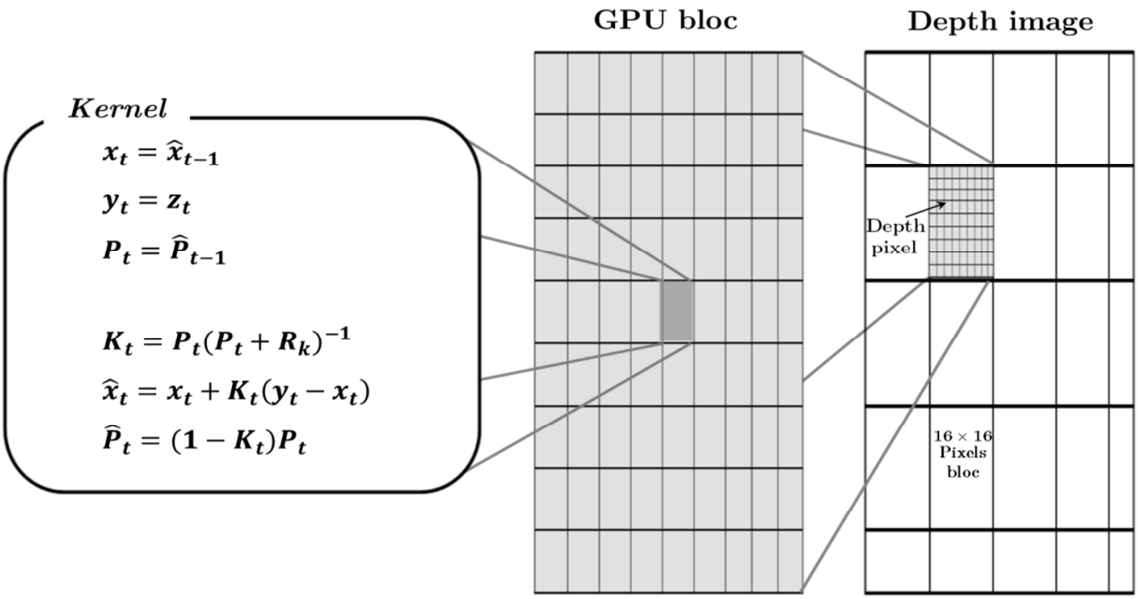
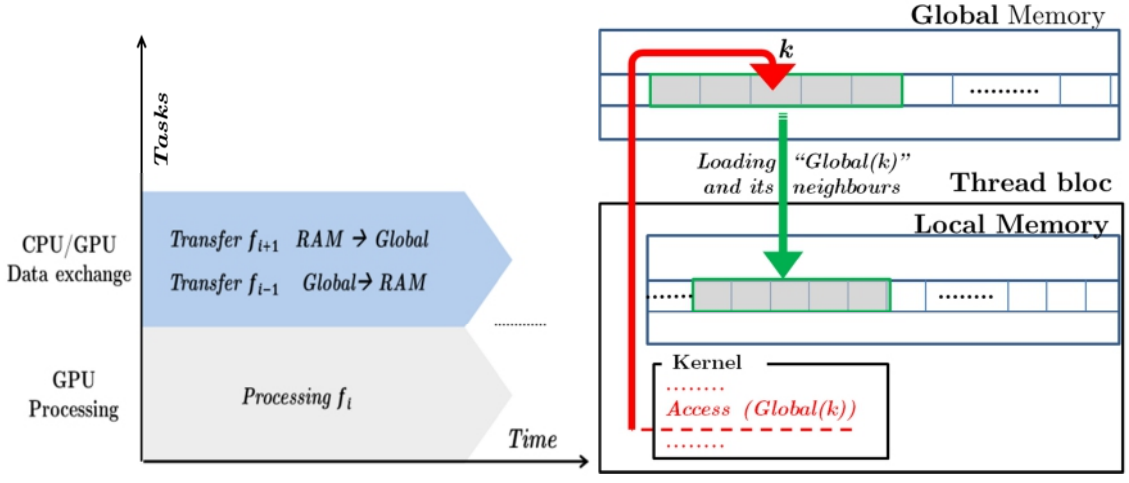


Figure 3.16 Kalman filter GPU implementation for depth map filtering



(a) (b)

Figure 3.17 Data exchange optimisations in the GPU. **(a)** Asynchronous transfer RAM/GMGPU. **(b)** Data loading GMGPU→ local memory

Appropriately organising the data in the GMGPU allows such a contiguous access to happen automatically. Structure of Arrays [83] instead of the easy to use Array of Structures significantly increases the chances of loading a chunk of memory containing the data not only for the thread which has requested it, but also for its neighbours in the warp. Figure 3.17 (b) illustrates what happens when a thread requests the content of a given cell in the global memory.

Figure 3.18 illustrates GPU/CPU benchmarking for the three tracking scenarios that will be discussed later. The outcome of GPU's implementation can be clearly seen. The whole frame rate of tracking is just below 30 FPS (almost one frame processed every 33ms).

3.8.2 Expending Sensor's Field of View

Another advantage, of using a Kalman filter, is extension of the native operational range regarding the sensor. The filter can compensate for the lack in accuracy at a larger range. As shown in Figure 3.19, an extra 1.5m could be afforded without any additional hardware improvement. As a result, the reachable space becomes broader, with a better accuracy. The latter remains proportional to the square of the depth, but its slope becomes less important compared to the raw measurements.

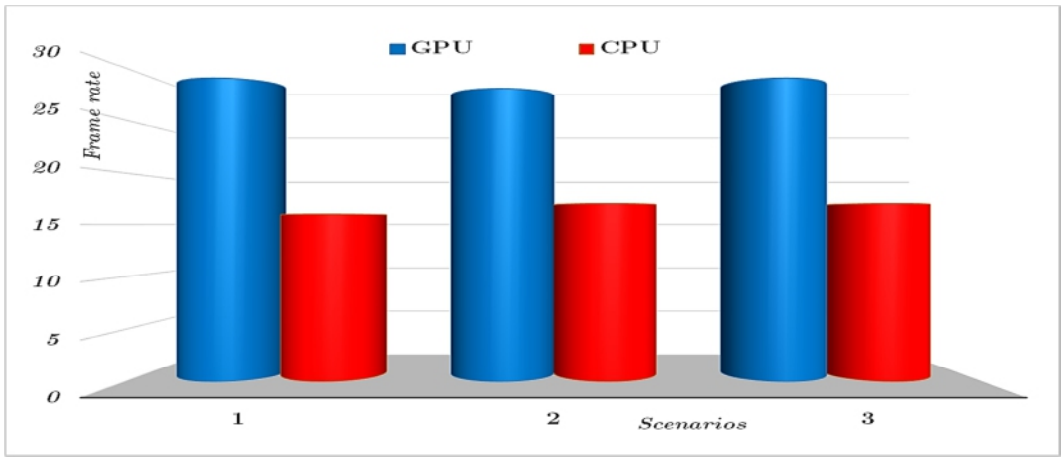
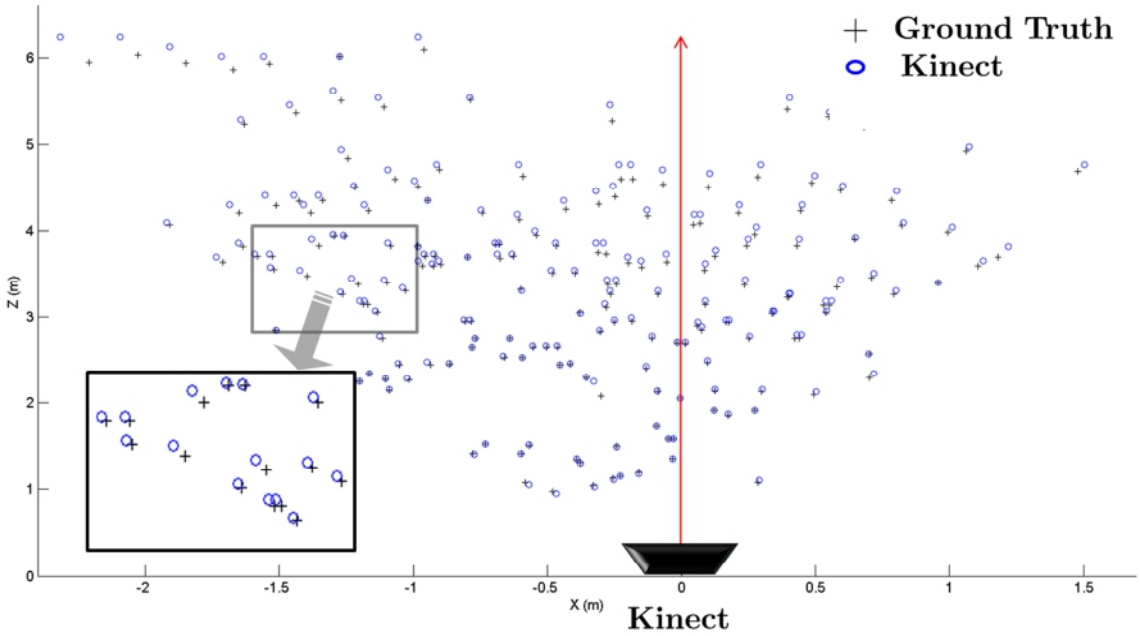
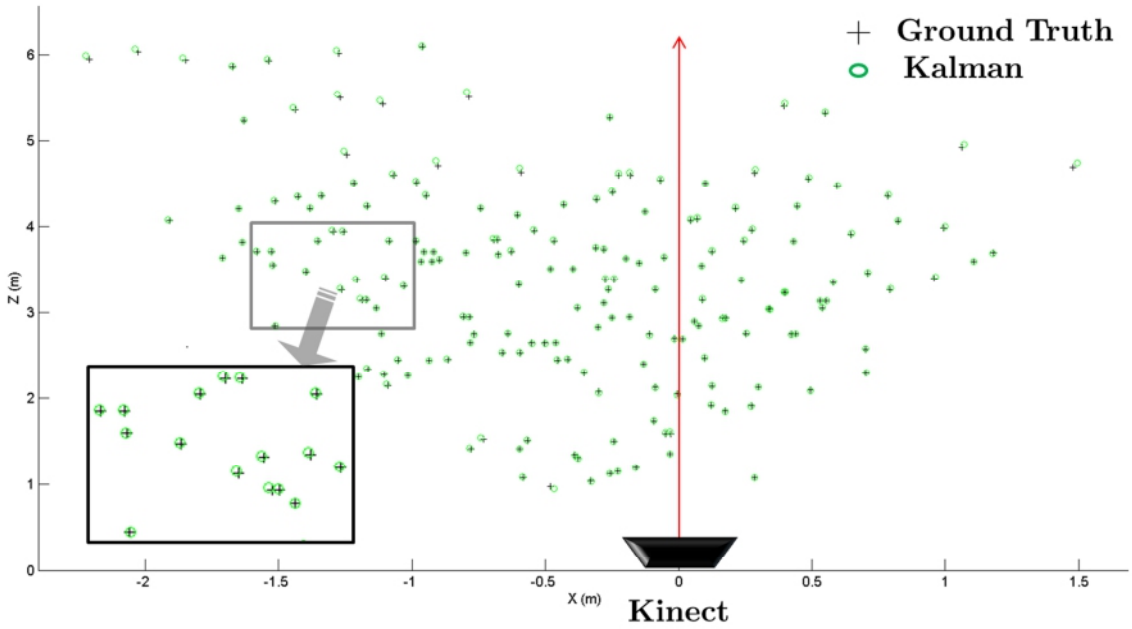


Figure 3.18 CPU/GPU benchmarking of KF for Kinect



(a)



(b)

Figure 3.19 Kalman filter effect on 3D points localisation at different distances. **(a)** Raw measurements. **(b)** Kalman correction

More importantly, the filter can be implemented to work in real-time. Hence, its processing load does not affect the frame rate of the sensor significantly (it filters 29.5 frames on average out of a total of 30 every second).

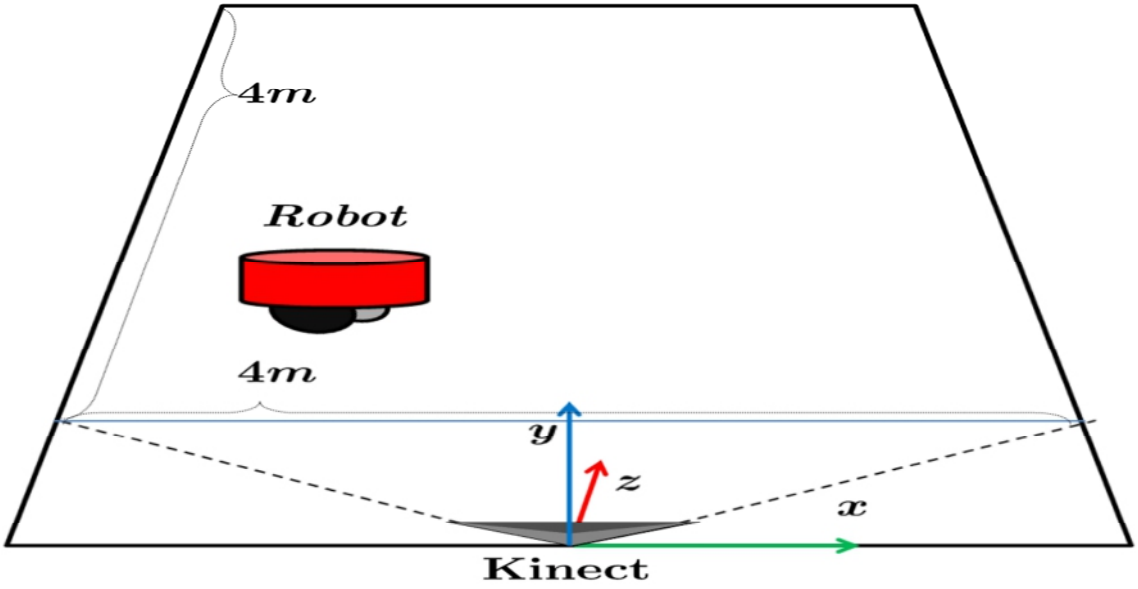


Figure 3.20 Robot tracking experimental setup

Note that when the range of 7.0m is exceeded, the Kinect's outputs become very noisy. The filter cannot handle this very low quality of measurements anymore. Such an inadequacy is a pure hardware limitation to fit the affordable price of the sensor [45].

3.8.3 Object Tracking Applications

To validate the Kalman filtering effect on Kinect data, an experiment, where a moving vehicle was tracked by one Kinect camera, was conducted, see Figure 3.20. The robot moves in a closed space of 4×4 m^2. The objective is the determination of its global position within the surrounding environment whilst scanning. The solution was tested against a Moving Average Filter (MovAve) [79] in order to justify the rationale of fitting Kalman equations to Kinect sensor.

The two components required for robot localisation are $\boldsymbol{z}$ and $\boldsymbol{x}$ coordinates. Even though $\boldsymbol{y}$ is almost unchangeable over time for ground robots, it can be included in the filter without any further restriction. The purpose of this approach is to test the accuracy of the Kinect in issuing three-dimensional positions for a given object in real-time. The filter affects only on the depth

data ($\boldsymbol{z}$ component in this setup). Afterwards, the computation of the two remaining coordinates is based on $\boldsymbol{z}$ and the calibration parameters of the IR camera, Equation (3.12). For the sake of generality, the tracker was tested on three different scenarios where the robots were moving in front of the camera (i) in a circular motion, (ii) Left to Right (swinging back and forth) and (iii) Front to Back (swinging left and right). For every scenario, the ground truth trajectory undergone by the vehicle along with the raw position and the filtered trajectories resulting from the Kalman filter (KF) were plotted as well as the moving average filter (MovAve).

To assess the accuracy of this approach, the Root Mean Square Error (RMSE) was evaluated for both $\boldsymbol{z}$ and $\boldsymbol{x}$. The results can be seen in Figure 3.21 to Figure 3.29.

The OptiTrack[3] (Chapter 2) system was used as a high precision ground truth reference. In all the three scenarios, it is plausible that $\boldsymbol{z}$ and $\boldsymbol{x}$ graphs, filtered with KF, are contaminated with less error due to KF optimal smoothing effect. Moreover, the further the robot gets from the camera to the upper left and right corners, the higher the error becomes. The noisy fragments of the trajectory correspond to peaks in $\boldsymbol{z}$ and $\boldsymbol{x}$'s error plots. Ultimately, the filtered trajectory (red) is always the closest to the ground truth (black). Its RMSE is also smaller than MovAve's one. In other words, at every position KF-RMSE is at least 1.0cm less than MovAve-RMSE Table 3.1. The raw data (blue) is sparser and un-steadier. Its corresponding error is remarkably larger. Kinect accuracy is acceptable in the range below 3.5m from the sensor (Maximum error 5cm). However, when the tracked object moves beyond this limit, the error increases dramatically to up to 20cm at 4.5m. The actual operating range of the sensor is (0.8m to 3.5m) out of which Kinect is not meant to work properly [2].

[3] https://www.naturalpoint.com/optitrack/. 2015

Scenarios		MovAve (cm)	KF (cm)	Difference (cm)
Scenario 1 (circle)	z	4.03	2.81	1.22
	x	8.17	5.63	2.54
Scenario 2 (left-to-right)	z	4.03	2.81	1.22
	x	8.17	5.63	2.54
Scenario 3 (front-to-back)	z	4.52	3.32	1.2
	x	8.94	5.88	3.06

Table 3.1 RMSE results of the tracking scenarios

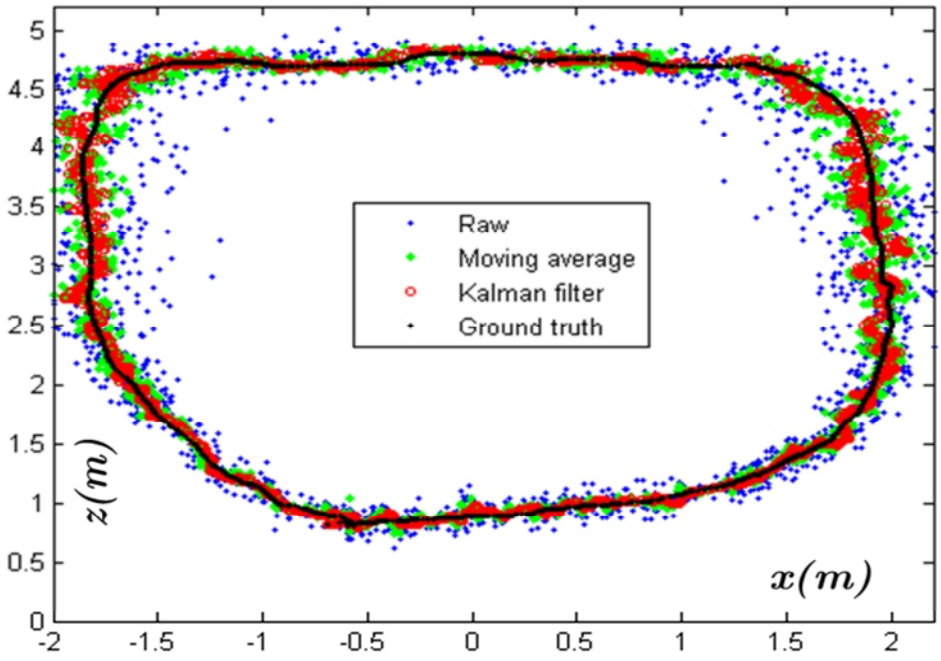
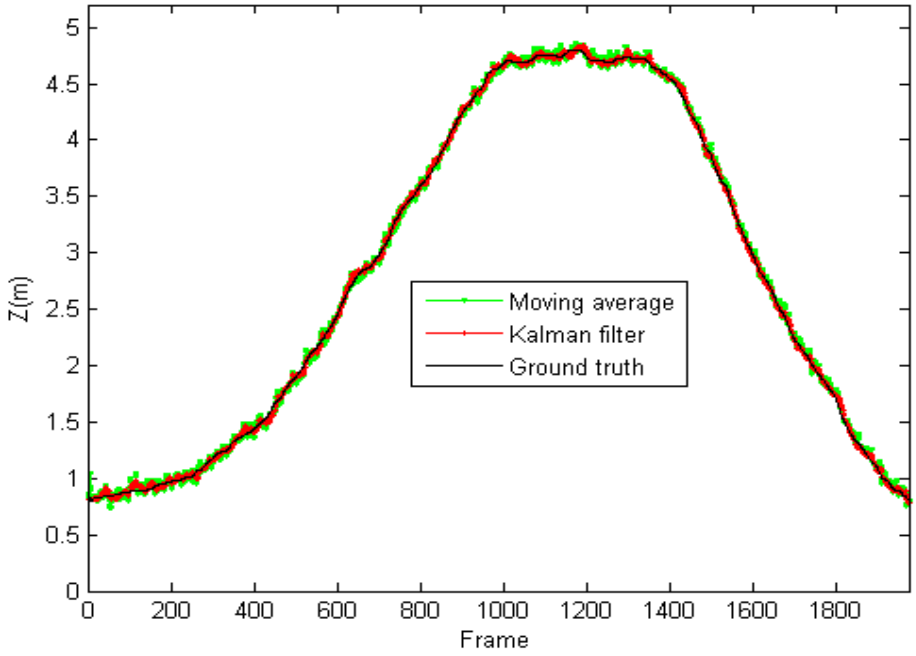


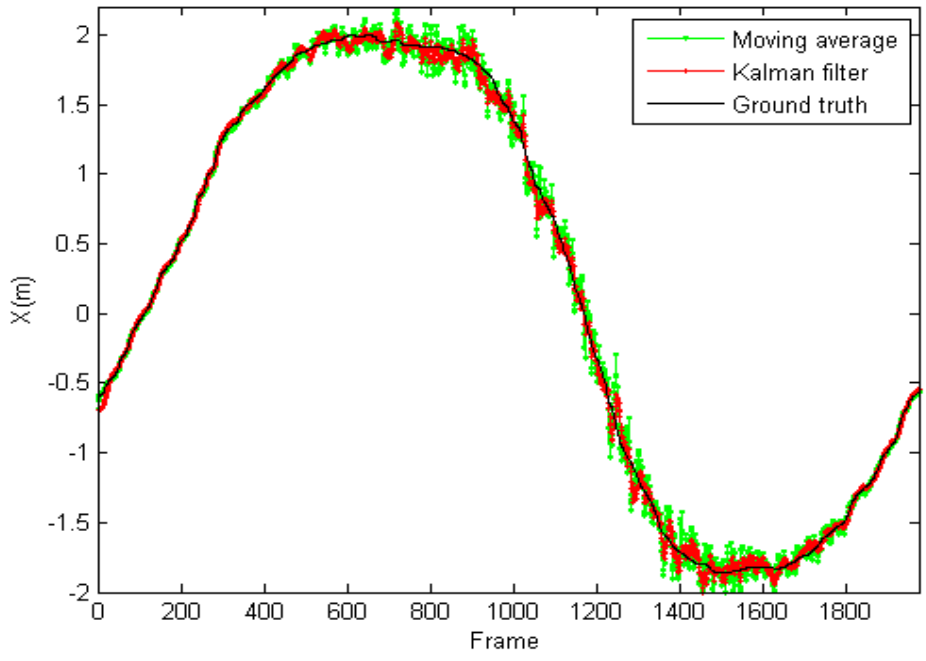
Figure 3.21 trajectories of the vehicles for tracking scenario 1 (circle)

3.8.3.1 Scenario 1 (Circle)

From an object tracking perspective, the aim of this scenario is to test the effectiveness of the filter at the boundaries of the working space. In addition, the closed shape of the trajectory gives an insight into prospective deviations of measurements over time. For instance, small frame-to-frame position estimation errors accumulate over time and may cause the estimated trajectory to diverge from the ground truth one. In other words, a correct loop closure means that the algorithm is drift-free. From Figure 3.21, one can see that the sensor delivers the worst measurements at the deepest extremities of $\boldsymbol{x}$-axis. $\boldsymbol{z}$ has a bell-shaped curve as shown in Figure 3.22 (a), ranging from a minimum value of 0.8m (minimum quantifiable depth) to a maximum of just below 5.0m. For instance, the maximum measurable depth is 4.5m. $\boldsymbol{x}$ on the other hand, is periodic in the interval $[-2.0, +2.0]$m, Figure 3.22 (b). In general, the RMSE regarding $\boldsymbol{z}$ for both the Kalman filter and its moving average counterpart are within the range $[0.0, 0.05]$m and $[0.05, 0.1]$m, respectively, Figure 3.23 (a). Likewise, $\boldsymbol{x}$ error varies in $[0.0, 0.25]$m for Kalman and in $[0.0, 0.5]$m for MovAve Figure 3.23 (b). Overall, the Kalman filter gives better results for both components $\boldsymbol{z}$ and $\boldsymbol{x}$.

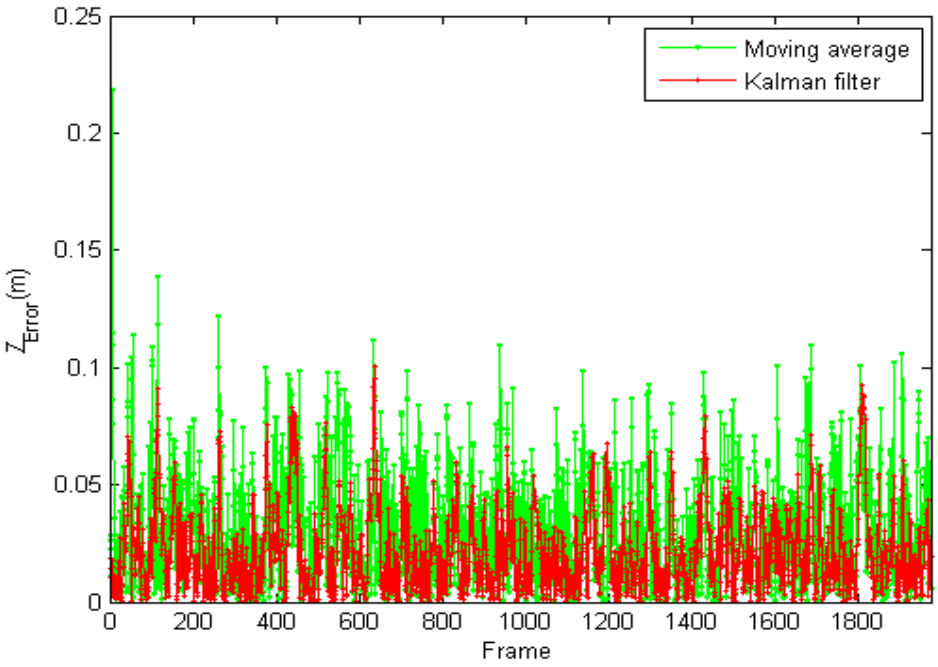


(a)

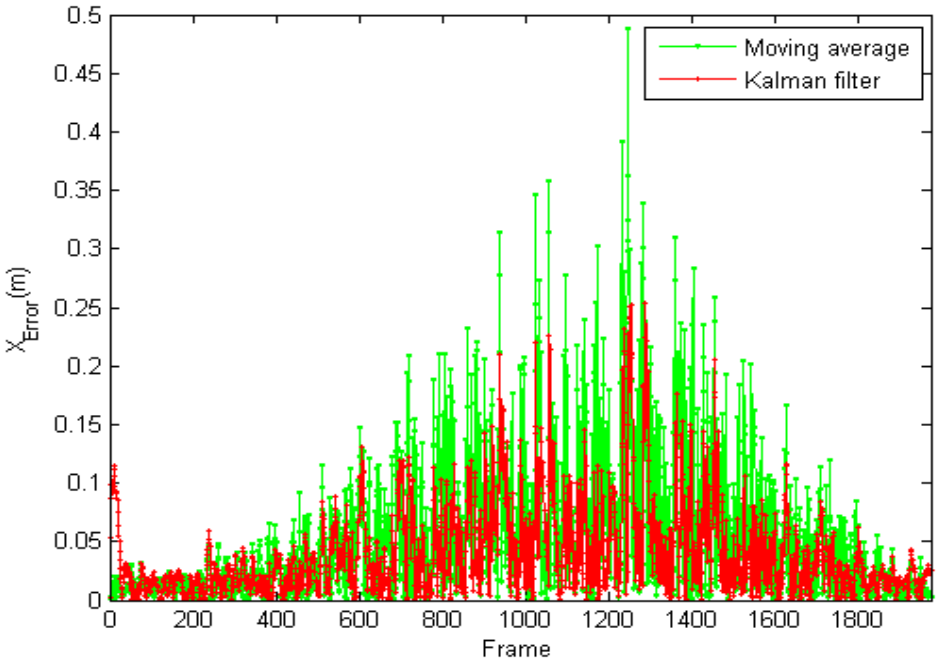


(b)

Figure 3.22 x and z variations. KF in red, MovAve in green and ground truth in black. **(a)** z. **(b)** x



(a)



(b)

Figure 3.23 Error graphs, KF in red, MovAve in green. **(a)** z. **(b)** x

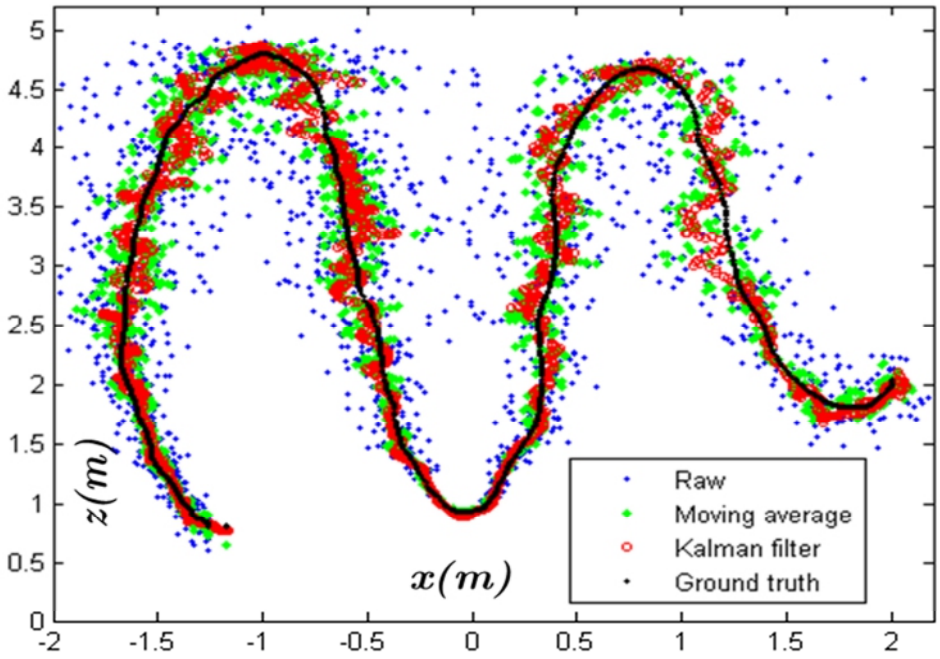
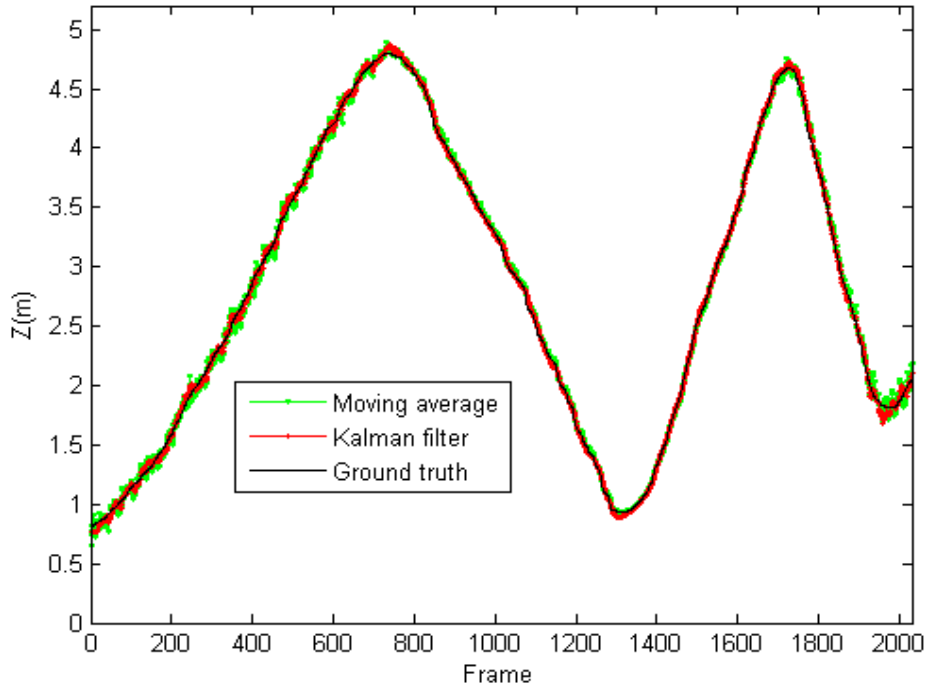


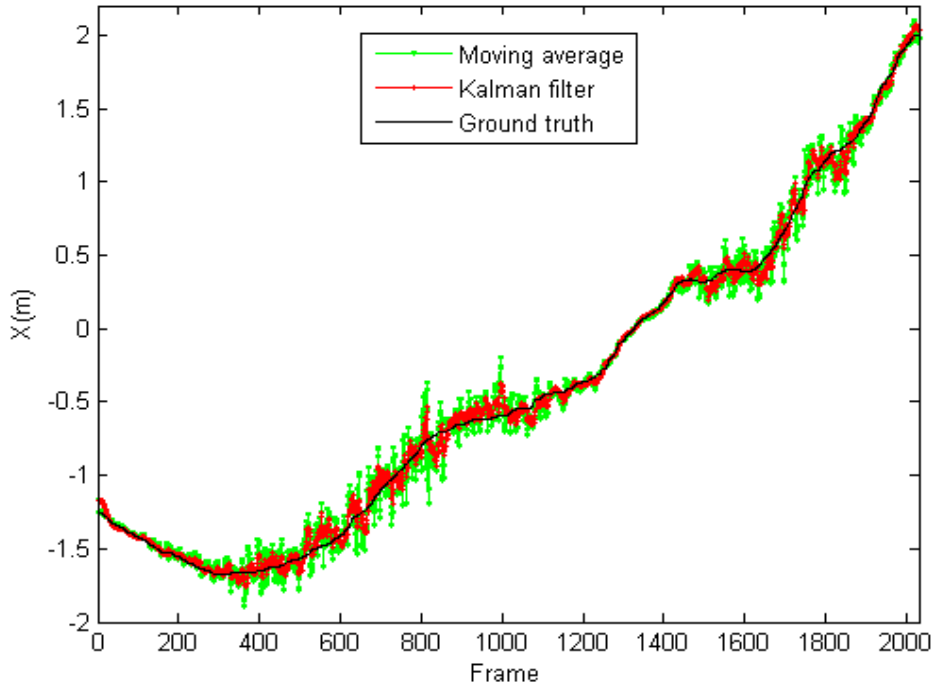
Figure 3.24 trajectories of the vehicles for tracking scenario 2 (Left-to-Right)

3.8.3.2 Scenario 2 (Left-to-Right)

The purpose of this scenario is to test the capabilities of the filter in the middle of the scene towards the positive direction of the x-axis (Left to Right). It is clear from Figure 3.24 that the inaccuracy of measurements streamed by the sensor becomes more significant at higher depth levels as well as at the two limits of the x-axis. For instance, z component variations are periodic with an extremal value of (~5m) at only two positions, see Figure 3.25 (a). Otherwise, z varies in the interval $[0.8, 5.0]$m. x, on the other hand, remains continually increasing within $[-1.75, +2.0]$m, Figure 3.25 (b). The RMSE of z component ranges between 0.0m and 0.08m for KF. Nevertheless, it marks an unusual level of 0.14m at a single position. MovAve error on the other hand, remains large throughout the whole period within [0.0, 0.16]m. It reaches its lowest levels when the robot is the closest to the camera, see Figure 3.26 (a).

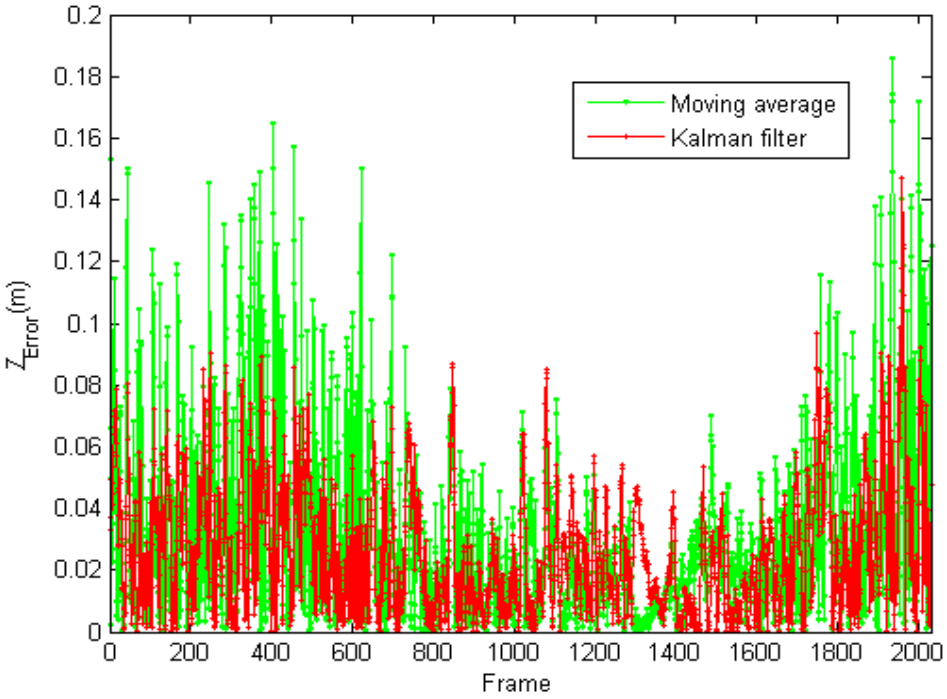


(a)

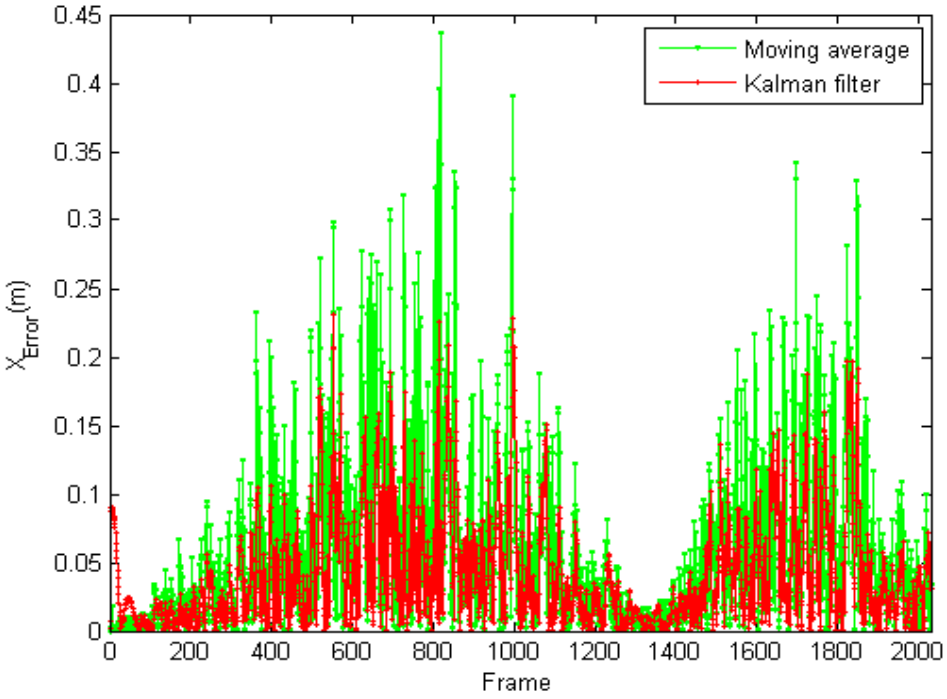


(b)

Figure 3.25 x and z variations. KF in red, MovAve in green and ground truth in black. **(a)** $\boldsymbol{z}$. **(b)** $\boldsymbol{x}$



(a)



(b)

Figure 3.26 Error graphs, KF in red, MovAve in green. **(a)** z. **(b)** x

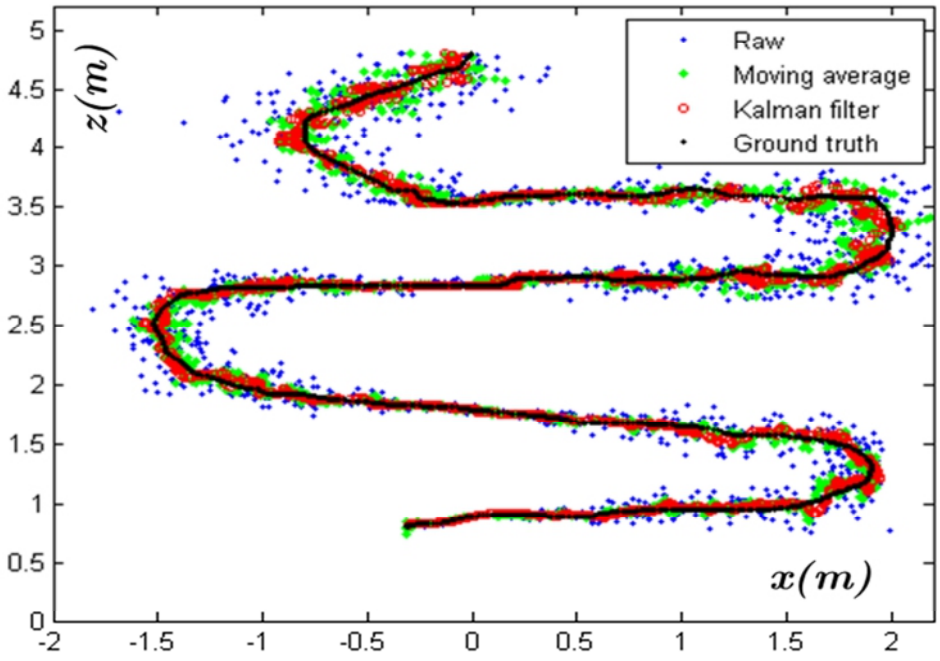


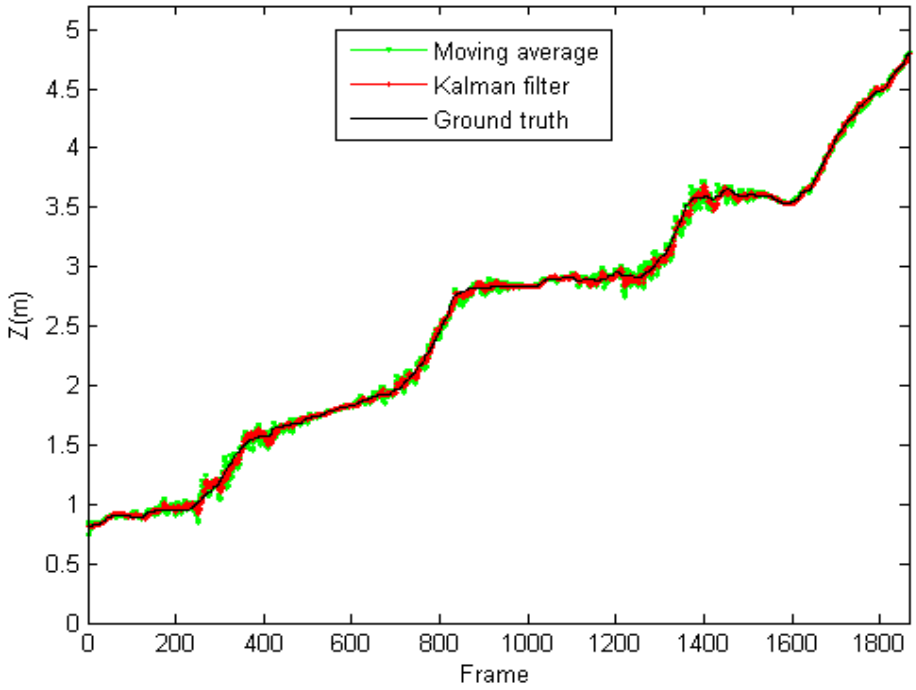
Figure 3.27 trajectories of the vehicles for tracking scenario 3 (Front-to-Back)

$\boldsymbol{x}$ error levels are more important since with KF the RMSE varies in the range $[0.0, 0.16]$m, and $[0.0, 0.4]$m for MovAve, Figure 3.26 (b).

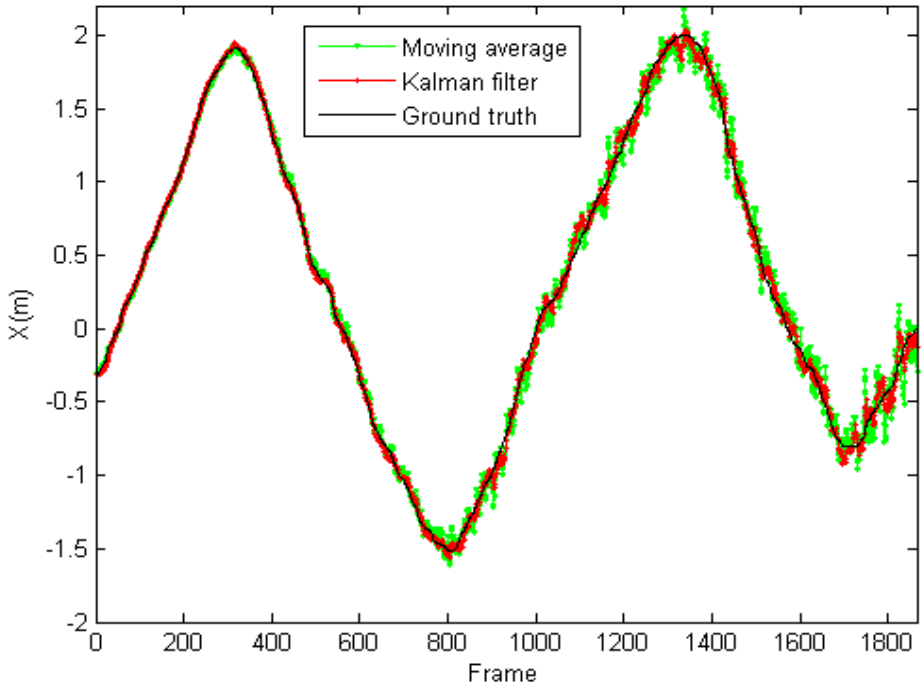
3.8.3.3 Scenario 3 (Front-to-Back)

This scenario has been carried out to test the capabilities of the filter in the middle of the scene towards a positive direction of the $\boldsymbol{z}$-axis (Front-to-Back).

From Figure 3.27, one can see that the overall accuracy of measurements is less noisy than the previous scenario, although, the trajectory still experiences substantial inaccuracies at the same weak spots (greater $\boldsymbol{z}$ and extreme $\boldsymbol{x}$). Conversely to Scenario 2, $\boldsymbol{z}$ is increasing continually in the interval $[0.8, 4.8]$ m Figure 3.28 (a). $\boldsymbol{x}$ component variations, however, are pseudo-periodic with a magnitude ~2m, Figure 3.28 (b).

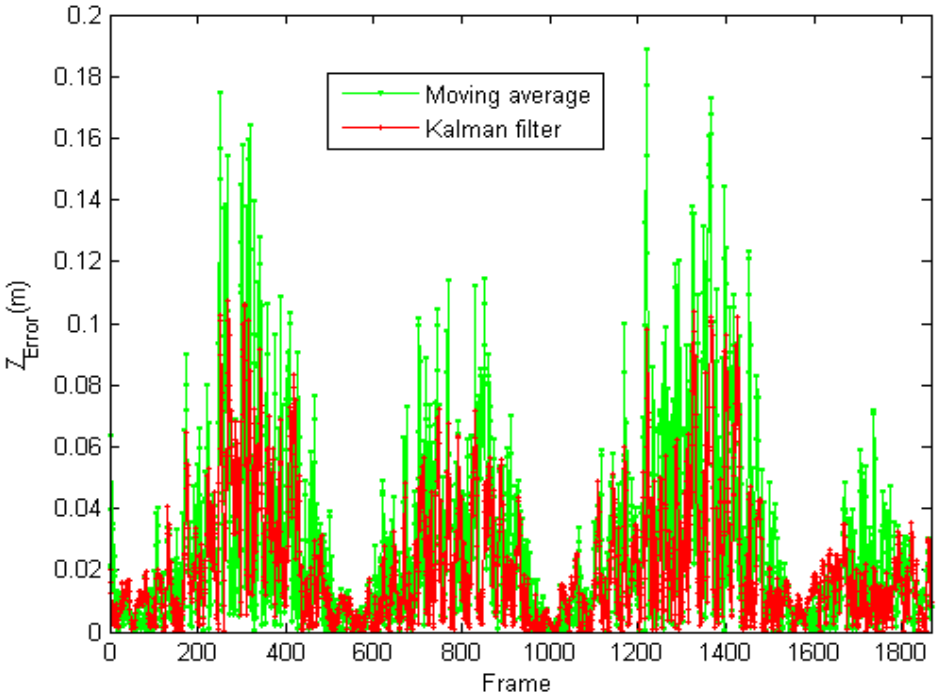


(a)

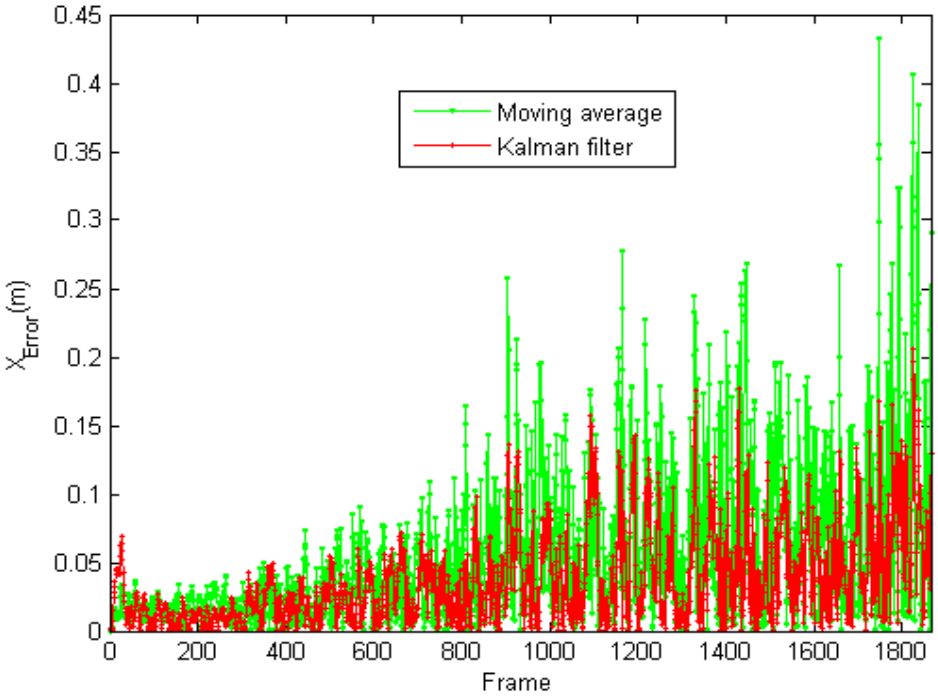


(b)

Figure 3.28 x and z variations. KF in red, MovAve in green and ground truth in black. **(a)** z. **(b)** x



(a)



(b)

Figure 3.29 Error graphs, KF in red, MovAve in green. **(a)** z. **(b)** x

The RMSE of z component ranges between 0.0m and 0.1m for KF. The latter reaches the highest levels at the three turning points corresponding to changes in the orientation of x, Figure 3.29 (a). z RMSE with MovAve becomes more important at the turning points in the same interval as in the previous scenario [0.0, 0.16]m, Figure 3.29 (a). The RMSE for x with KF progressively increases in the interval $[0.0, 0.15]$m and so does for MovAve in the interval $[0.0, 0.35]$m, Figure 3.29 (b).

3.8.4 Registration Applications

3.8.4.1 Off-Line Registration

The result delivered by the Kalman filter was tested within a 3D registration pipeline as presented in Figure 3.30. The process starts with the application of the filter to the 3D data streamed by the camera. In the module of feature extraction Figure 3.30, four key point extractors were investigated: TOMASI [50], SIFT3D [51], HARRIS3D [53] and THRIFT [59]. The latter fitted best to the need along with the CSHOT feature descriptor [84]. This descriptor leverages the colour information, available by default in Kinect data, for an even more distinctive matching of key points. Some results can be seen in Figure 3.31. Combining both THRIFT and CSHOT presents a twofold advantage because THRIFT describes very well the 3D geometry surrounding the features without considering colour information. Also, CSHOT builds the descriptors with both modalities, feature positions and its colour. Figure 3.31 (a) (b) (c) illustrates the intermediate results of the different steps regarding the off-line registration pipeline. The latter starts with the acquisition of the source and target point clouds to 3D key points extraction. Followed by correspondence matching and eventually the computation of the relative pose between two views. The resulting shiny surfaces are shown beneath the line of three images containing point clouds.

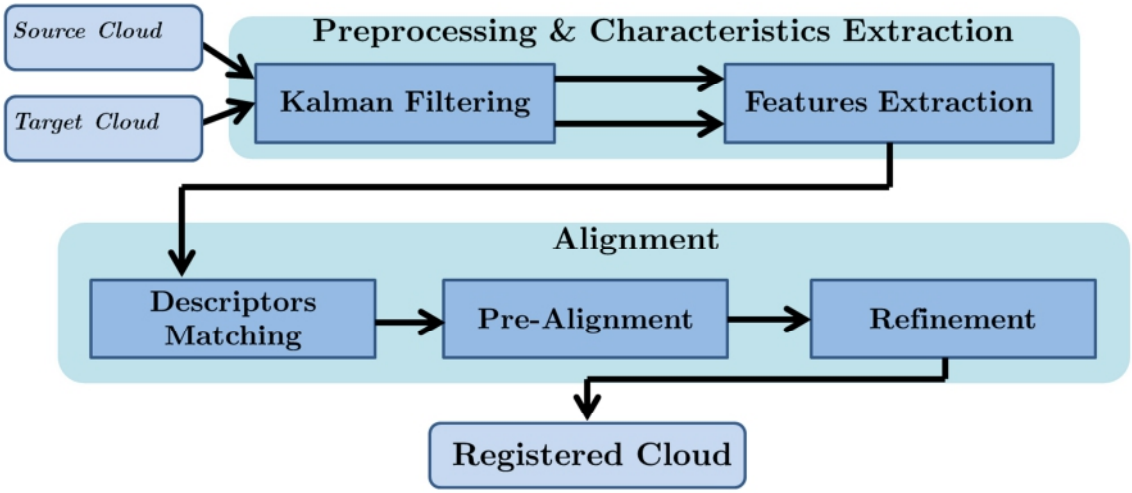
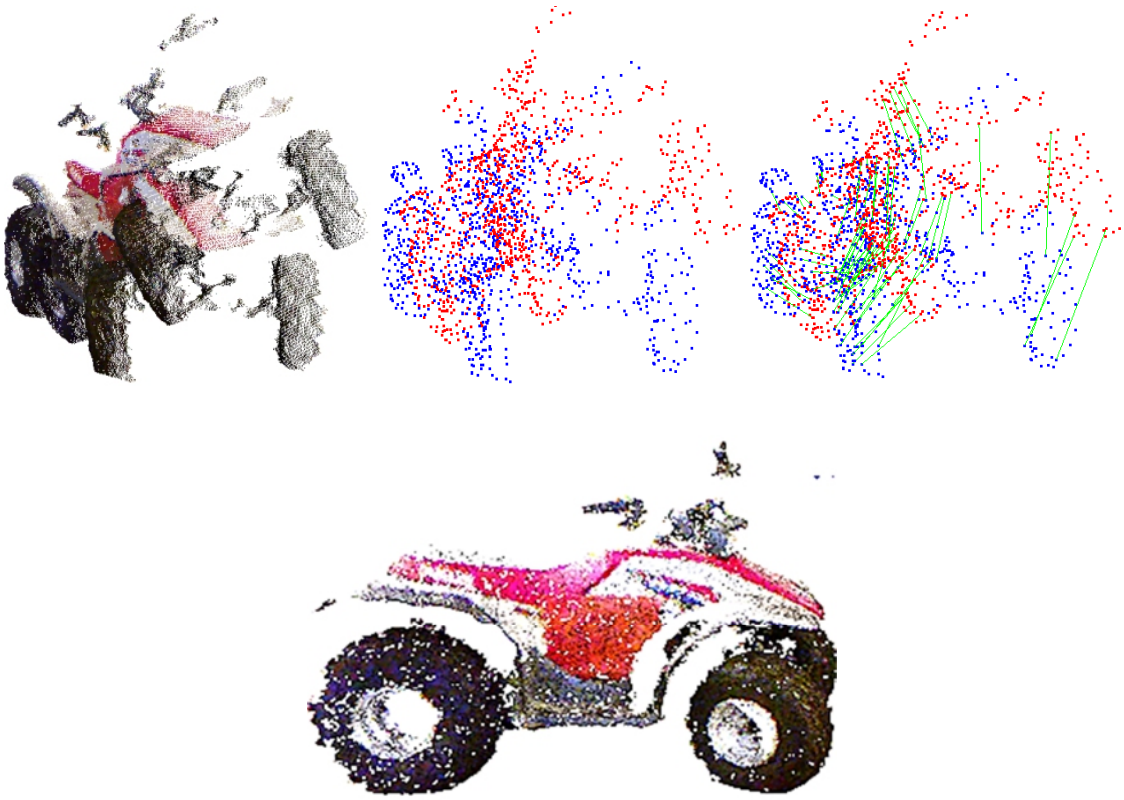


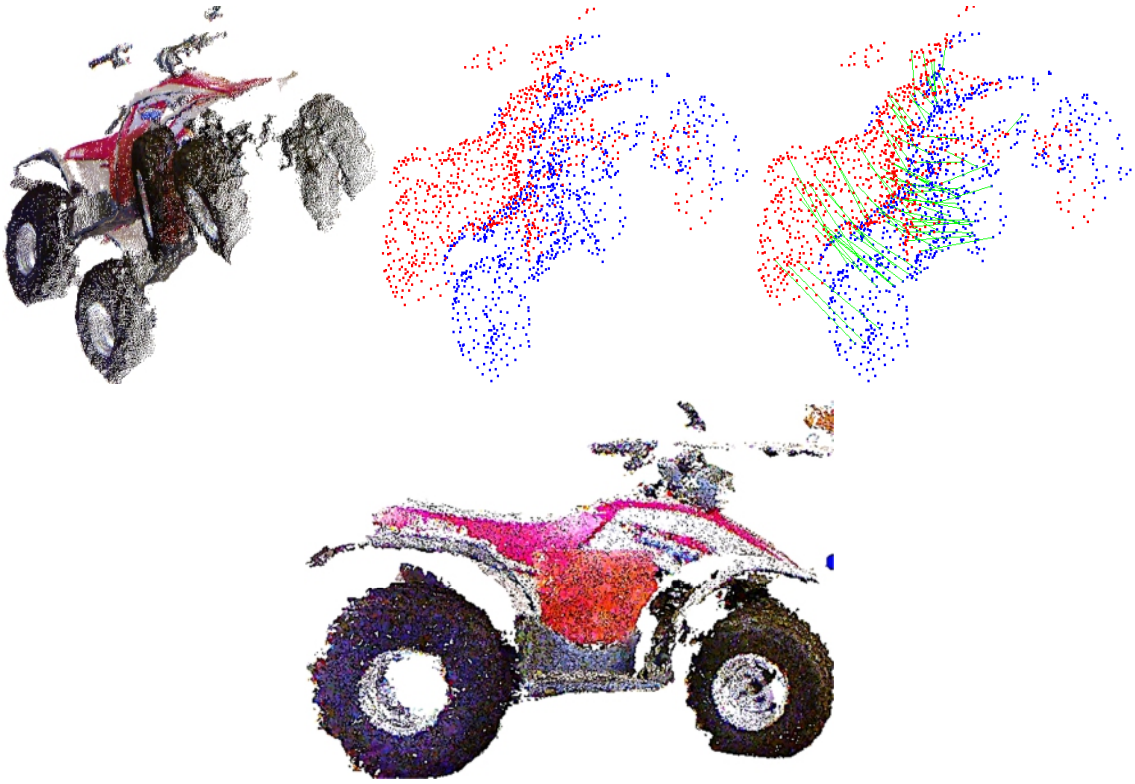
Figure 3.30 3D Registration pipeline

Although a good feature data was used (optimised with Kalman filter, THRIFT key points, and CSHOT descriptor), features-based registration does not always give the best result. It reduces computation time to over 1/80 the original time (the ratio between the size of key points' set and the whole point cloud). In most cases, the initial transformation gives a satisfactory result that hugely reduces the work expected from a further registration refinement stage. This is particularly the case, when there is no scale difference among the views.

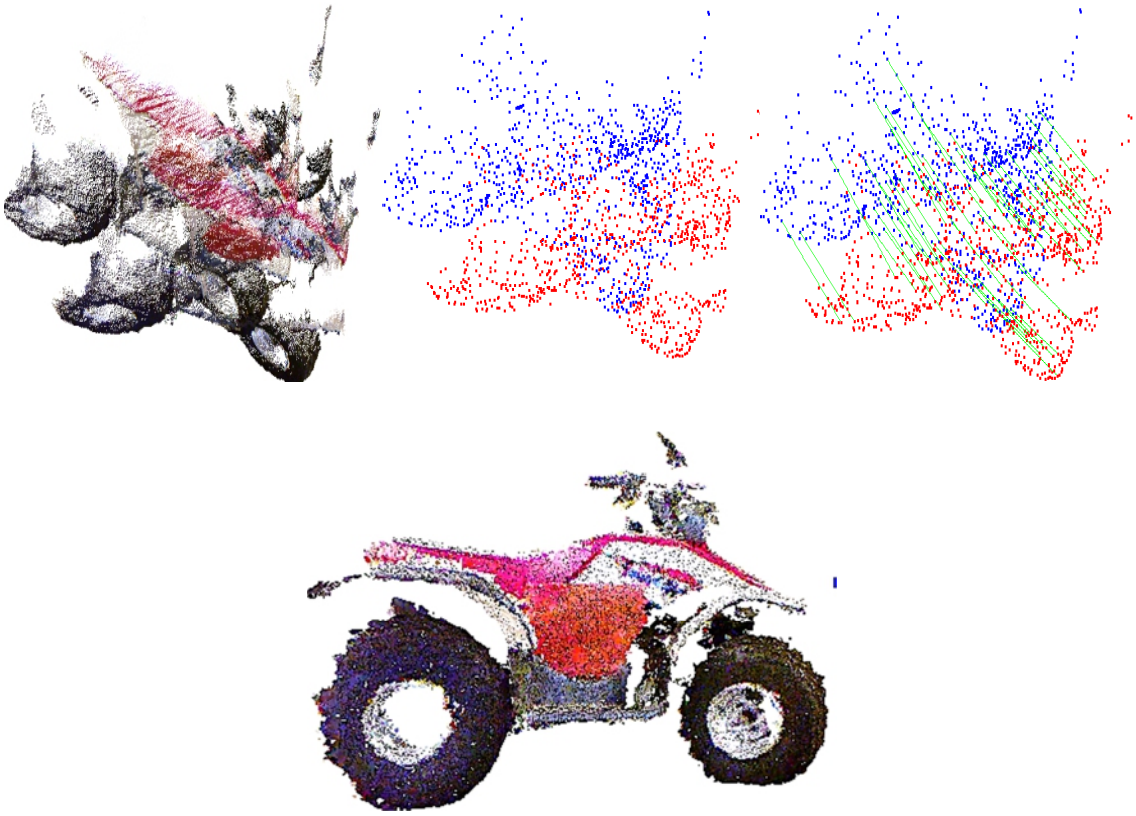
Nevertheless, a misalignment can appear when the features are correctly matched but the 3D geometry of the corresponding views is not similar; i.e. cases where the snapshots are captured at widely different scales, or when the objects in the scene are not totally rigid. For the latter (non-rigid objects), features on the surfaces do not conserve their relative world location and aspect. Rigid body transformations cannot handle this deformability, however. On the other hand, it is possible to remedy this limitation partially within the family of rigid body transformations. To this end, the misalignment can be overcome between feature points by taking into account not just the detected key points but the entire point cloud data. The corrective effect of this strategy can be seen in Figure 3.32. There may be a need to refine the registered point cloud for a smoother surface reconstruction.



(a)



(b)



(c)

Figure 3.31 Point cloud registration using THRIFT and CSHOT **(a)** view 1. **(b)** view 2. **(c)** view 3

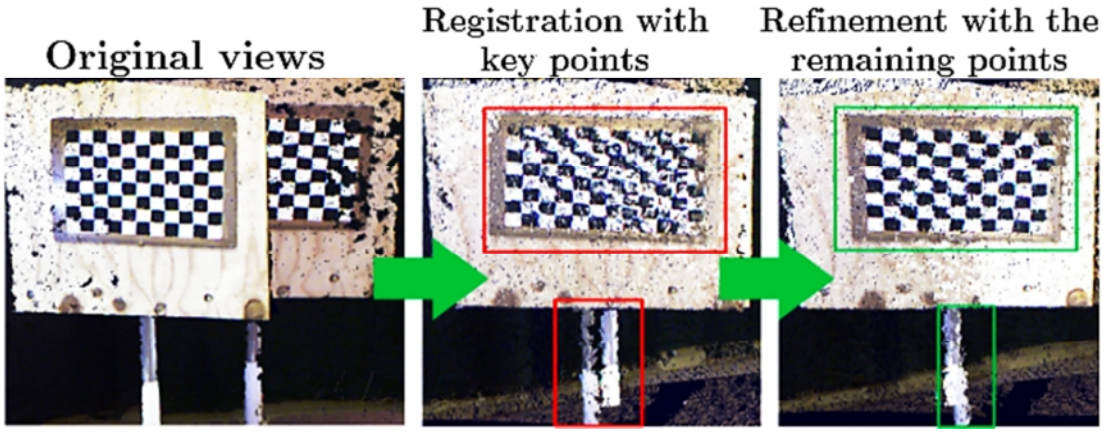


Figure 3.32 The effect of refinement

3.8.4.2 Real-Time Reconstruction

Unlike tracking, real-time registration applications use many features to find the correct mapping between the source and target views. In order to test the effectiveness of the filter for this type of application, some experiments on a real-time 3D scanning application with Kinect were carried out. In this experiment, a Structure From Motion algorithm was used to build gradually the 3D geometry of the scene as one moves around with a hand-held Kinect [85]. The algorithm reconstructs the 3D geometry of the site by aligning the freshly acquired frames on the already built model. Both the filtered and the raw 3D data were tested, see Figure 3.33.

The registration based on the raw data is prone to misalignments that in turn lead to rough 3D structure, particularly when the object is further away from the camera. As can be seen in Figure 3.33 column (b), raw depth points are lying in parallel planes (the discrete stripes can be clearly seen in Figure 3.33 column (b)). Such a structure demonstrates what has already been explained in Section 3.5. Feature positions are discretised and distributed on the available depth levels. Although surface reconstruction algorithm interpolates the gaps between the planes after triangulation, the resulting model still suffers from a rough and bumpy surface. This downside clearly appears after lighting the reconstructed structure. Nevertheless, for the model arising from the filtered data, illustrated in Figure 3.33 column (c), the geometry is smoother, and there is almost no misalignment between the views taken over time. The resulting 3D geometry is more realistic and less noisy. Hence, it does not need any further post-processing.

From a computational point of view, the 3D reconstruction algorithm runs at 20 FPS. This frame rate is less than the frequency of filtering that varies between 25 FPS to 30 FPS. In addition, one can visually evaluate the high quality of the outputs resulting from the scanning process. The filter successfully moves the discretised points back to their optimal 3D locations.

(a) (b) (c)

Figure 3.33 Experimental results for 3D on-line reconstruction applications. **(a)** RGB image. **(b)** 3D Scene reconstructed from raw depth data. **(c)** 3D Scene reconstructed from the filtered depth data

3.9 Conclusion

In this chapter, an innovative enhancement approach for the raw RGBD data issued by Kinect-like sensors was presented. The mathematical model representing the depth map was constructed and successfully adapted to a Kalman filtering scheme. The adaptation started from the demonstration that was conducted in order to satisfy the requirements of a Kalman filter in terms of linearity between the depth of points resting in parallel Z-levels and the Gaussian nature of their inherent noise.

Firstly, the filtering approach was tested on an object tracking algorithm to assess the accuracy of data after applying the filter. To this end, three fixed viewpoint tracking scenarios were carried out with one Kinect in front of a moving robot. The different scenarios were chosen to assess all the aspects of tracking based on RGBD sensors. Such a family of sensors shares the same deficiency of accuracy on the boundaries of the imager because of the unavoidable residual lens distortion, even after an accurate calibration. In addition to a substantial drop in accuracy at greater depth levels. The results have proved the effectiveness of the filter with the appropriate parameters that have been determined. The effective operational range of the camera was also extended from 4.0m for the raw output alone to 5.5m with the filtered outputs.

Secondly, the output of the filter in a 3D scanning application was tested to assess visually its effect on 3D model reconstruction. A complete pipeline for off-line RGBD data registration was proposed starting from feature extraction to descriptor computation to correspondence extraction and matching and finally the actual alignment followed by the refinement. The result expected from a feature-based pre-alignment registration is highly reliable and takes less time. The best results were reached with THRIFT key points in addition to CSHOT as the most efficient descriptor.

Thirdly, a possible architecture to integrate the filter directly in the existing driver of the camera was proposed. However, in order to maintain the real-time

nature of the sensor, an algorithm that applies in parallel a single kernel of processing launched on all the pixels of the depth image was used. Thus, the solution was implemented in the GPU to boost the frame rate. Practically, the filter was designed pixel-wise because each pixel is independent of its neighbours.

This solution could be easily extended to other 3D scanning devices. Such a facility would provide the users with great potential to reach a better accuracy without causing any latency to the system. Consumer cameras are now better endowed to achieve reasonably what could not be otherwise accomplished without expensive, sophisticated laser scanners and tracking frameworks.

4 RGBD Data Correction and Background Removal

In this chapter, two pre-processing algorithms aiming at the correction of depth measurements delivered by ageing RGBD cameras and background subtraction are presented.

The first contribution (ageing sensors correction) is a novel method to calibrate active depth cameras accurately. This approach can either be based on simple interpolation means or the more sophisticated Gaussian Process Regression (GPR). It is applied after the standard calibration, and it is particularly useful for worn depth cameras. The latter were proven to present a significant decay of accuracy that cannot be fixed with the standard pinhole mono or stereo calibration procedures.

The second contribution is another algorithm for background/foreground segmentation of RGBD data with the Gaussian Mixture Models (GMM). The algorithm begins with background subtraction from the colour and the depth

images separately. The foreground regions resulting from both streams are then fused for a more robust detection.

The experimental data, obtained when both algorithms are applied, show the weaknesses of standard calibration and the corrective asset of the proposed solution. They also demonstrate the robustness of the proposed segmentation approach in coping with illumination change, shadow and reflection. These findings can be further extended to fit any type of range cameras that have a similar working principle as the Kinect.

The two algorithms have been implemented in the GPU. Thus, they are suitable for real-time systems because they exploit the entire frame rate offered by the sensor (30 FPS).

4.1 Overview

4.1.1 Depth Sensors Correction

Despite widespread success achieved by RGBD cameras, their accuracy issues have not been fully addressed. Prior studies in the literature have focused on either regular monocular camera calibration (colour and infrared cameras independently) or stereo calibration, where both RGB and IR cameras are jointly characterised [46]. During experimentation with Kinects, a continually decaying quality of range measurements is noticed over time. This phenomenon persists even after carrying out a correct standard calibration. More importantly, this issue of accuracy regarding Kinect was very little discussed in the literature.

The working principle of the correction module is partly based on the findings presented in the previous chapter. In addition, a simple regression as well as a learning approach adapting GPR are applied to correct the drift of depth readings. GPR has been chosen because it provides a probabilistic framework to work directly with priors on a space of functions. It also provides a more accurate prediction and correction of the outputs. On the other hand, an

important limiting factor for the acceptance of this method in practice is the computational burden observed when training datasets grow. However, the training phase can be carried out offline, just as the regular calibration is performed. Generally, the algorithm requires a set of less than 1% of the entire working dataset to accomplish the learning phase correctly. The correction step applies only to the $\boldsymbol{z}$ coordinate (depth) of the query data. The remaining $\boldsymbol{x}$ and $\boldsymbol{y}$ components are deduced from the corrected $\boldsymbol{z}$ and camera calibration parameters.

Few works in the literature have yet discussed the issue of accuracy when depth cameras are used as a measuring tool. Zhu et al. [86] combined a Time Of Flight (TOF) camera with a colour stereo setup to correct the distribution of depth data. Chiu et al. [87] combined the depth images captured from a range sensor and the disparity map delivered by an IR/RGB pair to improve the accuracy of the 3D map. Xul et al. [52] used the same technique to improve the accuracy of the depth image covering the scene. Henry et al. [88] combined the visual features of an RGB camera and the shape-based alignment of a range sensor to reconstruct a reliable 3D geometry. Moreover, Matyunin et al. [89] used a temporal filtering of RGBD images to cope with occlusions and to increase the temporal stability of outputs.

In all of the above-cited works, the authors either compensate for one sensor's accuracy with alternative sensors' data or they filter the raw data streamed by the sensor itself. Nevertheless, the solution proposed in this thesis corrects the active depth cameras by leveraging the characteristic properties of structured light sensors, already presented in Chapter 3, and a learning algorithm (GRP) to readjust the measurements. The results are cleaner (fewer outliers), more stable (fewer abrupt fluctuations) and accurate sensory data.

4.1.2 RGBD Background Removal Methods

Moving objects tracked with RGBD cameras must first be detected in the image. This detection requires a tool able to extract foreground pixels

corresponding to the regions of interest from the remaining background. Hence, emerges the need to decide whether a newly acquired frame contains any foreground regions corresponding to the tracked objects and, if there are any, where they are located.

Several problems must, therefore, be tackled by a good background removal algorithm. Such an algorithm should be robust against non-stationary backgrounds such as waving trees, sudden illumination changes, shadows and camouflage. Most of the current solutions in the literature are able to cope decently with all the disturbances listed above. In addition, robustness and processing time limitations are also a critical requirement for real-time applications.

Many works in the literature have already addressed the possibility of using depth and colour images jointly for a better background segmentation. Cristani et al. [90] presented an overview of background subtraction solutions for mono as well as stereo cameras. Abramoff et al. [91] used a stereo pair of cameras for an automatic segmentation algorithm. Gordon et al. [92] added disparity information to the GMM for background modelling. In their approach, the authors found that the combination stereo/colour helps enormously overcoming the classic problems of colour segmentation.

Nevertheless, stereo data itself originates from pairs of RGB images. For this reason, it holds the same weaknesses towards change in illumination and shadows. Friedman and Russell proposed a GMM approach for background removal in [93]. A few years later, several innovations were added to the original model by Stauffer and Grimson [94]. Their paper is often regarded as a reference for GMM-based background/foreground segmentation. Lee et al. [95] later proposed a GPU implementation of Stauffer and Grimson's algorithm that has the same performance as the native implementation but it is much quicker.

Both algorithms of the present chapter, i.e. GPR correction and GMM foreground/background segmentation, will serve as an RGBD data pre-

processing module for the next chapter. Similarly, they can also be used by any other application to take full advantage of the potential of cheap depth sensors.

The remainder of this chapter is organised as follows: In Section 4.2, a depth sensor correction solution is provided. The problem of decay in accuracy is initially uncovered then two solutions are offered to solve it. In Section 4.3, another cooperative background removal method, using both RGB and Depth images, is detailed. In Section 4.4, test results are presented and discussed along with an analysis of computation time.

4.2 Depth Sensors Correction

4.2.1 Filtering Unreliable z_i

Here, filtering means eliminating the unreliable measurements. Kinect's driver computes the depth map from the raw disparity data delivered by the device. The IR camera/projector setup constitutes a stereo pair with a baseline of approximately 7.5cm (Chapter 2). The projector emits a beam of known IR patterns onto the scene, which are generated by a set of diffraction gratings with special care in order to reduce the effect of zero-order propagation at the central bright dot [96]. The actual depth is computed by a triangulation process that correlates each measured value to a reference disparity stored in the device. In other words, for each pixel in the IR image, a small correlation window is used to compare the local pattern at that pixel with the reference one (Chapter 2). This correlation yields an offset from the known depth value, which is the actual disparity measurement.

Kinect performs a further interpolation to reach sub-pixel accuracy. The camera computes the disparity according to the following equation:

$$d = \frac{1}{8}(d_{off} - k_d) \tag{4.1}$$

Where $\boldsymbol{d}$ is the normalised disparity, $\boldsymbol{k_d}$ is Kinect's disparity, and $\boldsymbol{d_{off}}$ is an offset value dedicated to the device. The ratio $\mathbf{1/8}$ appears because the value of $\boldsymbol{k_d}$ is expressed in $\mathbf{1/8}$ pixel unit[1].

Before proceeding with the correction of depth readings, the unreliable $\boldsymbol{z_i}$ are eliminated from the captured map. A depth value is considered as unreliable when no disparity information is available. Its corresponding location in the scene may be either out of reach regarding the sensor; or it may have a shiny or light-emitting surface. In addition, when the object is too close to the camera $\boldsymbol{z_i < 0.8\ m}$, the sensor delivers imprecise measurements, which are useless in their coarse state.

4.2.2 Kinect Depth Map Structure

In order to study the nature of sensory outputs, the camera was pointed parallel to a large flat wall [7]. This experiment results in a cloud of points from the entire operational range, see Figure 4.1 (a).

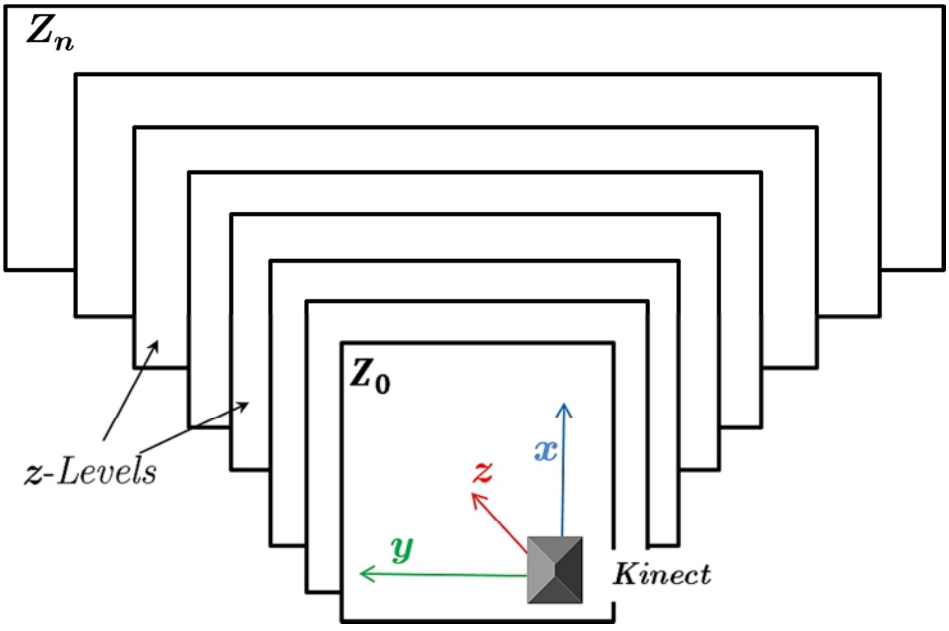
The 3D distribution of depth levels is illustrated in Figure 4.1 (b). Point data within the captured cloud lie in the independent parallel planes $\boldsymbol{Z_0, \dots, Z_n}$. Each of which, has its own depth value $\boldsymbol{Z_k}$, and set of points $\boldsymbol{P_i}$ $(\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{Z_k})$. The latter share the same range reading $\boldsymbol{Z_k}$. The number of Z-Levels determines the precision of depth measurement. In other words, the density of Z-Levels is proportional to the depth resolution of the sensor.

The discrete nature of captured data originates from the quantisation of the actual continuous distance separating the objects from the camera plane. In addition, they are limited in number ($\boldsymbol{n + 1 \approx 730}$ distinct depth levels). More importantly, the finite set of possible levels is similar for all Kinects running the same driver. The gap between two successive Z-Levels ($\boldsymbol{Z_{k+1} - Z_k}$) increases

[1] http://wiki.ros.org/kinect_calibration/technical. 2015

proportionally to the square of the distance separating the camera from the scene [46].

(a)



(b)

Figure 4.1 Kinect depth data. **(a)** Actual output of the wall. **(b)** 3D structure of the *Z*-Levels

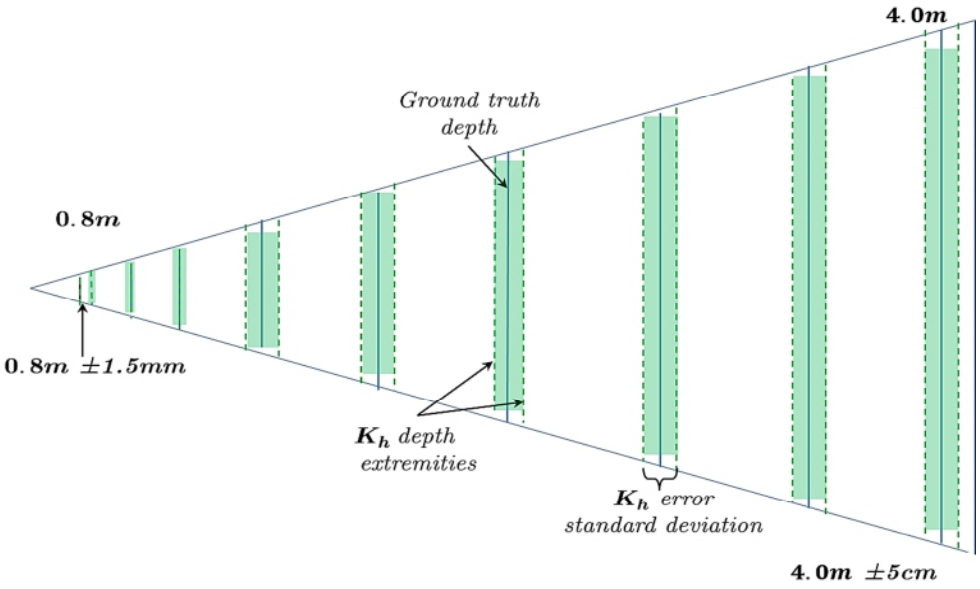
4.2.3 Problem Statement

After being used for an extended period, some electronic sensory device suffers from a decreasing accuracy. The quality of the 3D map generated by Kinect sensor is, therefore, highly affected by the performance of its IR setup. However, the RGB camera, passive part of the device, is more robust, since the colour image is less affected over time.

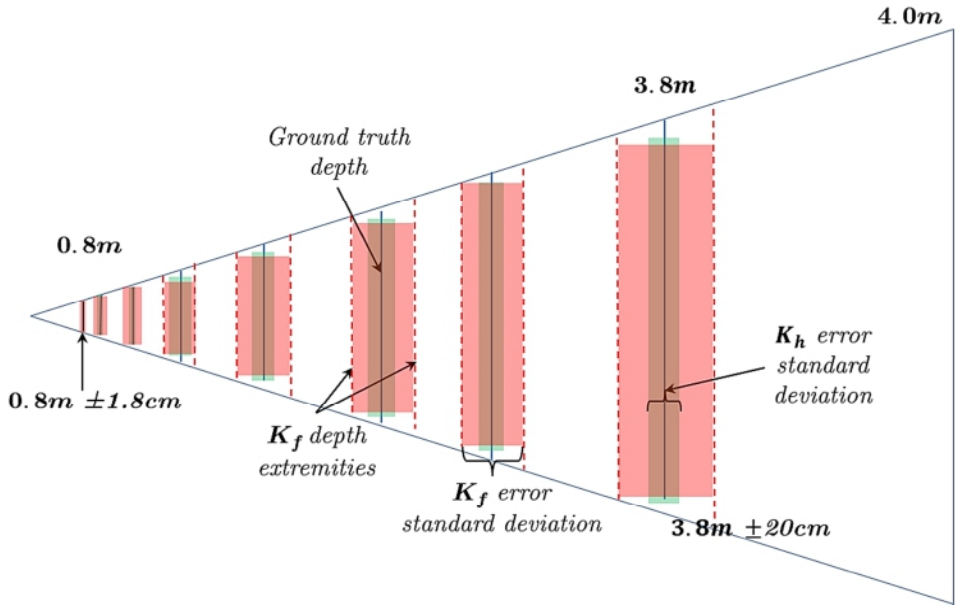
In all the following sections, the Kinects are assumed to have undergone a proper standard monocular calibration ($\boldsymbol{f_x}$, $\boldsymbol{f_y}$;$\boldsymbol{c_x}$, $\boldsymbol{c_y}$) for both cameras. In addition, a stereo calibration determines the parameters $[\boldsymbol{R}, \boldsymbol{T}]$.

Let $\boldsymbol{K_h}$ be a healthyKinect. Its resolution is the same as the native one characterising factory operational range regarding the sensor (±1.5mm at 80cm, ±5.0cm at 4.0m). On the other hand, $\boldsymbol{K_f}$ is a worn Kinect, for which the accuracy (± 1.8mm at 80cm, ± 20.0cm at 3.8m) cannot be recovered with standard calibration procedures. Thus, the problem becomes: given the trustworthy depth readings delivered by $\boldsymbol{K_h}$, how could one correct the shifted disparity image generated by $\boldsymbol{K_f}$?

The phenomenon of deterioration in accuracy is explained in Figure 4.2. $\boldsymbol{K_h}$ covers the entire operational range of the camera (0.8m to 4.0m) at a good accuracy, Figure 4.2 (a). The error in the depth estimation is shown with green intervals. On the other hand, $\boldsymbol{K_f}$'s range is smaller (0.8m to 3.8m) and the uncertainty in its depth measurements is larger compared to $\boldsymbol{K_h}$ (red intervals in Figure 4.2 (b)).



(a)



(b)

Figure 4.2 Error intervals in Z-Levels. **(a)** K_h depth map structure. **(b)** K_f depth map structure

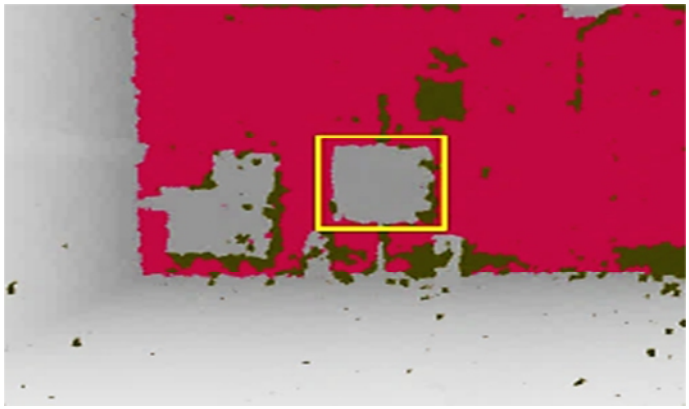
This decreasing accuracy is remarkably observed within ageing sensors. To the author's knowledge, the state of the art calibration and studies on depth cameras have not yet addressed either this issue or a solution for it.

Before this work was raised, many attempts were undertaken to correct the shift with optical calibration procedures. However, the erroneous range readings persisted. In the technical specifications of the sensor, it is mentioned that the computation of the disparity map is based on a triangulation algorithm applied to the IR projector/camera pair. The projector is an output device, hence, it is not capable of capturing any calibration patterns. In addition, the factory calibration parameters concerning the IR setup are embedded in the sensor itself and inaccessible from outside.

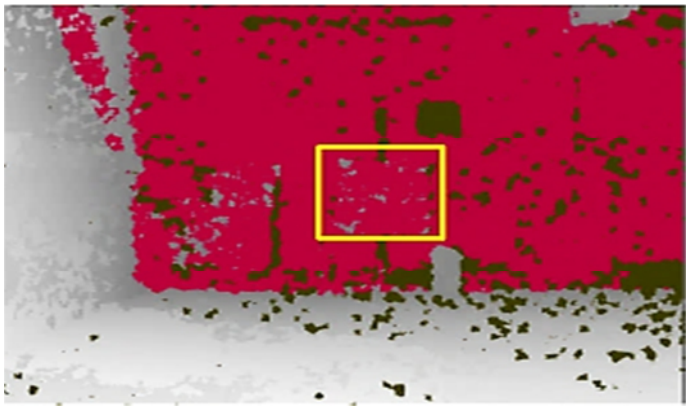
To explain the problem with a real scenario, a flat wooden panel was fixed in a fronto-parallel direction to the two Kinects at a distance of 3.90m Figure 4.3 (a). The upper camera ($\boldsymbol{K_h}$) is working properly, but the other one ($\boldsymbol{K_f}$) is worn. In this experiment, the range separating the panel from both sensors is measured. The disappearance of the grey square (panel) from the image is a sign of its non-detection in the native operational range. At 3.90m the depth map captured by the healthiest camera localises the panel at 3.89m, Figure 4.3 (b). However, the worn camera is unable to see it as shown in Figure 4.3 (c). Afterwards, the panel was moved forward to 3.70m, so both Kinects could detect it, Figure 4.4 (a). $\boldsymbol{K_h}$ indicates that the pattern is seen at 3.70m, Figure 4.4 (b). Whereas, $\boldsymbol{K_f}$ localises it at 3.95m, Figure 4.4 (c).

In order to fix this issue, a learning algorithm was proposed in order to leverage the precise outputs of the healthy sensor for the elimination of shift in the worn one.

(a)



(b)

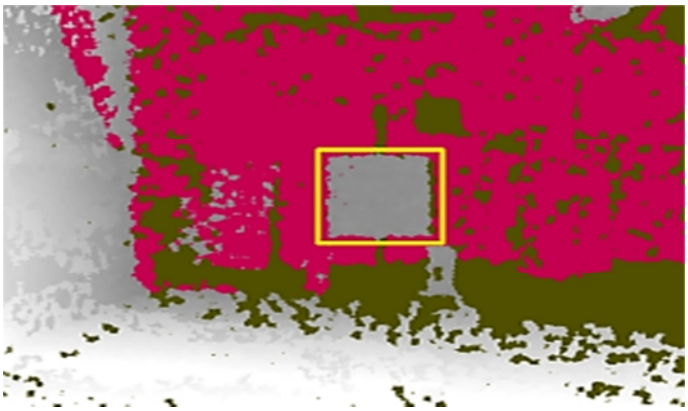


(c)

Figure 4.3 Accuracy difference between $\boldsymbol{K_h}$ and $\boldsymbol{K_f}$ at 3.9m. **(a)** Setup (3.90m). **(b)** $\boldsymbol{K_h}$ output (3.89m). **(c)** $\boldsymbol{K_f}$ output (unseen)

(a)

(b)



(c)

Figure 4.4 Accuracy difference between K_h and K_f at 3.7m. **(a)** Setup (3.70m). **(b)** K_h output (3.70m). **(c)** K_f output (3.95m)

4.2.4 Depth Correction with Interpolation

To remedy the erroneous measurements, a mathematical model is first built from the observed data and then attributed to the respective sensor for correction. The design of such a model is based on the determination of the appropriate function that readjusts the inherent shifted raw measurements. This function should be able to remap each shifted depth value (z_i) to its respective real world position. Hence, the depth metrics streamed by the sensor and their corresponding ground truth ones, i.e. pairs ($z_i, \tilde{z}_i$), are related with the equation $f(z_i) = z_i - \tilde{z}_i$ (shift from the true value), Figure 4.5.

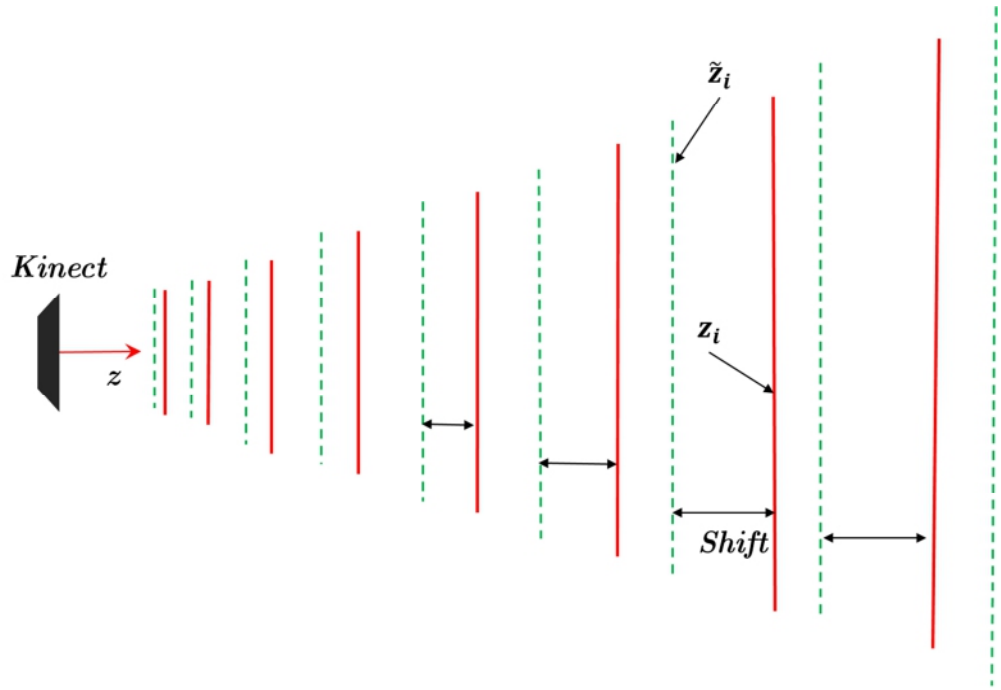


Figure 4.5 Shift in depth values of the worn sensor

$f(z_i)$ computes for each depth value the corresponding error based on a precise ground truth reference. This function results from the interpolation of the points taken simultaneously from the Kinect's point cloud and a high precision tracking system[2]. Alternatively, a trustworthy sensor, capable of delivering good quality measurements, can be used as well. The sparse data ($z_i, \tilde{z}_i$) is fitted

[2] https://www.naturalpoint.com/optitrack/. 2015

with a polynomial function (f) that analytically approaches the distribution of samples. This function takes as input the raw worn measurements (z_i) and outputs correct correspondents ($\tilde{z}_i$).

Although the correction helps significantly to overcome the shift in the depth data, the drop in the resolution of the sensor could not be entirely resolved. In other words, the correction algorithm improves the accuracy of the device, but it cannot surpass the native resolution due to the innate hardware limitations.

Practically, such an algorithm is more suitable to run in the GPU simultaneously with the image acquisition process. The measurements are therefore corrected without incurring an extra computational load to the remaining layers of processing.

4.2.5 GPR on Kinect Depth Data

The Gaussian Process Regression is a generic supervised learning method initially designed to solve regression problems [97]. This method has the advantages of being: *predictive*, it interpolates all the available training data; *probabilistic*, one can compute empirical confidence intervals that may be used to refit the prediction in some regions of interest and *versatile*, because different linear regression models can be specified [98].

In this thesis, the proposed correction procedure is underlined by the GPR in order to fit the finite set of sampled points. The latter (priors) are considered as training data based on which the GPR computes the posterior distribution. The more likely functions are, therefore, those passing through training points. As a result, the GPR provides a continuous predictive distribution whose mean is the estimation of the underlying model and variance is the confidence in measurement (see Figure 4.6).

The model is non-parametric and fully specified via mean and covariance functions. The latter often have hyper-parameters that would be optimised to fit the model to a given training dataset. Given a set of n training pairs $\{(z_i, \tilde{z}_i)\}$,

where z_i denotes the erroneous depth measurements and $\tilde{z}_i$ are their respective ground truth counterparts. The purpose is to learn a model $f(z_i)$, which is able to attribute to every shifted z_* a single accurate correspondent $f(z_*)$. This function is therefore able to accurately express the actual range that should be seen from the query pixel instead of z_*. The process is modelled by the following function:

$$\tilde{z}_i = f(z_i) + \ e_i \tag{4.2}$$

Where e_i is a random variable that has a Gaussian distribution of mean zero and variance σ_n^2.

The fundamental assumption in GPR modelling is that the data is sampled from a multivariate Gaussian distribution [97]. The observed pairs of

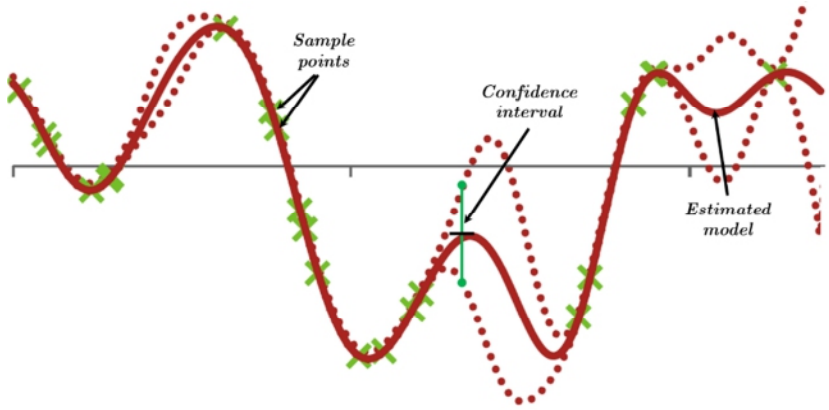


Figure 4.6 GPR components

$(drifty, correct)$ depth values can also be described as a Gaussian distribution $\tilde{z} \sim \mathcal{N}(\mathbf{0}, K(Z,Z) + \sigma_n^2 I)$ where $Z = \{z_i\}$, and $K(Z,Z)$ is the covariance matrix computed using a known covariance function $k(z_p, z_q)$:

$$K(Z,Z) = \begin{bmatrix} k(z_1,z_1) & k(z_1,z_2) & \dots & k(z_1,z_n) \\ k(z_2,z_1) & k(z_2,z_2) & \dots & k(z_2,z_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(z_n,z_1) & k(z_n,z_2) & \dots & k(z_n,z_n) \end{bmatrix} + \boldsymbol{\sigma_n^2} I \tag{4.3}$$

The diagonal elements of $\boldsymbol{K}$ are equal to $\boldsymbol{\sigma_n^2 + \sigma_s^2}$. Where, $\boldsymbol{\sigma_s^2}$ is the maximum allowable variance between two input variables. The extreme off-diagonal elements tend to zero when a large domain is spanned.

The Gaussian and Squared Exponential are the most commonly used covariance functions [4]. In this work, Squared Exponential has been chosen as a covariance function:

$$\begin{aligned} cov(f(z_p), f(z_q)) &= k(z_p, z_q) \\ &= \sigma_s^2 e^{-\frac{1}{2l^2}((z_p - z_q)^T W (z_p - z_q))} \end{aligned} \tag{4.4}$$

The covariance between the outputs $\boldsymbol{f(z_p), f(z_q)}$ (corrected depth values) is described as a function of the inputs $\boldsymbol{z_p, z_q}$(drifty depth values). It reaches its maximum value $\boldsymbol{k(z_p, z_q) = \sigma_s^2}$ when the two variables become very close to each other $\boldsymbol{z_p \approx z_q \xrightarrow{yields} e^{-\frac{1}{2l^2}((z_p - z_q)^T (z_p - z_q))} \approx 1}$. This means that the two outputs are nearly perfectly correlated. On the other hand, when the two variables are far away from each other, $\boldsymbol{k(z_p, z_q) \approx 0}$. This means that the two points $\boldsymbol{z_p, z_q}$ are very weakly correlated, and so are the outputs $\boldsymbol{f(z_p), f(z_q)}$. The relationship between the covariance function and the distance between two elements causes distant observations to carry a negligible effect during the interpolation at a new point. The length parameter $\boldsymbol{l}$ defines how much effect this separation has. The latter controls the flexibility built into Equation (4.4).

The joint distribution of observed $\boldsymbol{\tilde{z} = [\tilde{z}_1 \dots \tilde{z}_n]}$ and the predicted values $\boldsymbol{f(z_*)}$ for a query point $\boldsymbol{z_*}$ is given by:

$$\begin{bmatrix} \tilde{z} \\ f(z_*) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}\right) \tag{4.5}$$

With:

$$K_* = [k(z_*, z_1) \quad k(z_*, z_2) \quad ... \quad k(z_*, z_n)]$$

$$K_{**} = [\ k(z_*, z_*)\ ]$$

The posterior distribution $\boldsymbol{p}(\boldsymbol{f}(\boldsymbol{z}_*)|\tilde{\boldsymbol{z}})$, i.e. given the data $\tilde{\boldsymbol{z}}_{\boldsymbol{i}}$, how likely is the prediction $\boldsymbol{f}(\boldsymbol{z}_*)$, is given by:

$$p(f(z_*)|\tilde{z}) \sim \mathcal{N}(K_* K^{-1} \tilde{z}, K_{**} - K_* K^{-1} K_*^T) \tag{4.6}$$

The best estimation of $\boldsymbol{f}(\boldsymbol{z}_*)$ is the mean value of the distribution in Equation (4.6):

$$f(z_*) = \ K_* K^{-1} \tilde{z} \tag{4.7}$$

The uncertainty in the estimation is the covariance of the distribution given by:

$$Var(z_*) = \ K_{**} - K_* K^{-1} K_*^T \tag{4.8}$$

If the Gaussian kernel is used, the hyper-parameter $\boldsymbol{\theta}$ of the Gaussian is given by $\boldsymbol{\theta} = [\boldsymbol{\sigma_n^2}, \boldsymbol{\sigma_s^2}, \boldsymbol{W}]$, where $\boldsymbol{W}$ is the width of the kernel [97]. These parameters are the only free parameters. Their optimal values for a particular dataset can be automatically estimated by maximising the **log** marginal likelihood with standard optimisation methods.

The purpose of adapting the GPR in this work is the learning of a function $\boldsymbol{f}(\boldsymbol{z}_*)$ from the training data $\{(\boldsymbol{z_i}, \tilde{\boldsymbol{z}}_{\boldsymbol{i}})\}$. These pairs of $(observed, ground\ truth)$ depth values will serve to correct the query points $\boldsymbol{P}_*(\boldsymbol{x}_*, \boldsymbol{y}_*, \boldsymbol{z}_*)$. Nevertheless, the correction applies only on the depth component $\boldsymbol{z}_*$. Thereafter, the computation of the remaining two coordinates $(\hat{\boldsymbol{x}}_*, \hat{\boldsymbol{y}}_*)$ is based on the corrected depth $(\boldsymbol{f}(\boldsymbol{z}_*) = \hat{\boldsymbol{z}}_*)$, and the native calibration parameters of the IR camera $(\boldsymbol{f_{x\_ir}}, \boldsymbol{f_{y\_ir}}; \boldsymbol{c_{x\_ir}}, \boldsymbol{c_{y\_ir}})$. For instance, the depth measurements are projected in the 3D frame of the IR camera as follows:

$$\begin{cases} \widehat{x}_* = (\hat{z}_*/f_{x\_ir})(u_* - c_{x\_ir}) \\ \widehat{y}_* = (\hat{z}_*/f_{y\_ir})(v_* - c_{y\_ir}) \end{cases} \tag{4.9}$$

$(\boldsymbol{u}_*, \boldsymbol{v}_*)$ are the 2D image coordinates of the target pixel where the depth value $\boldsymbol{z}_*$ has been captured.

4.2.6 Depth Map Correction Procedure

The complete correction procedure of the worn sensor $\boldsymbol{K_f}$ is illustrated in Figure 4.7. This procedure starts with a *training* phase, Figure 4.7 (a); followed by *correction* (test) stage, Figure 4.7 (b). The pairs $\left\{(\boldsymbol{P_i}, \widetilde{\boldsymbol{P}}_{\boldsymbol{i}})\right\}$, i.e. $(faulty, correct)$ corresponding points, are utilised in the construction of the training set $\{(\boldsymbol{z_i}, \tilde{\boldsymbol{z}}_{\boldsymbol{i}})\}$. These points are selected from a collection of feature points sampled from the whole operational range of the camera. At this level, both cameras ($\boldsymbol{K_f}$, $\boldsymbol{K_h}$) are supposed to have accurately undergone a mono, IR and RGB calibration separately, and a stereo calibration IR/RGB.

More importantly, in this experiment the healthy camera $\boldsymbol{K_h}$ was tested against the accurate range measurements delivered by a rangefinder[3]. The wooden panel facing the camera (see Figure 4.3 (a)) is set to a position orthogonal to $\boldsymbol{z}$-axis. Both, camera and rangefinder, are located at the origin $\boldsymbol{z} = \boldsymbol{0}$.

[3] Bosch Laser Range Finder DLE40

The panel is moved over several distances from the sensor. Each distance is attributed a marker. The collection of the pairs worn/correct range values is achieved jointly between the sensor and the rangefinder. For every known distance, a human agent fires the laser beam at the panel and reads the distance. Simultaneously, the cameras capture a 3D snapshot of the scene. The imaged snapshots are saved together with their respective correct readings obtained from the rangefinder. Afterwards, the extraction of planar surfaces within the scene is performed. For instance, the panel is easily recognisable among alternative shapes. Later, the average value of z component regarding the points resting in the panel is computed. All these points are normally at the same range from the sensor.

Despite the tediousness of this data collection process (because the human agent needs to fire the laser beam and read the distance for every sample), the ground truth range readings represent the actual depth value very accurately. However, if the user does not have a rangefinder, an easier alternative can be used as well. The latter requires a healthy camera to be associated with the worn one. The rigid body transformation linking both cameras can be obtained from the alignment of the set of features extracted from the respective colour images. The procedure of correction is described below:

- Set the two cameras at a close scale (small scale difference between point clouds). At this stage, the worn/healthy rigid-body transformation between cameras is assumed to be known.

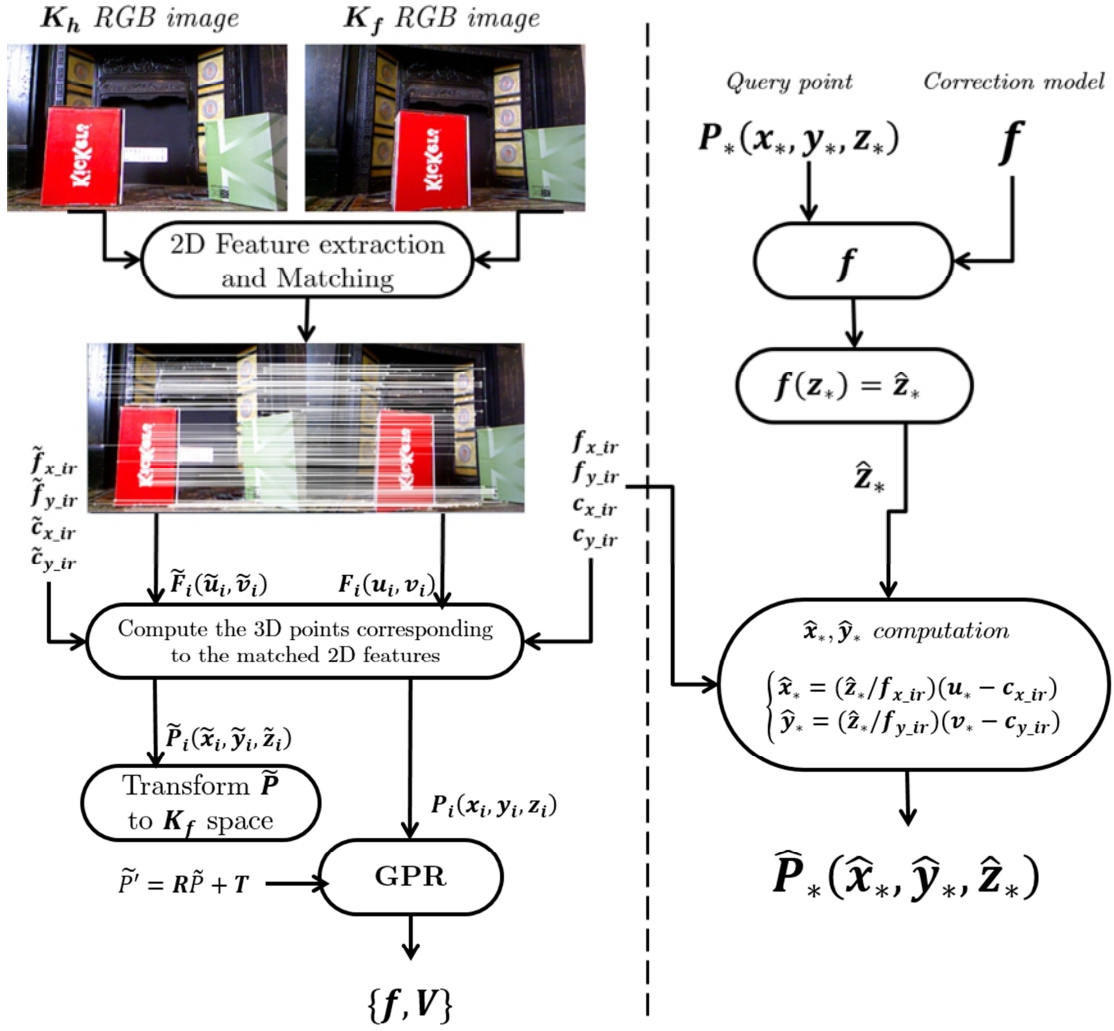


Figure 4.7 GPR-based RGBD drift correction with another sensor. **(a)** Training. **(b)** Correction (test)

- Extract the 2D features from each of the two RGB images delivered by the sensors. The pair of colour images (K_f , K_h) is required in order to extract good distinctive features. Alternatively, one can visually select different pairs of points from anywhere in the colour image. Preferably from regions at close proximity to the camera in order to prevent potential errors. This manual action has become possible due to the knowledge of stereo-calibration regarding the IR/RGB stereo setup.

- The training stage begins with the matching of the extracted 2D features $\boldsymbol{F_i}, \tilde{\boldsymbol{F}}_{\boldsymbol{i}}$, Figure 4.7 (a). The result of the matching is a list of corresponding pairs $\{(\boldsymbol{F_i}, \tilde{\boldsymbol{F}}_{\boldsymbol{i}})\}$.

- The 3D points ($\boldsymbol{P_i}(\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{z_i})$, $\tilde{\boldsymbol{P}}_{\boldsymbol{i}}(\tilde{\boldsymbol{x}}_{\boldsymbol{i}}, \tilde{\boldsymbol{y}}_{\boldsymbol{i}}, \tilde{\boldsymbol{z}}_{\boldsymbol{i}})$) are then computed based on ($\boldsymbol{u_i}$,$\boldsymbol{v_i}$) positions of their respective 2D correspondents ($\boldsymbol{F_i}(\boldsymbol{u_i}, \boldsymbol{v_i})$, $\tilde{\boldsymbol{F}}_{\boldsymbol{i}}(\tilde{\boldsymbol{u}}_{\boldsymbol{i}}, \tilde{\boldsymbol{v}}_{\boldsymbol{i}})$) as well as calibration parameters of the camera, Figure 4.7 (a). After computing the 3D coordinates of points, the actual GPR is applied. The latter computes the correction model $\boldsymbol{f}$. This model will be considered as an extra calibration parameter for the worn sensor.

- Correction phase uses the outputs of the training algorithm ($\boldsymbol{f}$) to readjust the wrongly captured readings, Figure 4.7 (b). The entire depth map $\{\boldsymbol{z}_*\}$ undergoes the correction in the same way the ordinary calibration is applied.

- Corrected depth data $\hat{\boldsymbol{z}}_*$ is used to process the remaining two coordinates $\{\boldsymbol{x}_*, \boldsymbol{y}_*\}$. As a result, a more accurate 3D point cloud $\hat{\boldsymbol{P}}_*(\hat{\boldsymbol{x}}_*, \hat{\boldsymbol{y}}_*, \hat{\boldsymbol{z}}_*)$ is obtained.

The number of features necessary for correction module to work properly depends on the size of images and the field where the end application operates (1% of the pixels is generally sufficient). For experiments with VGA resolution images, 130 samples are adequate, but the greater the number of samples, the better the correction.

The set of training features must be varied and taken at different 3D locations in the scene. Otherwise, the training will not be complete, and the sensor delivers erroneous measurements for the regions where no samples had been provided.

4.3 Real-Time RGBD Data Segmentation

4.3.1 GMM for RGBD Background Subtraction

The segmentation algorithm starts by applying the GMM separately on both frames streamed by the cameras (Depth and RGB). The resulting foregrounds are then fused based on another algorithm. The power of the GPU is widely leveraged throughout this background removal solution to ensure the real-time performance of the system. Real-time responsiveness is crucial for segmentation as it was for the correction because both components (correction and segmentation) constitute a pre-processing module required for the next chapter. From here on, it is assumed that depth data used by the segmentation algorithm is delivered by the correction stage proposed in Section 4.2.

4.3.2 Background Modelling

The GMM is a parametric probability density function represented as a weighted sum of Gaussian distributions [99], see Figure 4.8. The estimation of its parameters is based on the training data either with the iterative Expectation-Maximization (EM) algorithm or the Maximum a Posteriori (MAP) estimation of a trained model. The foreground detection follows the steps listed below:

- Modelling the values of a particular pixel as an m ($3 \leq m \leq 5$) mixture of Gaussians $G\{\mu_i , \sigma_i , w_i\}$; $1 \leq i \leq m$, Figure 4.9 (a). Each distribution has a mean μ_i and a variance σ_i^2, as well as a weighting coefficient w_i to quantify its importance. The weightings w_i are positive and add up to one.

- Determining the Gaussian that corresponds to the background model based on the mean and the variance of each of the m distributions.

- The foreground is then defined as the set of pixels that do not fit into the background model.

- Updating the parameters of the distribution with the newly detected foreground pixels in order to be taken into account in the following detections.

- The regions that do not match one of the m Gaussians representing the background are grouped into a foreground blob.

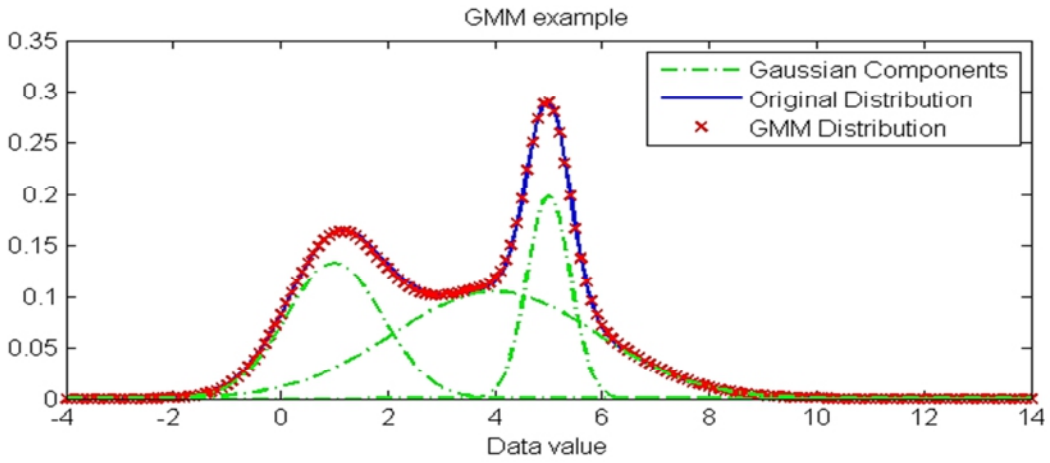


Figure 4.8 GMM Distribution[4]

[4] http://www.maths.adelaide.edu.au/matthew.roughan/code.html. 2015

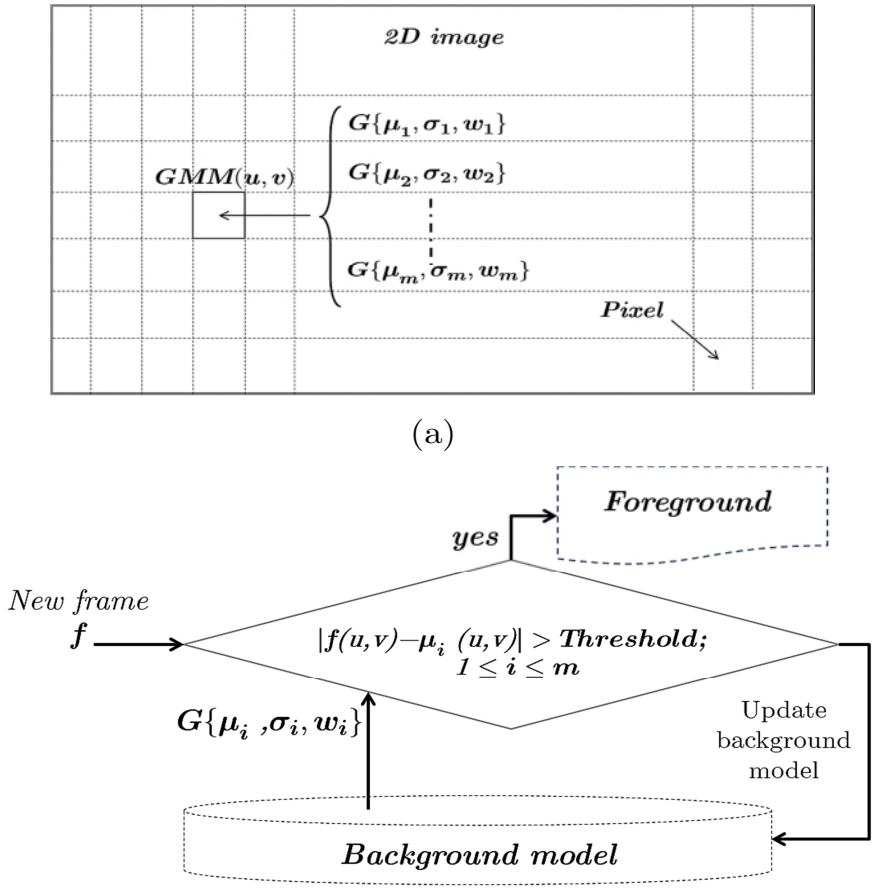


Figure 4.9 GMM architecture. **(a)** Background model. **(b)** Working principle

For every new frame $\boldsymbol{f}$, the GMM computes the distance between the pixel $\boldsymbol{f}(\boldsymbol{u}, \boldsymbol{v})$ and each of the means $\boldsymbol{u_i}(\boldsymbol{u}, \boldsymbol{v})$ characterising the $\boldsymbol{m}$ distributions, Figure 4.9 (b). The new pixel is tested against the highly weighted distributions first, then against the remaining ones according the descending order defined by the weightings. If the new pixel does not match any of the recorded distributions, it will be considered as a foreground element. The background model is also updated with the intensity of the newly acquired pixel, i.e. the parameters of the corresponding distribution (mean, covariance) are re-evaluated with the value of the new pixel.

4.3.3 GMM on RGBD Data

Colour images are perceptually more representative of the real world. Nevertheless, colour representation is naturally sensitive to the illumination perturbations. The depth data, on the other hand, has been proven robust to changes in the lighting of the scene, but it lacks the ability to detect the texture of objects. The concept of fusing depth and RGB images, therefore, emerges as a tool of choice for background removal, which in turn is a prerequisite for marker extraction.

The fusion of the two modalities can be approached in two different manners:

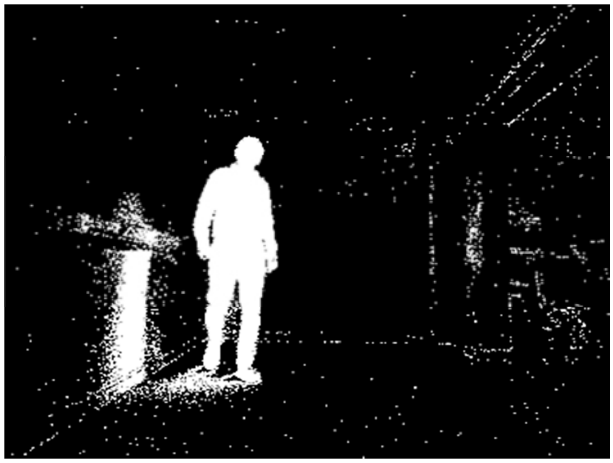
- Either by augmenting the three colour channels with a fourth depth component. Experimental tests have shown that the resulting image still suffers from the classic artefacts caused by light intensity fluctuations. The contribution of depth data in this segmentation model is weakened by the remaining three colour channels. Hence, the resulting image is almost identical to the one that does not take into account the depth information, Figure 4.10.

- Or by completely separating the two modalities during background removal phase, i.e. the GMM is applied on RGB image and depth map independently. The resulting foreground regions are then combined into a single global result, Figure 4.11. This alternative has been selected in this thesis because of its reliability.

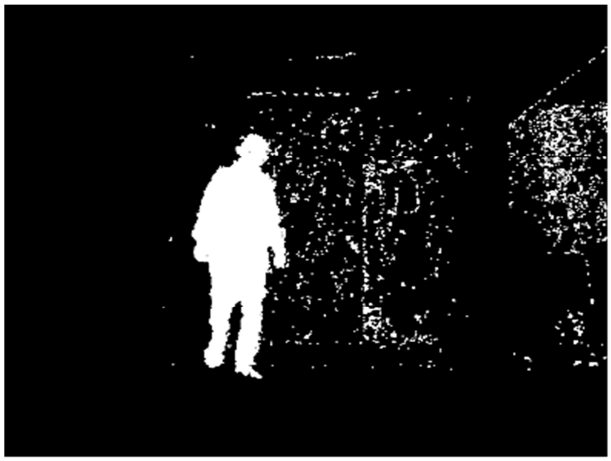
(a)

(b)

Figure 4.10 RGBD segmentation result (Depth as a fourth component). **(a)** RGB. **(b)** RGB+D



(a)



(b)

(c)

Figure 4.11 RGBD segmentation result (Fusion of independent foreground regions). **(a)** RGB. **(b)** Depth (D). **(c)** RGB||D

4.3.4 RGBD Background Fusion

Before combining the results of the two independently segmented images, the foreground resulting from the depth image is mapped with the stereo calibration parameters to the colour frame. This mapping is required because the two modalities do not share the same coordinate system. Algorithm 4.1 is proposed in order to fuse the two foreground regions.

When the two responses at a given pixel in depth and colour binary images are distinct, a decision should be taken regarding the fused result. To this end, the following rule is applied: the pixel retains the same state, i.e. as it was before the confusion occurs, as long as no consecutive three frames with the same pixel value are met. This parameter may be subject to change. Empirically, three was adequate for our experimental scenarios.

This assumption is empirically motivated by the fact that permanent incoherence between colour and depth GMMs (in one image the pixel is white and in the other it is black) is considered as a confusion. In such a situation, it is not possible to decide correctly which modality, depth or colour, holds the actual state of scene's active foreground.

Algorithm 4.1 RGBD foreground fusion algorithm

```
2D binary images: RGB, D, Forg;
//these image containers hold foreground regions only
int ** cpt;
for (u=1;u<=640;u++)// VGA resolution 640X480
{
    for (v=1;v<=480;v++)
        {
         if (RGB(u,v) == D(u,v))
         //if the segmented pixels are the
         //same in both modalities
         {
             Forg(u,v)=D(u,v);
             cpt(u,v) = 0;
         }
         else
         //if RGB and Depth segmentation
         //results are different
         {
           if (cpt(u,v) == +3)
           //the pixel is more likely to follow
           // colour segmentation result
            {
                Forg(u,v) = RGB(u,v);
                cpt(u,v)  = 0;
            }
           else if (cpt(u,v) == -3)
           //the pixel is more likely to follow
           // depth segmentation result
            {
                Forg(u,v) = D(u,v);
                cpt(u,v)  = 0;
            }
           //increment the number of successive
           //times when Forg(u,v) == RGB(u,v)
           else if (Forg(u,v) == RGB(u,v))
                    cpt(u,v)++;

           //decrement the number of successive
           //times when Forg(u,v) == D(u,v)
           else if (Forg(u,v) == D(u,v))
                    cpt(u,v)--;
         }
    }
}
```

4.3.5 GPU Acceleration of the GMM

Here, both colour and depth images are assumed to have a VGA resolution. In order to achieve the segmentation in the GPU, a thread is associated with each pixel. The treatment is, therefore, run on every element independently from its neighbourhood.

Other memory storage optimisations should be taken into account in order to benefit from the hardware architecture. As has been seen in Chapter 3, when the current frame is being processed in the GPU (device), the following pair of images is dispatched as well. Simultaneously, the already available foreground results are delivered back to the central memory.

Due to the large quantity of image data required for the joint colour/depth segmentation, the GMGPU (see Figure 2.34) should be carefully utilised. Some design considerations should also be taken into account to promote the download of contiguous memory chunks instead of a single cell. Figure 4.12 (b) depicts how the array of data structures regarding RGB image as well as their GMM workspace ($\{\boldsymbol{\mu_r}, \boldsymbol{\mu_g}, \boldsymbol{\mu_b}, \boldsymbol{\sigma}, \boldsymbol{w}\}, \{\boldsymbol{\mu_d}, \boldsymbol{\sigma}, \boldsymbol{w}\}$) are recast into a Structure of Arrays. At the request of a given memory cell, the internal design of the GPU causes the content of adjacent cells to be automatically downloaded. This policy is motivated by the manufacturer's assumption about the data stored in neighbouring memory emplacements to be soon requested as well [8].

The programmer should, consequently, take advantage of such a policy by reorganising their data into a Structure of Arrays. In other words, the RGB data should be divided into three arrays, each corresponding to a single channel (Red, Green or Blue), Figure 4.12 (a). The same procedure is applied to the parameters of the Gaussian distributions constituting the GMM, Figure 4.12 (b), (c). On the other hand, the depth map does not require to be reshaped since it has only one component.

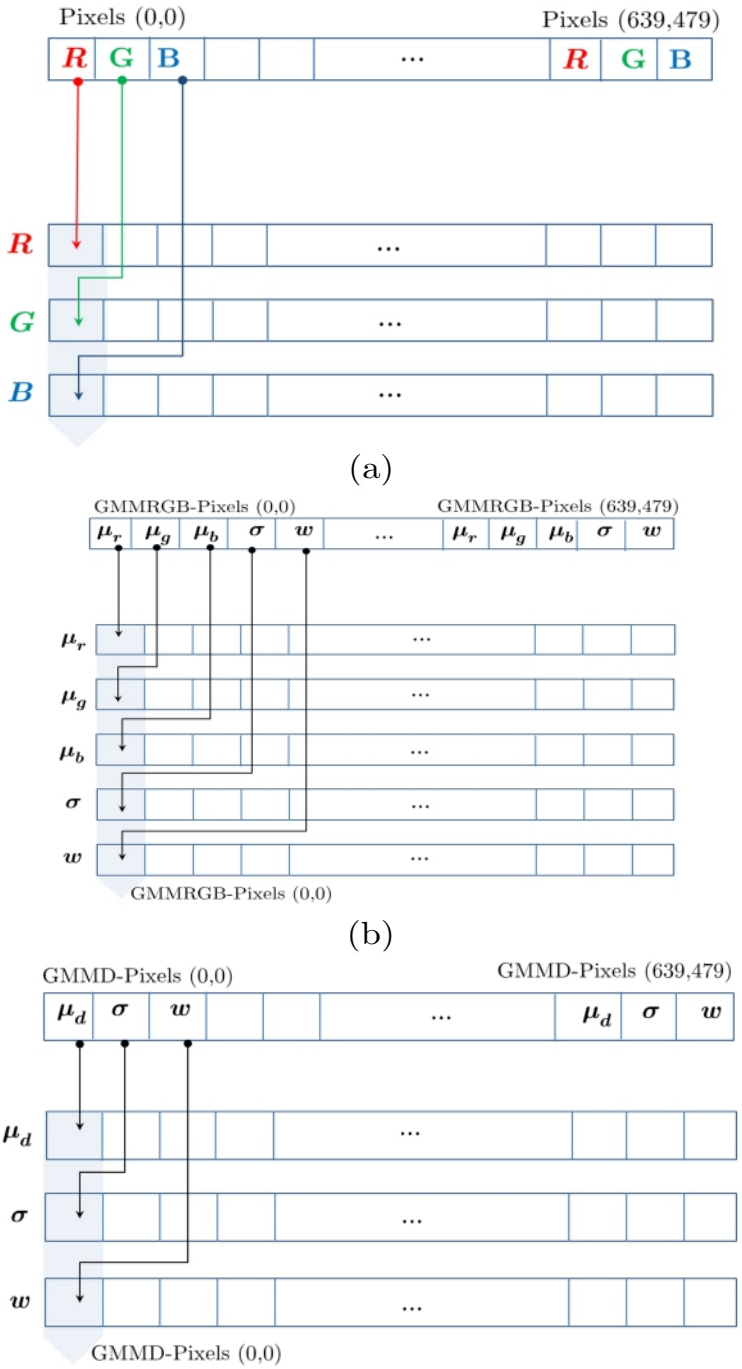


Figure 4.12 Array of Structures to Structure of Arrays transformations. **(a)** RGB image memory coalescing. **(b)** RGB-GMM workspace. **(c)** D-GMM workspace

4.4 Results & Discussions

4.4.1 Correction with Polynomial Interpolation

To illustrate the effect of the proposed sensor-correction method, some experiments were carried out. The previous two Kinects V1 with different measurement precision grades were corrected. Figure 4.14 to Figure 4.16 illustrate two cases where a worn and a healthy sensors were adjusted using an eighth degree polynomial interpolation. In other words, a standard polynomial interpolation is initially tested instead of the GPR in order to approximate the underlying model. Figure 4.14 illustrates the values of the function $\boldsymbol{f}(\boldsymbol{z_f}) = \boldsymbol{z_h}$, where $\boldsymbol{z_f}$ is the shifted depth and $\boldsymbol{z_h}$ is the ground truth reference depth. The most $\boldsymbol{f}(\boldsymbol{z_f})$ fitting line is similar to $\boldsymbol{y} = \boldsymbol{x}$, the more accurate the sensor. In other words, range measurements obtained from the Kinect will be closer to their ground truth counterparts. With the worn sensor, Figure 4.14 (a), the curve representing $\boldsymbol{f}(\boldsymbol{z_f})$ is clearly shifted below $\boldsymbol{y} = \boldsymbol{x}$.

This behaviour occurs because of the overestimation of the distance separating the objects from the sensor. The erroneous interpretation of depth weakens the ability of the sensor to capture correctly the 3D geometry within the functional range. On the other hand, with the good sensor, Figure 4.14 (b), the representative curve is overlapping $\boldsymbol{y} = \boldsymbol{x}$ as long as the depth of the object remains below 4.0m. When the target moves beyond the maximum range advised by the manufacturer ($\boldsymbol{z_f} > 4.0$m), the accuracy declines exponentially (see Figure 4.13) and the corresponding fitting line deviates from $\boldsymbol{y} = \boldsymbol{x}$.
To calibrate the sensor, first, all points $(\boldsymbol{z_{sh}}, \boldsymbol{z_{sh}} - \boldsymbol{z_{cor}})$ are plotted. Then, fitted with a polynomial $\boldsymbol{f}(\boldsymbol{z_f})$ that minimises the gap between $\boldsymbol{z_f}$ and $\boldsymbol{z_h}$ in the Least Squares sense. Empirically, it has been found that an eighth degree polynomial suffices to interpolate the set of points with a relatively small error. The correct depth values corresponding to the shifted ones $(\boldsymbol{z_f})$ are inferred from the evaluation of $\boldsymbol{f}(\boldsymbol{z_f})$ for the whole depth map.

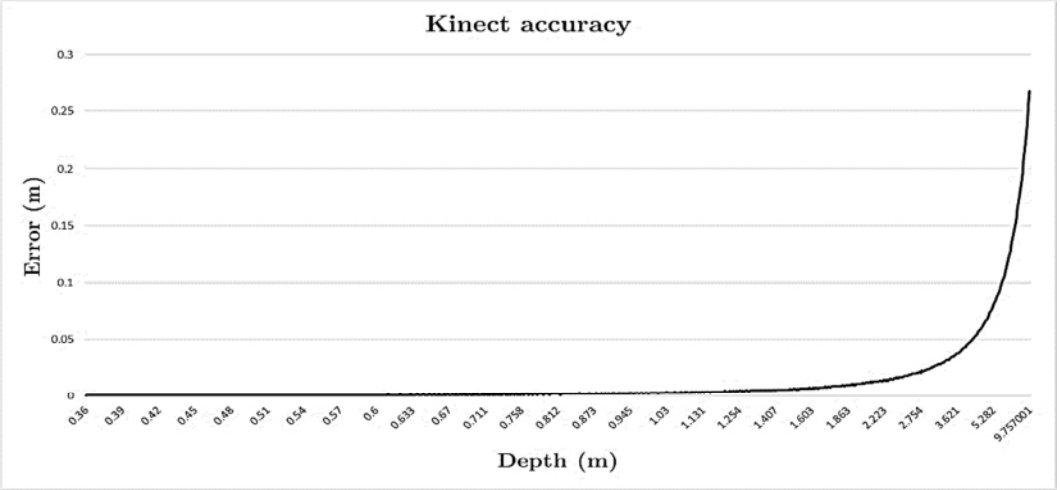


Figure 4.13 Kinect error in depth measurement.

More importantly, the depth values that can cohabit in any point cloud generated by the Kinects is limited to a known list. As a result, to every element in this list z_f is attributed a respective corrected value z_h. The correction of the depth image is therefore reduced to the re-adjustment of the known depth levels [7].

This algorithm (Depth correction) is launched in the GPU, and it is designed to work in a pixel-wise fashion. The correction is applied to every valid depth reading before any further processing takes place. The purpose of pre-processing is the guarantee the best conceivable quality of data. This strategy enables an optimal utilisation of the full potential offered by the sensor.

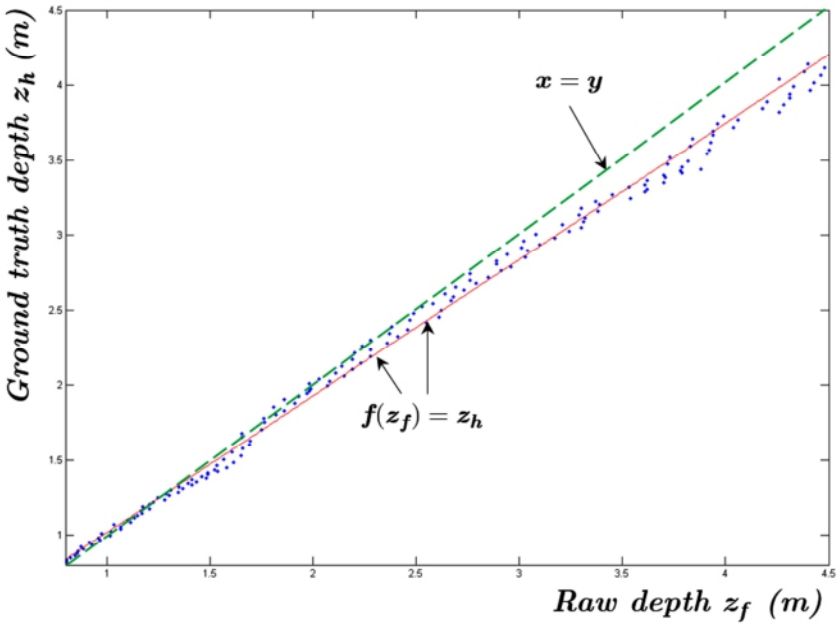
After the correction, see Figure 4.16, the depth data becomes similar to the ground truth one. However, at the further distances, the resolution of the sensor decreases and only some discrete sparse measurements can be acquired. It can be seen from Figure 4.16 that the density of learning samples is inversely proportional to the actual depth.

Table 4.1 reveals the error in the outputs of some Kinects of different measurement accuracies. These cameras will be utilised in the multiview

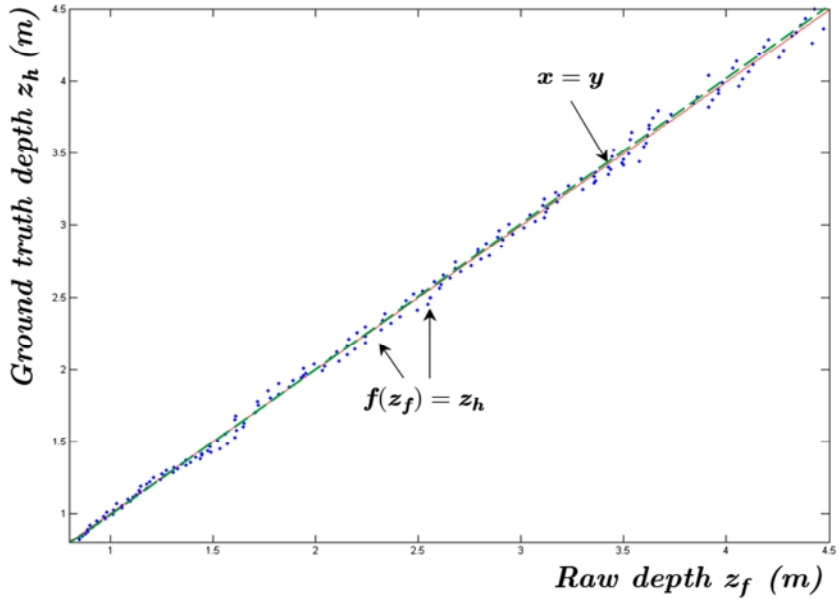
tracking experiments of the next chapter. The larger the difference between RMSEs before and after the correction, the more worn the sensor.

RMSE (m)	Kinect_0	Kinect_1	Kinect_2	Kinect_3
Before	0.1114	0.1474	0.2189	0.0703
After	0.0490	0.0598	0.0633	0.0538

Table 4.1 RMSE before and after interpolation-based correction for some Kinects used in Chapter 5

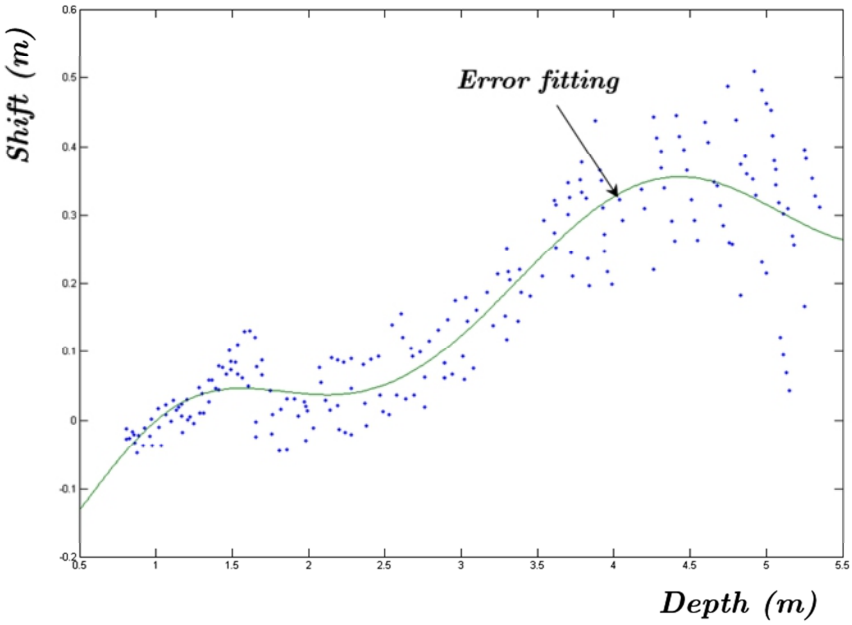


(a)

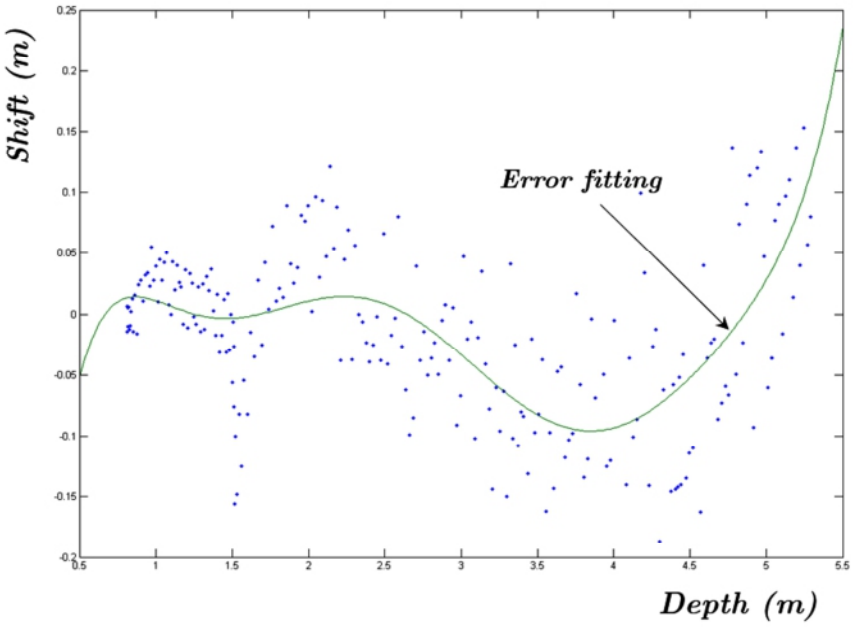


(b)

Figure 4.14 $f(z_f)$ distribution before correction. **(a)** K_f. **(b)** K_h

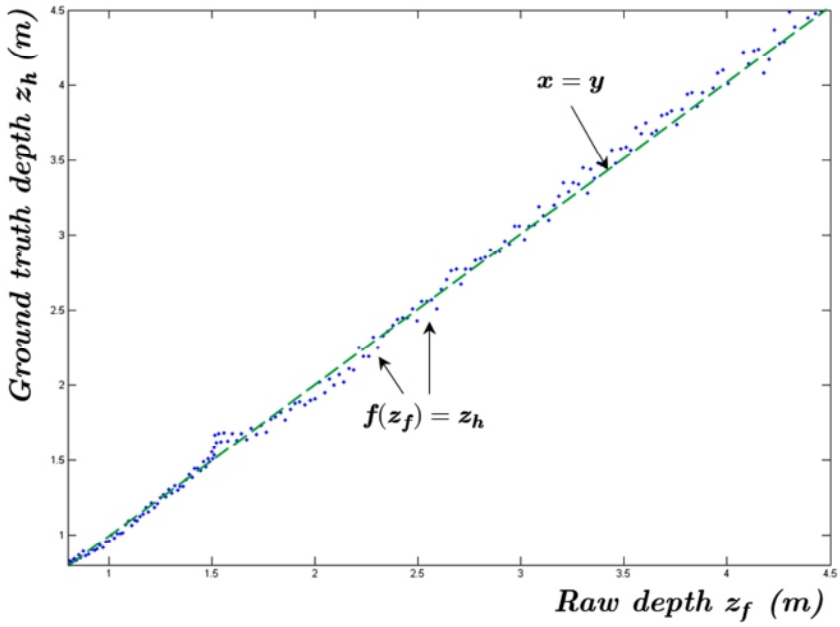


(a)

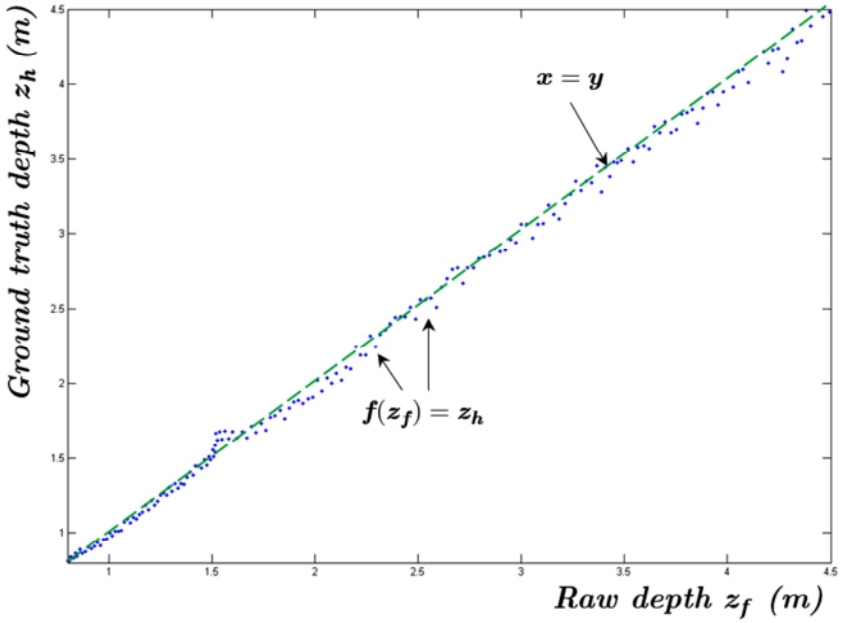


(b)

Figure 4.15 $(z_f - z_h)$ shift fitting. **(a)** K_f. **(b)** K_h



(a)



(b)

Figure 4.16 $f(z_f)$ distribution after correction. **(a)** K_f. **(b)** K_h

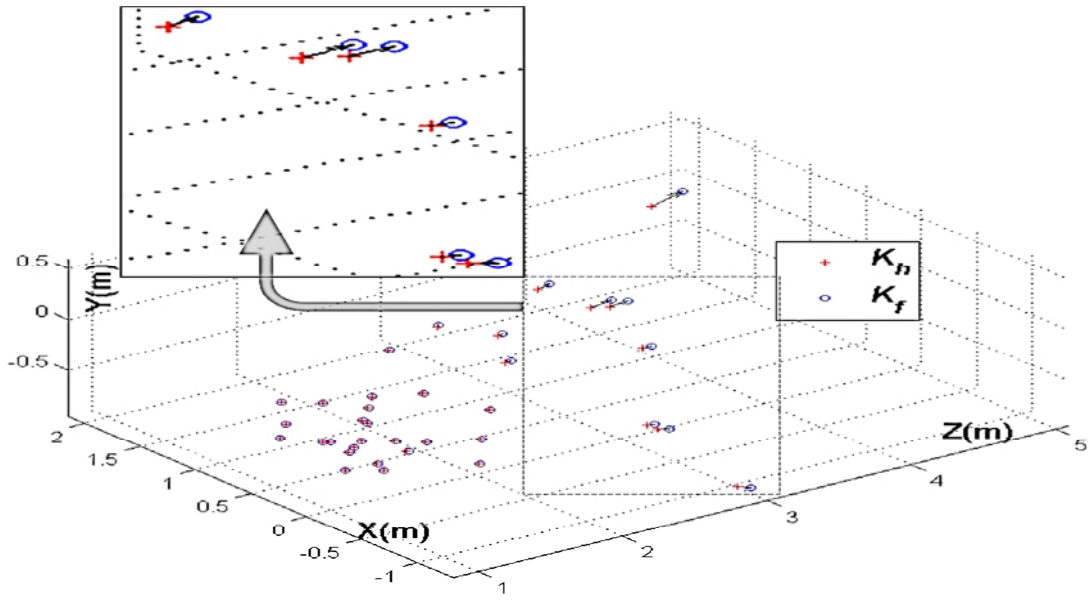
4.4.2 Depth Map Correction with GPR

To validate the GPR depth data correction approach, a series of experiments were conducted on the same cameras that have been used previously. The set of 3D points was extracted from both the healthy ($\boldsymbol{K_h}$) and the worn cameras ($\boldsymbol{K_f}$). After applying ordinary camera calibration, the 3D points delivered by both sensors were plotted on the same graph, see Figure 4.17 (a). The shift incurred by the range data belonging to the worn sensor is expressed by the black arrows stretching from the red crosses towards the blue circles. The length of these arrows quantifies the magnitude of shift. The latter is the distance between the correct points and their drifty correspondents. Furthermore, this distance is proportional to the range separating the sensor from the scene.

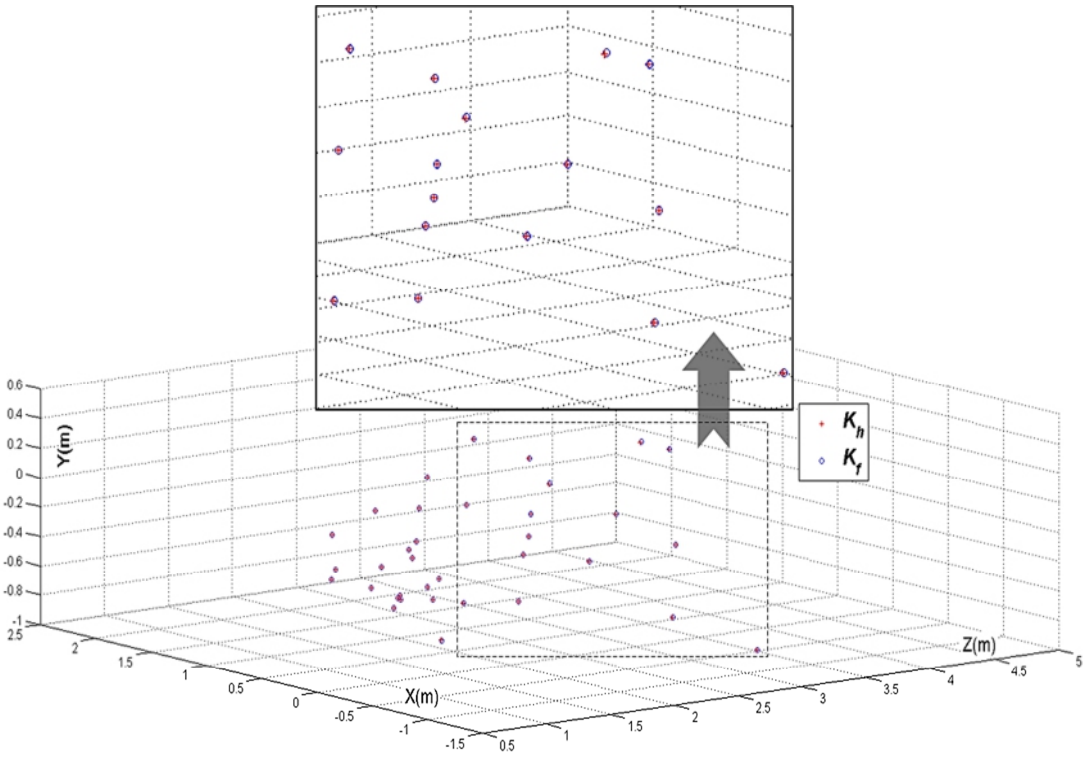
The impact of correction can be seen in the significant improvement of accuracy, Figure 4.17 (b). The wrongly captured depth readings were substituted by their respective estimates computed by the GPR. Error bars (Figure 4.18) depict the error in points' positions before (red) and after correction (green). The error in $\boldsymbol{z}$ is increasing with growing depth values. However, the error in both $\boldsymbol{x}$ and $\boldsymbol{y}$ does not depend only on the depth, but also on the positions of the pixels relative to the centre of the image. Table 4.2 and Table 4.3 illustrate a comparison between the RMSE before and after the correction.

RMSE (m)	Kinect_0	Kinect_1	Kinect_3	Kinect_4
Before	0.1114	0.1474	0.2189	0.0703
After	0.0230	0.0312	0.0462	0.0162

Table 4.2 RMS error before and after GPR correction for some Kinects used in Chapter 5

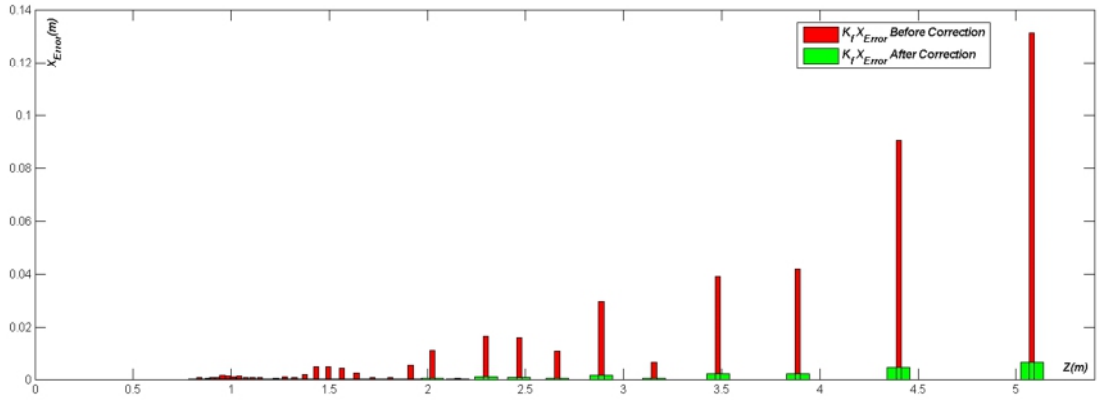


(a)

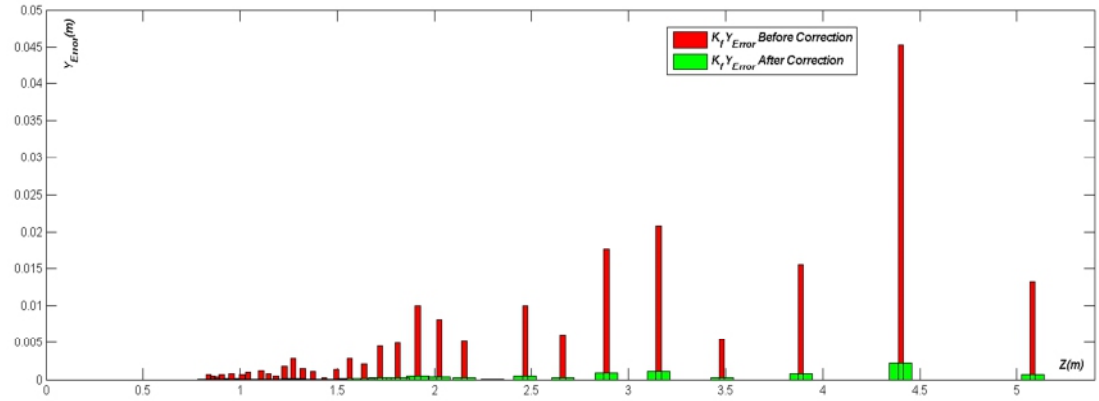


(b)

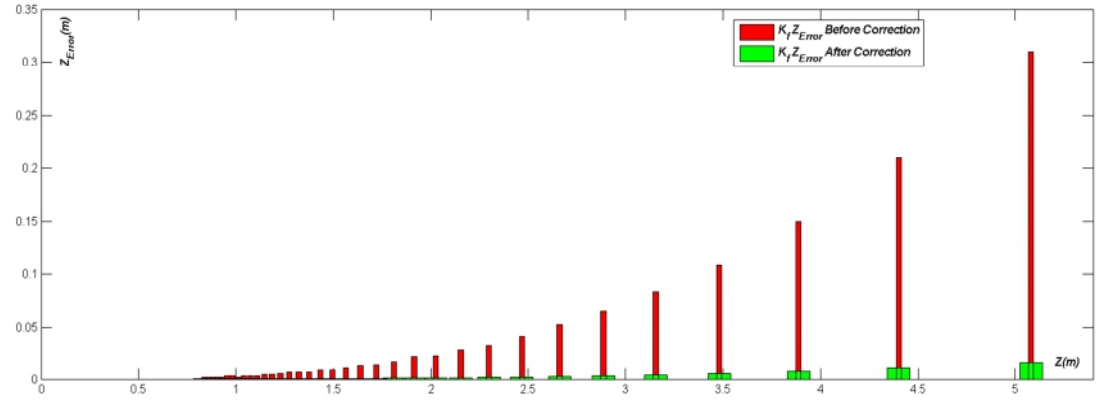
Figure 4.17 GPR correction result. **(a)** $\boldsymbol{K_h}$ and $\boldsymbol{K_f}$ 3D data after ordinary calibration and before GPR. **(b)** $\boldsymbol{K_h}$ and $\boldsymbol{K_f}$ 3D data after GPR correction



(a)



(b)



(c)

Figure 4.18 Depth measurement error before and after GPR correction. **(a)** x. **(b)** y. **(c)** z

RMSE (m)	x	y	z	*Overall*
Before	0.0295	0.0102	0.0743	0.0809
After	0.0015	0.0051	0.0037	0.0065

Table 4.3 RMSE before and after GPR correction for x, y and z components regarding the Kinect used in the experiments

4.4.3 RGBD-GMM Segmentation

Two experiments were carried out on an ordinary scene with two significant illumination changes and a moderately changing background. In addition to a challenging scene, the illumination keeps fluctuating during the whole capture, see Figure 4.19 and Figure 4.20. Evaluation parameter F_1 [100] is computed from the false (F) and the true (T) positives (P) and negatives (N). The resulting four combinations are (TP, FP, TN, FN). TP is the number of foreground pixels that were detected as foreground in the input image. FP is the number of background pixels that were detected as foreground. TN is the number of background pixels that were detected as background. FN is the number of foreground pixels that were detected as background.

Recall is the true positive rate:

$$R = \frac{TP}{TP + FN} \tag{4.10}$$

Precision is the ratio between the number of the correctly detected pixels and the total number of pixels clustered in the foreground blob.

$$P = \frac{TP}{TP + FP} \tag{4.11}$$

The accuracy metric F_1 combines the precision and the recall to objectively evaluate the accuracy of segmentation.

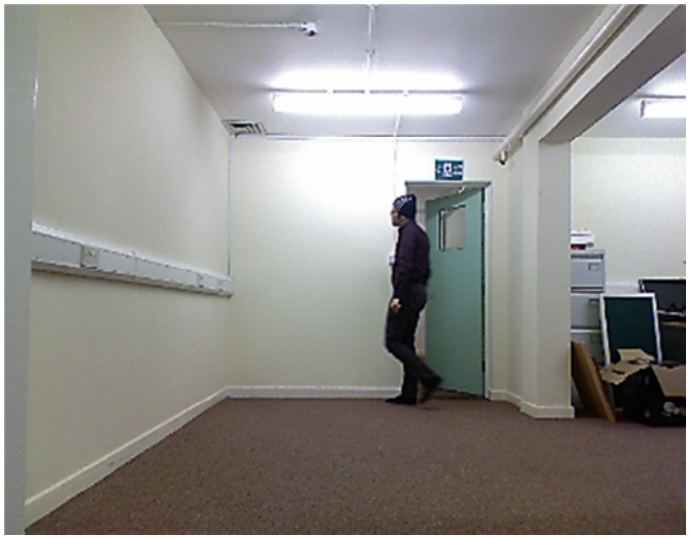
$$F_1 = 2\frac{PR}{P+R} \tag{4.12}$$

$\boldsymbol{F_1}$ is a good indicator of segmentation robustness. The higher it is, the better the performance.

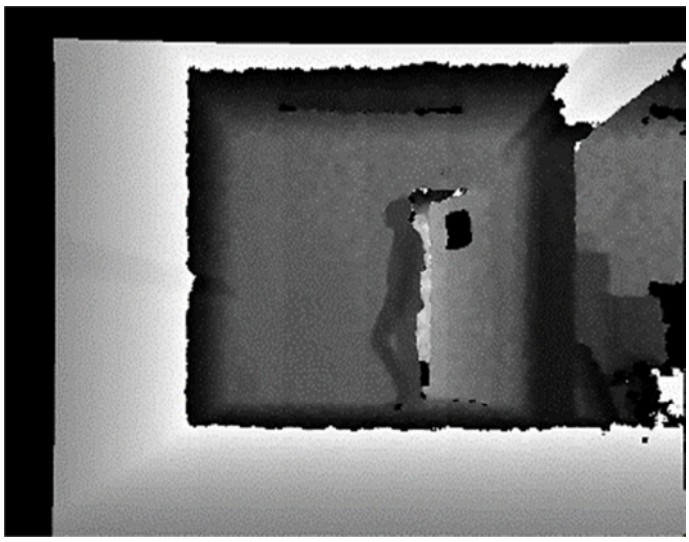
The graphs in Figure 4.21 illustrate the behaviour of $\boldsymbol{F_1}$ in two different experimental scenarios. The first scenario was conducted in a typical indoor environment, where light intensity was only very slightly changing. The imaged scene contains several moving objects that should be segmented from input frames. In the second scenario, the lighting was constantly changing due to the varying intensity.

$\boldsymbol{F_1}$ is plotted for all three alternatives (RGB, Depth, and the fusion of both RGB||D). Figure 4.21 (a) shows that the behaviour of $\boldsymbol{F_1}$ regarding the colour image undergoes two major drops that correspond to important perturbations. These are due to the abrupt decays in the quality of detection (Red). Whereas, the depth map is corrupted by the shadows caused by moving entities (Blue). Its $\boldsymbol{F_1}$ generally remains above 0.90. However, $\boldsymbol{F_1}$ for the fused outputs (Green) remains above 0.97. This value clearly shows the robustness of the proposed fusion against each modality treated independently.

On the other hand, in Figure 4.21 (b) all the three methods are similarly disturbed by the perturbations during the capture because the background was significantly changing. The RGB image remains the most affected, however. The depth map is also affected, but the latter was robust to disturbance. Despite the challenging conditions, $\boldsymbol{F_1}$ generally stays above 0.80 for the fused outputs. Such a robustness allows an accurate detection of the markers in the scene that in turn will lead to significant improvements in the results of the tracking as will be shown in the next chapter.

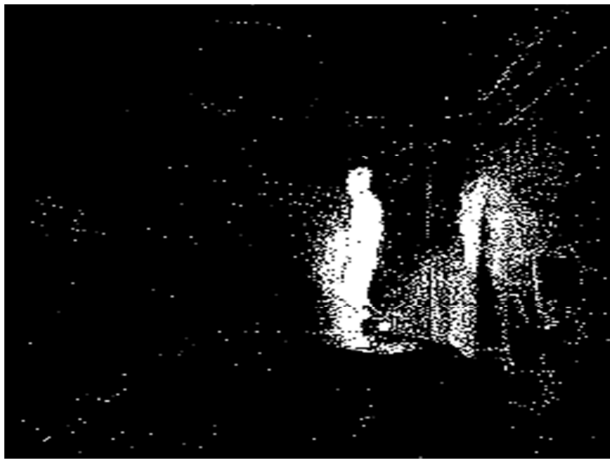


(a)

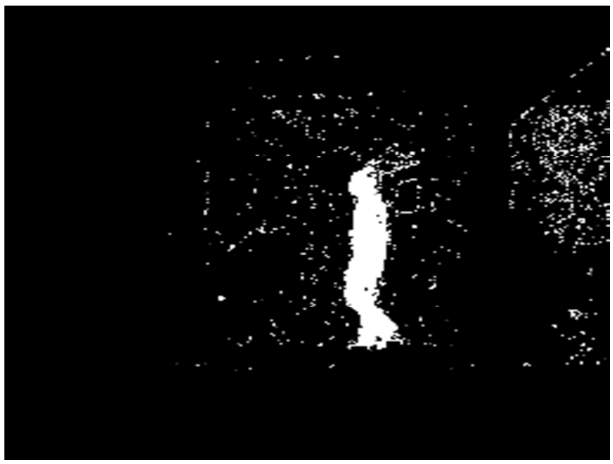


(b)

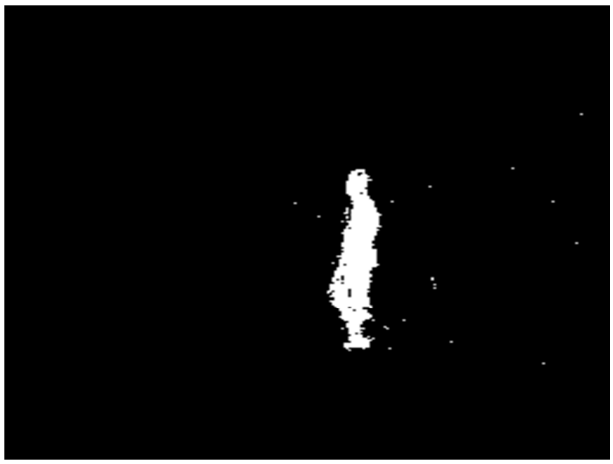
Figure 4.19 GMM input data. **(a)** RGB. **(b)** Depth



(a)

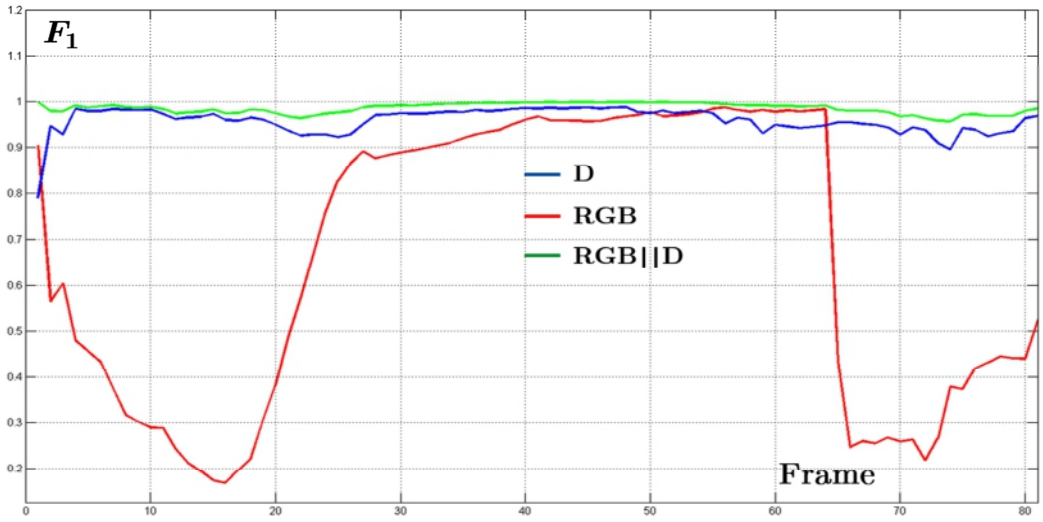


(b)

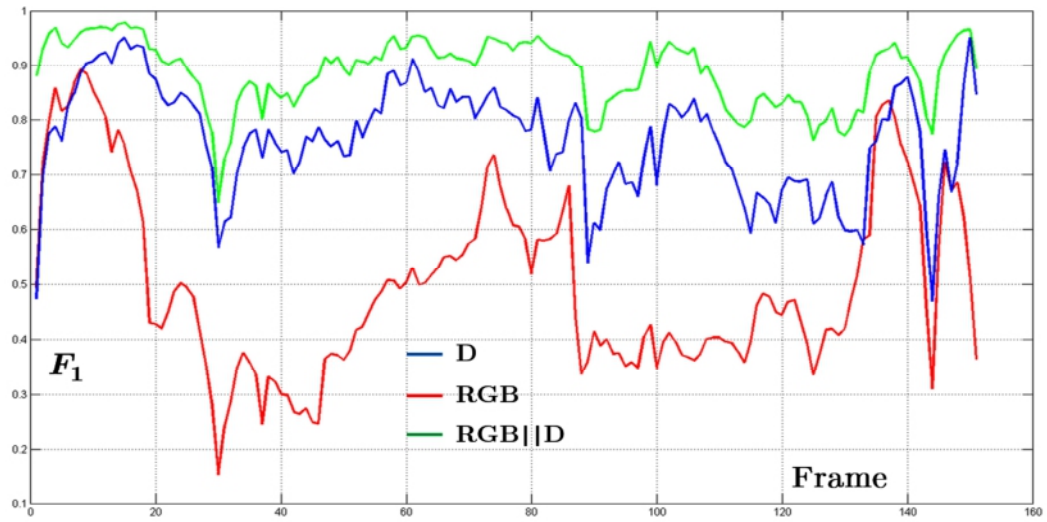


(c)

Figure 4.20 RGB||D GMM segmentation result. **(a)** RGB-GMM. **(b)** D-GMM. **(c)** RGB||D-GMM



(a)



(b)

Figure 4.21 Segmentation results F_1. **(a)** Ordinary scene. **(b)** Challenging scene

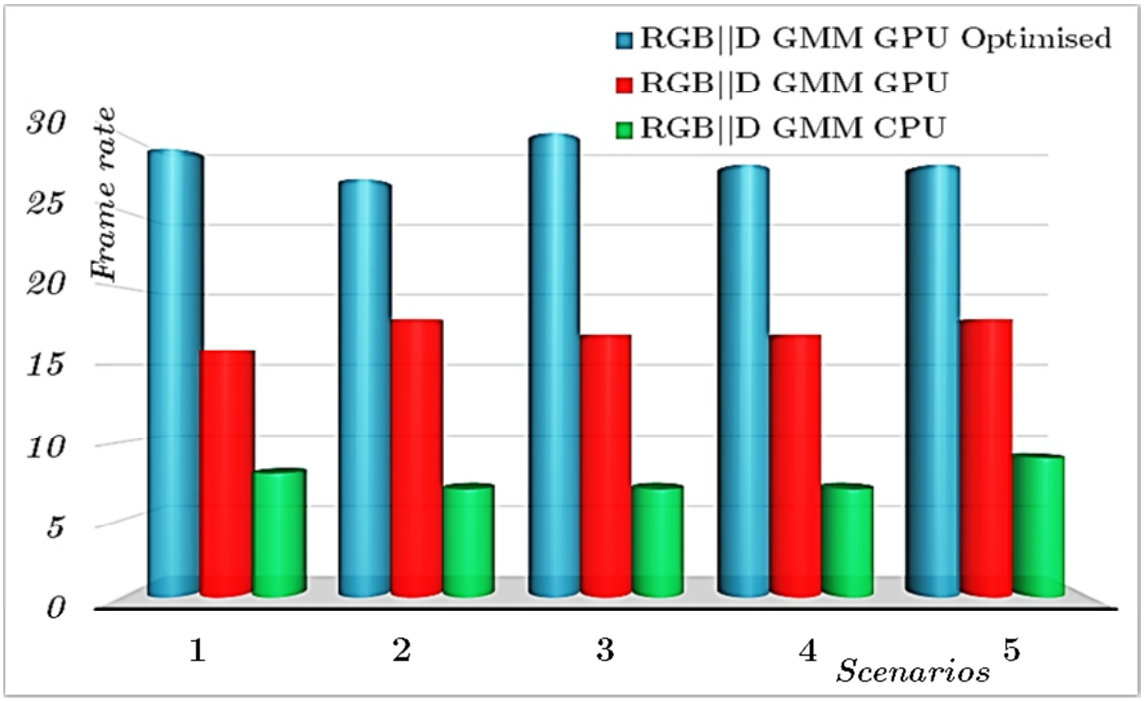


Figure 4.22 Computation time results

4.4.3.1 Processing Time Metrics

Both algorithms, sensor correction and background segmentation, were implemented in the GPU. Hence, the supply of subsequent applications with a full advantage of the available frame rate provided by the Kinect sensor became possible. The frame rate remained close to 30 FPS for the different experimental scenarios. The responsiveness of the GPU can be clearly seen in the blue and red bars against CPU's, green bars in Figure 4.22. More importantly, the improvements after considering memory coalescing and the optimised utilisation of the bus linking the RAM and the GMGPU increased the final frame rate of segmentation to just below 30 FPS.

4.5 Conclusion

A new phenomenon of wear-related deficiency in the capabilities of RGBD sensors was discovered and investigated. The classic mono, as well as stereo calibration techniques were proven incapable of compensating for this lack of

precision. The author proposed a regression-based solution to leverage the capabilities of a healthy sensor for the correction of a worn one. This alternative is underlined by the properties of the depth map and the GPR.

An innovative real-time background removal approach based on a joint RGB/Depth fusion technique was presented. The latter adapts a GPU implementation of the GMM for background subtraction. The proposed solution was validated in real test scenarios under different lighting conditions. The results were promising, even though no additional filtering was incorporated. This fusion approach opens a new perspective on the combination of colour and depth modalities to tackle the inherent problems of colour imagery.

A parallel algorithm has been designed to benefit from the potential asynchronous data exchanges between the host (CPU) and the device (GPU). In addition, data structures were organised in the GMGPU in a way that allows a higher degree of memory coalescing.

Correction and background removal blocks will constitute a single module that serves as a pre-processing framework to clean the raw depth outputs and extract the markers attached to the vehicle from the containing image.

5 Real-Time Multiview Data Fusion for Object Tracking with RGBD Sensors

In this chapter, a novel approach for accurate tracking of moving vehicles with a multiview setup of RGBD cameras is presented. The first step of this solution is a correction phase where the shift that occurs in the outputs of depth sensors when they become worn is eliminated, as has been done in Chapter 4. This problem cannot be fixed with the ordinary calibration procedure. A second depth data correction/enhancement stage is considered to cope with the inherent noise contaminating the sensing. This phase is based on the Kalman filter of Chapter 3. After refining the depth data, markers extraction is performed by means of background subtraction as presented in Chapter 4. The latter (markers) are attached to the robot. Next, comes the actual tracking of the vehicles; i.e. the Robust H_∞ (RF) and the Covariance Intersection (CI). The former is a sensor-wise filtering algorithm used to correct for an unknown vehicle motion. This filter results in the sensor-wise position estimates. The

latter is another algorithm that aims at data fusion. It is proposed for optimal merging of the single-view estimated trajectories.

The computation-costly fragments of the proposed solution are implemented in the graphical processor. As a result, the whole tracking system is capable of operating at up to 25 FPS with a multithreaded architecture of five cameras. Test results show the achieved accuracy and the robustness of the solution to overcome the uncertainties in measurements, as well as in modelling.

5.1 Overview

Image base real-time tracking of moving objects is regarded as a fundamental tool to acquire the dynamics of the physical scene. The ultimate purpose of this acquisition is virtual representation of the real world or the localisation of the entities of interest in their neighbourhood. Surveillance, sports reporting, video annotation, and traffic management systems are a few of the domains that have extensively benefited from advancements in this field [1]. At the highest current performance level, the RGB data persists as a limiting agent in providing a complete view of the real-world. Recent off-the-shelf RGBD sensors, such as the Microsoft Kinect exhibit a high potential for a better perception of the virtual space [2]. These cameras can simultaneously stream both the 3D map of the scene along with its corresponding colour image at a frequency of 30 FPS.

The main purpose of the solution proposed in this chapter is the use of multiple RGBD sensors to localise moving robots accurately. The result is used to feed augmented reality, and robotic systems with real-time 6 DOF pose data.

Cooperative multiview sensing is more adequate than mono view to overcome the occlusions among various angles. Such an architecture leverages the joint action of all the sensors for more reliable tracking. Nevertheless, the processing of large amounts of 3D data flowing from multiple cameras is computationally expensive. As such it may inversely affect the response time of the system. For

this reason, a need for a compromise between performance and response-time arises.

The GPUs are a very powerful tool whenever the load of data treatment can be distributed over several threads running concurrently [3]. In this study, the large amount of 3D data issued by the cameras is subject to smoothing and fusion algorithm. The processing begins with the procedure of data acquisition. Captured data is then sent through a smoothing stage to enhance its quality. The 3D positions of the targets are then computed and forwarded to the Robust H_∞ framework for correction. Based on the single estimates, a data fusion algorithm is adapted to combine all sensor-wise position-estimates into a unique, consistent result.

The major contributions of the present chapter are:

- Coping with the uncertainties in the model describing the motion of the vehicle using the Robust H_∞ filter. The latter has the ability to handle measurements as well as modelling uncertainties.

- A Covariance Intersection algorithm for data fusion with the adaptive weighting coefficients computed from the particular properties of the tracking setup.

The chapter is structured as follows: in Section 5.2, the state of the art regarding image-based tracking applications is discussed. Then the architecture of the whole system is explained in Section 5.3. Details of the first two modules of the system are provided in Section 5.4. Then in Section 5.5, the modelling regarding the uncertainties and how the objects can be accurately tracked without a prior knowledge of their motion are described. In Section 5.6, the covariance intersection technique is presented along with the procedure of weighting coefficients determination required to sort the single estimates according to their quality. In Section 5.7, a clarification of the possibility to compute the orientation of the vehicle in the current solution is provided. The

author's findings are validated with experimental results in Section 5.8, where error graphs obtained from real scenarios are plotted. Finally, possible improvements of the current solution to achieve even better performance are mentioned in Section 5.9.

5.2 Related Works

The tracking problem is divided into three levels of processing: motion detection, object segmentation and object tracking [101]. Single-camera tracking methods suffer from object/object or object/obstacle occlusions. This shortcoming leads to failure as the tracked entities may become incorrectly associated [102]. Zhao et al. [103] presented a method for people tracking with a single camera. They used the 3D shape models of people that were projected into the image space in order to perform the segmentation and resolve the occlusions. Each human hypothesis is then tracked in 3D with a Kalman filter using the appearance of the object constrained by its shape. Okuma et al. [104] proposed a combination of Adaboost for object detection and several variants of the particle filter for multiple objects tracking.

The combination of both approaches results in less failure than by using either one on its own. Moreover, both the detection and the consistent track formation are in the same framework. Leibe et al. [105] presented a pedestrian detection algorithm for crowded scenes. Their method iteratively aggregates local and global patterns for a better segmentation. These and other similar algorithms are challenged by the entirely and partially occluding objects and appearance changes.

On the other hand, cooperative multiview object tracking has the advantage of possessing a broader coverage of the scene compared to a single camera setup [102]. This benefit is also an asset in handling occlusions. The KidsRoom system

[106] developed at MIT Media Laboratory[1] uses a real-time tracking algorithm based on contextual information. The algorithm uses an overhead camera view of the space to minimise the possibility of one object occluding another. The system can track and analyse the actions and the interactions of people as well as objects. The lighting is assumed to remain constant during the runtime. A background subtraction technique is used to segment the objects [107], and the foreground pixels are clustered into 2D blobs. The algorithm then maps each person known to be in the room with a blob in the incoming image frame.

Pfinder (Person-finder) is another real-time system for the tracking and interpretation of human motion [108]. Motion detection is performed using the background removal techniques, see Chapter 4, where the statistics of the background pixels are recursively updated using a simple adaptive filter. The human body is modelled as a connected set of blobs formed by a combination of spatial as well as colour cues. Pfinder has been applied in a variety of applications including video games, distributed virtual reality, interface to information spaces and sign language recognition. The solution proposed by the author of this thesis, however, is only concerned with indoor tracking of moving robots.

A background subtraction procedure is also used to extract the positions of the markers attached to the vehicle. The computation of the actual centres of mass is based on the extraction of contours for every marker and the calculation of their respective zeroth and first moments [109].

From a filtering point of view, object tracking is considered as a sequential recursive estimation problem where each frame is processed within a single time step. The estimator combines knowledge about the previous state (position and orientation) and the current measurement using a state-transition model. The statistics concerning the behaviour of the estimation and the measurements are

[1] http://www.media.mit.edu/. 2015

deduced from the noise processes affecting both measured and estimated positions.

The state/space formalism [110], where the currently tracked object's properties are described in an unknown state vector updated by the noisy measurements, is very well adapted to the object tracking problem. The sequential estimation has an analytical solution under a very restrictive hypothesis.

The Kalman filter (KF) is an optimal solution for the class of linear Gaussian estimation problems [111]. For nonlinear systems, a number of Bayesian techniques have been proposed to perform the optimisation. If a Gaussian distribution is assumed, the commonly used approaches include the extended Kalman filter (EKF) [112] and the Unscented Kalman filter (UKF) [113]. The particle filter is another numerical method that enables an approximate solution to be found with the sequential estimation. [114].

All the previously cited filters are very sensitive to error in the system's model. In other words, if the system is imprecisely modelled, which is indeed very likely in real scenarios [77], estimation accuracy is not optimal. To remedy uncertainty in the system, the Robust H_∞ filter (RF) is known for its ability to cope with the inaccuracies contaminating the model and the measurements. Such an innovative adaptation of the RF for accurate tracking with imprecise motion models has not yet been discussed in the published literature of multiview tracking. In addition, the Covariance Intersection technique [115] is applied to combine the estimates computed from the raw output of each camera in a way that minimises global estimation error [116].

Joint colour/depth information enables the full advantage of both colour image and 3D geometry approaches to cope with the traditional problems of many robotics and computer vision applications. Some examples are: human pose estimation [117], robot navigation [118], SLAM [88], object tracking [119] and 3D scanning [85] to name a few. However, the size of the 3D point data is larger than the corresponding RGB image. Hence, the GPU is leveraged to achieve a

real-time performance, as has been done in Chapter 3. There are several state-of-the-art examples showing that processing bottlenecks could be reduced when the solution is re-designed to run on the GPU. Kinect Fusion [85] and the work of Tong et al. [6] are well-known examples.

5.3 System Overview

5.3.1 Kinect V1 Camera

Although the Kinect is distributed with factory embedded calibration parameters, $(\boldsymbol{f_x}, \boldsymbol{f_y}, \boldsymbol{c_x}, \boldsymbol{c_y})$ intrinsic parameters for both RGB and IR cameras and the extrinsic parameters $[\boldsymbol{R}, \boldsymbol{T}]$ of the IR/RGB stereo setup, the actual resolution of capture may range from less than one millimetre to many centimetres. This plausible difference depends on the state of the sensor, the target application and the nature of the scene [120]. For a robust tracking of moving objects with Kinect, the sensor should be correctly recalibrated. The native parameters are more generic and similar for all the Kinects in the market. However, the frequency of usage and the external factors, that vary widely from one application to another, can significantly affect the precision of measurements [119].

5.3.2 Hardware and Software Configuration

The real-time tracking setup is composed of $\boldsymbol{N = 5}$ Kinects V1 covering a volume of $4 \times 4 \times 3\ m^3$. All the sensors are connected to the same computer (Figure 5.1). The hardware configuration that has been used for this multiview tracking scenario is the same as the one in Chapter 3. The target setup is developed as a multiview ground robot tracking system (Pioneer P3-DX[2] for the current experiments). However, the tracking system can be used to estimate the trajectory of any ground and aerial vehicle moving in indoor environments. The robot moves freely in the space covered by the cameras according to an on-line

[2] http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx. 2015

obstacle avoidance algorithm that runs simultaneously with the capture. A more general motion model is therefore adapted. The last-mentioned accommodates every possible other type of motion describing the displacements and the aspects of the vehicle over time. From a software point of view, the algorithm needs to access the outputs of all the Kinects simultaneously in real time. Thus, Kinect

Figure 5.1 Multi-Kinect real-time tracking system

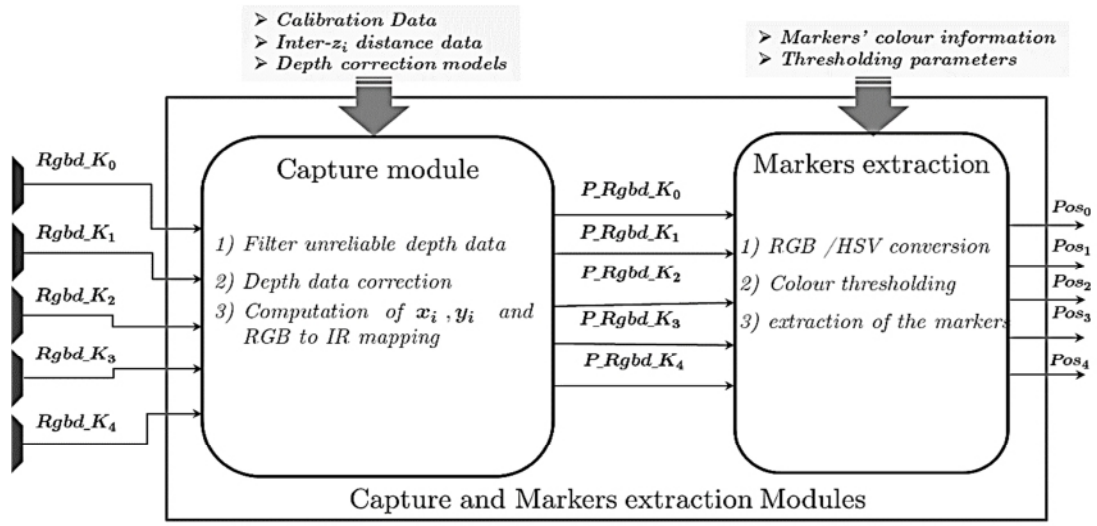
SDK 1.7.0[3] and CUDA[4] is used for GPU programming along with the Boost[5] library.

5.3.3 Real-Time Multi-Kinect Tracking Architecture

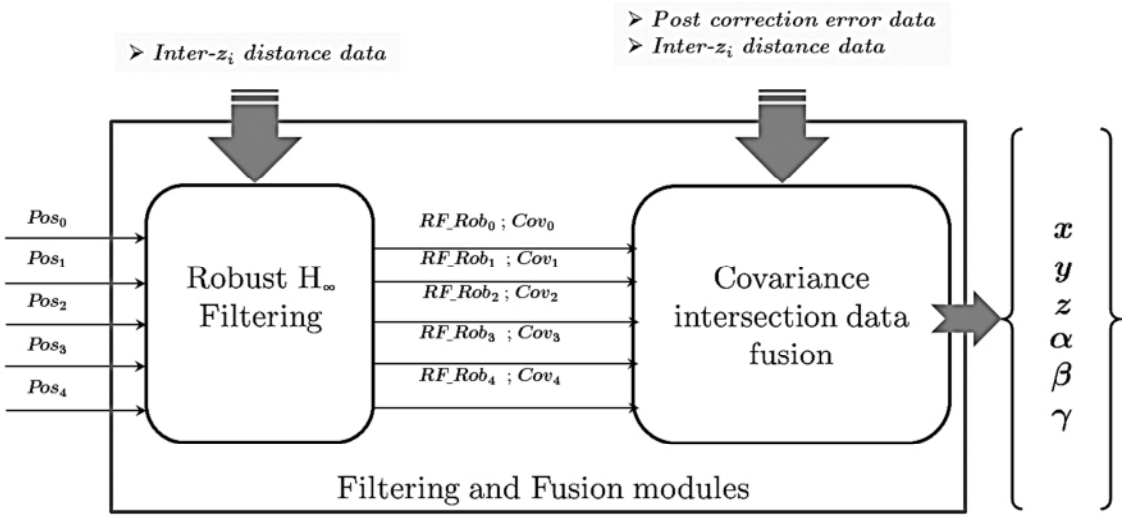
The system consists of four main modules through which flow the RGBD frames streamed by the five Kinect sensors, see Figure 5.2. The sensors are assumed to capture RGBD data concurrently to cooperatively estimate the pose of the robot. Thus, some insignificant interference may appear among them. One might think of Time-Division Multiplexing as an alternative solution that allows a single sensor to be operational at a given frame. Indeed, such a solution leads

[3] http://www.microsoft.com/en-us/kinectforwindows/develop/learn.aspx. 2015
[4] http://www.nvidia.com/object/cuda_home_new.html. 2015
[5] http://www.boost.org/. 2015

to the elimination of interference between IR patterns, but the multiview setup would be reduced to a single view one at every frame. Hence, cameras perform individually and estimation accuracy decreases. On the other hand, our solution takes into account all the data delivered by the sensors at every frame. Such a strategy allows us to benefit from the best of each camera for a more accurate tracking.



(a)



(b)

Figure 5.2 Filtering modules

5.3.3.1 Capture Module

This module is responsible for the delivery of the 3D point clouds to the tracker and the subsequent stages of the tracking system. At this level, a thread is associated with each sensor. Such an architecture permits a complete occupation of both the CPU and the GPU during the capture. Nevertheless, other sensor-related limitations should be taken into account when using multiple Kinects simultaneously. The IR beams emanating from the projectors interfere with each other. They confuse the IR cameras during the evaluation of the disparity as it would be impossible to decide which IR speckle belongs to which sensor. As a result, some holes appear in the 3D data because of the undefined disparity information [45]. In the current architecture, each thread operates independently by loading the data into the GPU and running the following computations:

- Filtering the unreliable z_i elements (empty pixels where no disparity information is readable).
- Correcting the remaining valid depth values using the appropriate correction modules.
- Computing x_i , y_i for only the valid points using the intrinsic parameters of the IR camera.
- Mapping the colour image onto the depth one using the stereo calibration parameters.

5.3.3.2 Markers Extraction Module

To compute the position and the orientation of the robot, three distinctive markers are fixed on its top (see Figure 5.1). The 3D pose of the moving object is obtained by estimating its centre of mass and the corresponding orientation. This processing becomes possible because of the correct association of colour data to the depth pixels. As a result, only the localisation of the markers in the 2D colour image is needed. Then the corresponding 3D positions can be easily

resolved. The background/foreground segmentation module Chapter 4 takes as input the aligned colour image and follows these steps:

- RGB to HSV (Hue, Saturation and Value) conversion. This conversion is motivated by the fact that the HSV space is more robust to changes in light intensity [121].
- Colour thresholding to separate the markers from the background.
- Erosion and dilation of the thresholded binary image.
- Actual extraction of the markers.

The output of this module is the pixel positions of the marker attached to the vehicle.

5.3.3.3 x_i , y_i Computation and Stereo Mapping

The data streamed by the IR camera is just the disparity resulting from the comparison between the shape of the reflected pattern and its reference correspondent. These disparity pixels are rendered to the actual depth measurements. The latter indicate how far the objects are from the sensor, by the driver responsible for the exchange of data between the computer and the device.

Until now, the depth corresponding to a given pixel in the colour image cannot be inferred because the two cameras have different coordinate frames. Thus, an accurate stereo calibration must be performed to get the right $[\boldsymbol{R}, \boldsymbol{T}]$ transformation that links both cameras.

The computation of the 3D $\boldsymbol{x_i}$, $\boldsymbol{y_i}$ coordinates is possible after using the $\boldsymbol{z_i}$ enhanced by Kalman scheme (Chapter 3) as shown in Equations (5.1) and (5.2).

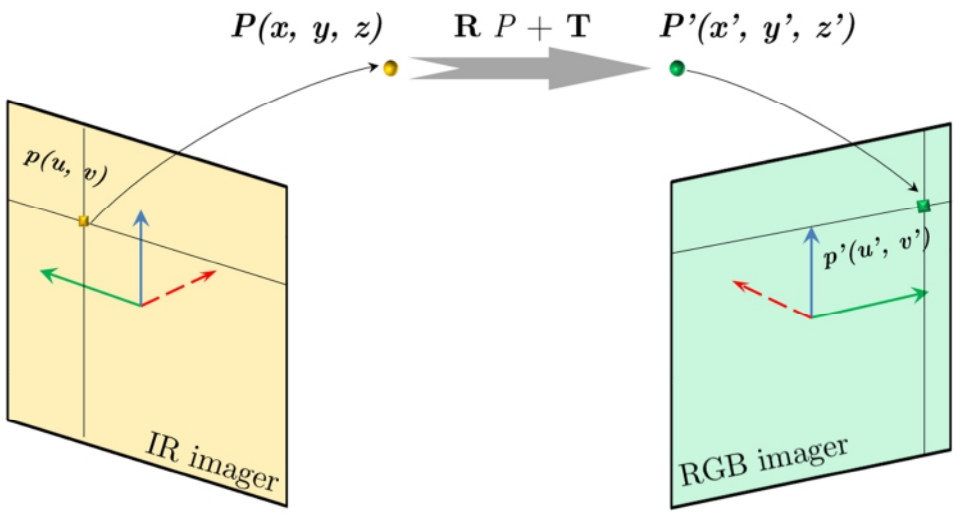


Figure 5.3 Kinect IR/RGB mapping

The achievement of this step requires knowledge of the calibration parameters characterising the IR camera:

$$x_i = (\ u_{i\_ir} - \ c_{x\_ir})\ \ z_i\ /\ \ f_{x\_ir} \tag{5.1}$$

$$y_i = (\ v_{i\_ir} - \ c_{y\_ir}\ )\ \ z_i\ /\ \ f_{y\_ir} \tag{5.2}$$

As shown in Figure 5.3, stereo calibration parameters $[\boldsymbol{R}, \boldsymbol{T}]$ relating the coordinate systems of RGB and IR cameras belonging to the same Kinect are used to transform the point $\boldsymbol{P}(\boldsymbol{x_i}\ ,\ \boldsymbol{y_i}\ ,\ \boldsymbol{z_i}\ )$ from the IR coordinate system to the RGB one $\boldsymbol{P'}(\ \boldsymbol{x'_i}\ ,\ \boldsymbol{y'_i}\ ,\ \boldsymbol{z'_i}\ )$, Equation (5.3).

$$\mathrm{P}' = R\mathrm{P} + T \tag{5.3}$$

Afterwards, $\mathbf{P}'$ is re-projected to the RGB imager using the intrinsic parameters of the colour camera, Equations (5.4) and (5.5), to complete the correspondence between the 3D points and their colour.

$$u_{i\_rgb} = (\ \ x'_i \ \ f_{x\_rgb} \ / \ \ z'_i\ ) + \ \ c_{x\_rgb} \tag{5.4}$$

$$v_{i\_rgb} = (\ \ y'_i \ \ f_{y\_rgb} \ / \ \ z'_i\ ) + \ \ c_{y\_rgb} \tag{5.5}$$

All computations of this stage are executed in parallel for every pixel inside the GPU. The output is a coloured 3D point cloud, where every point $\boldsymbol{P}(\boldsymbol{x_i}\ ,\ \boldsymbol{y_i}\ ,\ \boldsymbol{z_i}\ )$ has its own colour information and world coordinates.

5.3.3.4 Robust Filtering Module

The purpose of the filtering module is quality enhancement of the position and the orientation information issued by the stages above. It acts on the whole trajectory traversed by the moving robot over time by filtering it according to a roughly predefined state-transition motion model. However, for the sake of generality, i.e. to allow the solution to work for any ground or aerial vehicle, it is assumed that the exact motion model regarding the object is not available. Hence, a classical Newtonian framework is chosen with rough parameters.

The loose fit of this standard model to the actual system can be compensated for as well when applying the robust H_∞ filtering scheme [122]. This filter is capable of dealing with the uncertainties in system's and measurements' matrices. In the current case study, it is possible to correct for the lack of knowledge about the modelled system by adapting the filter to Newton's equations of motion.

This filter is implemented in the CPU, and its output is the filtered object position and orientation resulting from every camera alone. The filter is applied to the tracking data delivered at a given time-step (the actual frame that is being processed by the pipeline), not on the entire depth map as with the correction led by the KF in Chapter 3. In addition, it takes intuitively into account the whole history of estimation since the beginning of capture.

5.3.3.5 Data Fusion Module

After getting each sensor's result regarding position and orientation data from the previous filtering algorithm, comes the combination of all the sensor-wise localisation information to produce a more complete and accurate result. Prior to this stage, the output of every camera was treated separately by its respective thread. The multithreaded design permits an optimised usage of the multicore processor.

The fusion of the single estimates begins with the transformation of all the sensor-wise positions into a common reference frame. Afterwards, the mapped data are combined with a covariance intersection technique [123][115]. The latter allows optimal merging of all the outputs of the cameras into a single and more precise estimate benefitting from the best of each sensor.

The entire procedure is summarised in the following steps:

- Transforming each position data for every Kinect to a global reference frame.
- Applying the covariance intersection algorithm to the data in order to produce decent single position and orientation information.

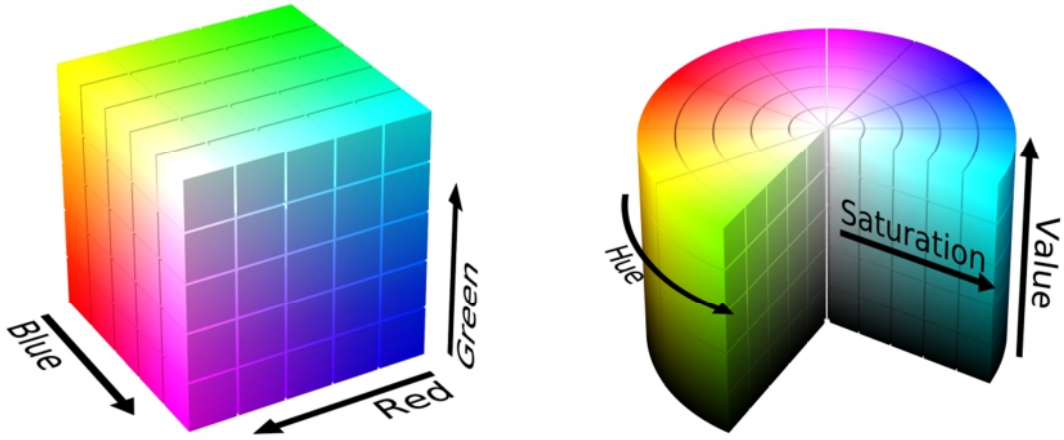
This procedure is also executed in the CPU because the subject data is just the $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ positions delivered by the cameras of the setup.

It is necessary to overcome the occlusions when the markers are not seen from all the viewpoints. In addition, if more than one position information is available, it is possible to fuse them altogether using a set of weighting coefficients. These coefficients promote the most accurate estimate. For instance, the accuracy is determined by the estimation-error covariance matrices resulting from RF module.

5.4 Capture and Marker Extraction

5.4.1 RGB to HSV Conversion

RGB colour model, Figure 5.4 (a), is commonly used in computers and electronic systems such as televisions and photographing cameras. However, this



(a) (b)

Figure 5.4 Colour representation. **(a)** RGB. **(b)** HSV

well-established colour-coding experiences some inaccuracies when the colour is seen from a perceptual point of view [121]. In other words, when the problem of deciding whether an object of a known colour can be localised in a given image is encountered, a significant difference between the respective RGB combinations of the source and the target regions is noticed. This difference appears even when a tiny change in lighting intensity occurs.

On the other hand, HSV colour coding (Figure 5.4 (b)) has been proven to withstand robustly unsteady lighting conditions and shadows [124]. Figure 5.5 depicts a case of a homogeneously coloured surface (orange coating). However, exposure to light creates many appearance differences amongst various regions. These areas are supposed to have the same colour (orange). The distance (L_2 norm in the 3D space between two colour codes) in the RGB space between the two regions surrounded by the black squares is 50.16 pixels Figure 5.5.

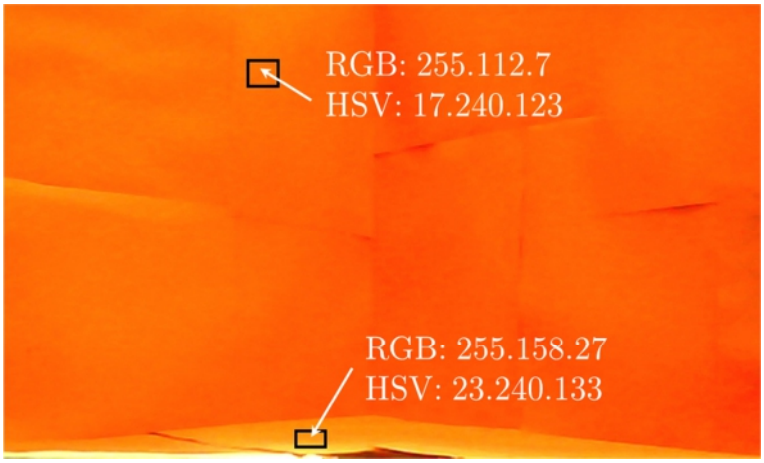


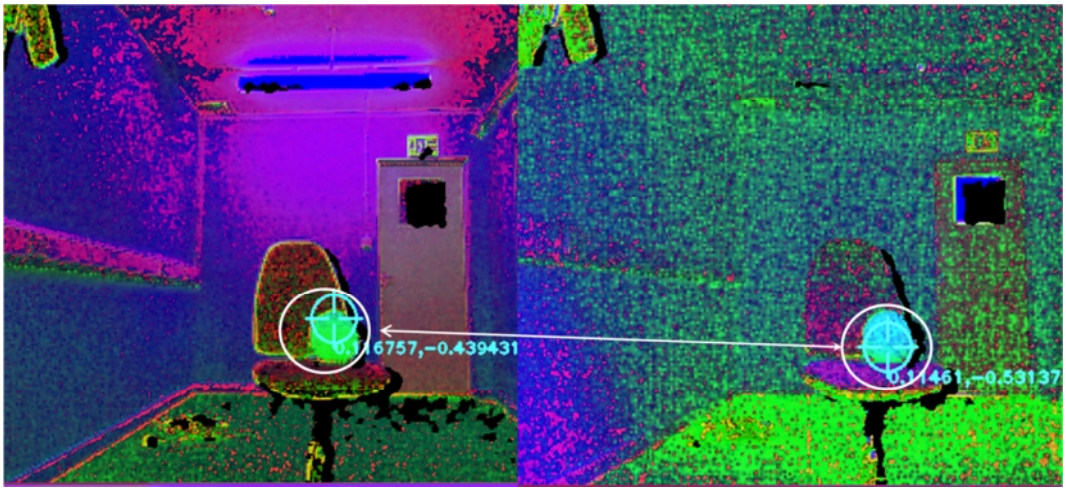
Figure 5.5 RGB vs HSV. RGB_Diff = 50.16 pixels; HSV_Diff = 11.66 pixels

Whereas, in the HSV space it is only 11.66 pixels. This property of HSV coding helps enormously in localising an object of a given colour in the image streams. In the present study, the HSV colour coding is therefore convenient for extracting the markers quickly and efficiently from the images.

As the robot is assumed to move freely over different positions, where luminosity is not necessarily similar, the tracking solution should be robust to unstable lighting conditions in order to maintain an accurate tracking during the whole scenario.

In the following example, a localisation experiment is conducted on a yellow ball resting on top of a chair in different lighting conditions (with the light of the room switched on then off). The objective of markers extractor is their separation from the background. As depicted in Figure 5.6, the yellow ball captured by Kinect is detected in the image with the light of the room switched on (the left image), and the image on the right, obtained with the light switched off. The HSV images in Figure 5.6 (a) are steadier as the distance separating the colours of the target in the two different images is smaller (34.91 pixels). The ball (in the white circle) appears clearly with almost the same HSV colour. Whereas, in the RGB images (Figure 5.6 (b)) the distance is 206.43 pixels. As a consequence, the tracking cross does not appear on the right RGB image as the

algorithm was initially set to track the target whose colour was sampled from the scene with the light switched on.



(a)

(b)

Figure 5.6 Colour distance. **(a)** HSV 34.91. **(b)** RGB 206.43

This example explains the necessity for converting Kinect's colour stream from RGB to HSV. Computationally, the complexity of conversion algorithm is linear to the size of the RGB image $O\,(640 \times 480)$. In addition, this operation is scheduled on the GPU because the same conversion kernel runs in parallel for all the pixels in the image.

5.4.2 Colour Thresholding and Morphological Operations

After converting the image to HSV format, the localisation of the areas that have the same colour as the target is undertaken (target's colour has been initially captured in normal lighting conditions).

The aim of thresholding and binarisation is the recognition and the isolation of the three yellow markers from the background. Then, the alignment between the colour image and the depth map enables the resolution of the 3D coordinates regarding each marker.

Thresholding parameters should be relaxed to fit all possible colours that can be taken by the markers over different lighting conditions during the whole scenario. After that, comes the binarisation of the HSV image by attributing a white colour to the markers and a black to all the remaining regions in the frame (background), Figure 5.7 (b). The extraction of the markers in the binary space facilitates the computation of the respective centres of mass.

Image-erosion is applied afterwards to the binary image in order to eliminate the disturbing noise spots, followed by the dilation of the eroded regions to recover the areas corresponding to the markers.

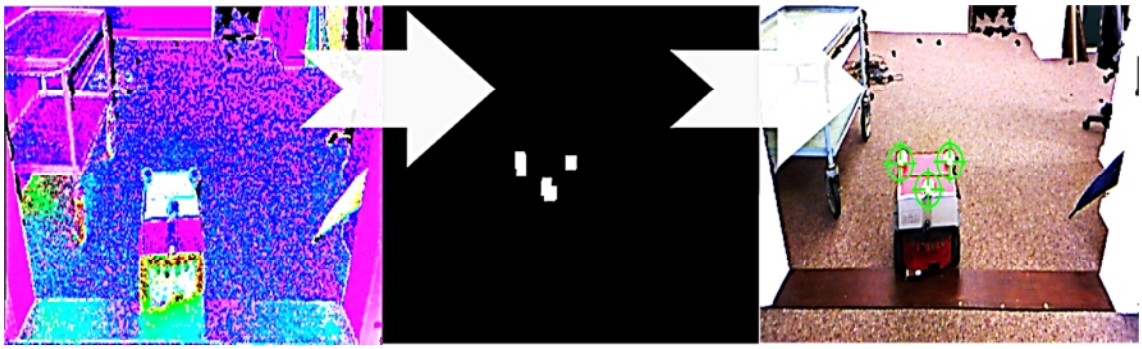
The 3D location of the robot is superimposed on the centre of the 3D triangle formed by the three markers. All the markers should be visible to at least one camera for the computation of position. After localising the markers in the binary image, comes the actual computation of their centres of mass. The latter starts with the extraction of the contours for every marker in the binary image (white dots in Figure 5.7 (b)). Afterwards, comes the computation of the zeroth and the first moments for each marker in the binarised image [109].

Analytically, the moments of a two-variable function are given by:

$$\mu_{m,n} = \iint_{-\infty}^{+\infty} (x - c_x)^m (y - c_y)^n f(x, y)\, dydx \tag{5.6}$$

Here, $\boldsymbol{f(x, y)}$ is the actual image that is assumed to be continuous. The discrete version of Equation (5.6) is:

$$\mu_{m,n} = \sum_{x=0}^{\infty} \sum_{y=0}^{\infty} (x - c_x)^m (y - c_y)^n f(x, y) \tag{5.7}$$



(a) (b) (c)

Figure 5.7 Markers extraction. **(a)** HSV image. **(b)** Blobs in the binary image. **(c)** Tracked markers

The moments are calculated at the origin $(\boldsymbol{c_x}, \boldsymbol{c_y}) = (\mathbf{0}, \mathbf{0})$ as the mean value. Equation (5.7) becomes:

$$\mu_{m,n} = \sum_{x=0}^{\infty} \sum_{y=0}^{\infty} x^m y^n f(x, y) \tag{5.8}$$

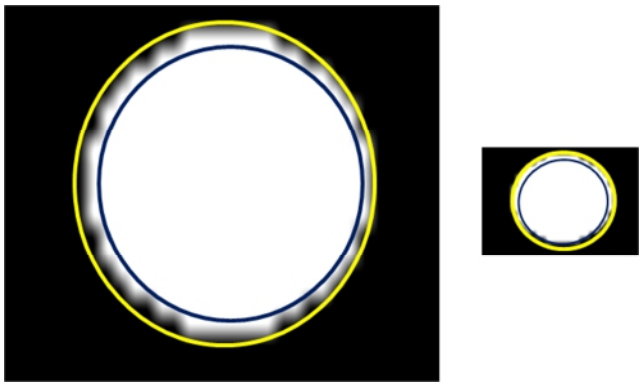
To compute the centroid of the markers in the binary image, the first moments $\boldsymbol{\mu_{01}}$, $\boldsymbol{\mu_{10}}$ are calculated along with the zeroth one $\boldsymbol{\mu_{00}}$ (the area covered by the marker). The coordinates of marker's centroid are;

$$(x_0, y_0) = \left(\frac{\mu_{10}}{\mu_{00}} , \frac{\mu_{01}}{\mu_{00}} \right) \tag{5.9}$$

For a binary image, the moment is the sum of white pixels' coordinates forming the contour around the area that corresponds to the marker.

Figure 5.7 summarises the entire extraction. This technique is robust to noise. The centroid might be a little bit shifted because of some noisy contour elements. However, the error in its position does not significantly affect the accuracy of tracking even when the target is further away. The scale of the markers is inversely proportional to the distance from the sensor. In other words, when the robot is far away from the camera, the regions corresponding to the markers in the image become smaller. Consequently, the error in their position-estimates decreases proportionally. The size of the marker in the image also depends on the size and the shape of the patch that was used in the morphological operations.

Figure 5.8 depicts the relationship between the accuracy of centroid and contour noise originating from the discrete nature of the image data. Before computing the centroid of the marker, the observed pixel-contour is bounded by two circles where the inner (blue) represents a lower bound.



(a) (b)

Figure 5.8 Marker's size and contour

The disc delineated by this circle contains white pixels only. On the other hand, the exterior circle is the upper limit joining the white pixels. Using the two circles one can compute an accurate centroid for the marker.

The noisier the contour, the wider the area between the two circles and their respective radii. This behaviour contributes an extra noise to the detected centroid. Thus, accurate detection is seen in smooth contours in both situations:

When the marker is large, this corresponds to a higher resolution observed when the target is close to the camera, the difference between the radii of the two circles is small even if the area separating them is large (Figure 5.8 (a)). Alternatively, when the marker is small (target further away from the camera), the two circles become almost superimposed. Such a situation also leads to a reasonably accurate detection of the centroid based on the available data and the assumption that the original marker was circular (Figure 5.8 (b)). Subsequently, the heading of the robot can be obtained from the centre of the triangle defined by the three markers and the frontal one.

5.5 Robust Filtering

5.5.1 Motion Model

Raw Kinect position measurements are only precise at a close range because error in position remains below 5.0cm for depth measurements of up to 3.0m without the correction module seen in Chapter 4. Whenever one goes beyond this distance or the sensor becomes worn, the error can grow up to $\pm$20cm. On the other hand, when the correction module is used, the error becomes less than 5.0cm for the depth value rising up to 4.0m. However, the accurate tracking of moving objects requires a higher accuracy with an acceptable additional computational load. To fulfil this requirement in a relatively large indoor space, the robust H_∞ filtering [122] is adapted to every trajectory delivered by a single camera. The need for such a filtering scheme is motivated by its ability to deal with the problem of uncertainty in the model describing the motion of the vehicle as well as the measurements. If the filter over fits the imprecise model of the vehicle, the tracking would fail within a few iterations.

The tracked entities are assumed to move irregularly. The translation, the rotation and the occlusions between the rigid bodies (robots, obstacle) often occur in real situations. As a consequence, a constant velocity model is difficult to adapt. On the other hand, acceleration of ordinary ground and flying vehicles is more stable and does not change in magnitude as it does in sign, because of the smooth and gradual variations of the velocity. Consequently, the motion of the vehicle follows a Newtonian model with a varying velocity and a bounded acceleration.

To correct the raw position supplied by the cameras in real time, the filter should be able to predict robustly the state of the vehicle $[\boldsymbol{x_k}\ \boldsymbol{y_k}\ \boldsymbol{z_k}\ \dot{\boldsymbol{x}}_{\boldsymbol{k}}\ \dot{\boldsymbol{y}}_{\boldsymbol{k}}\ \dot{\boldsymbol{z}}_{\boldsymbol{k}}]^{\boldsymbol{T}}$. Subsequently, it should also be able to correct the state after obtaining the measurements and the control input (acceleration). However, the filter is not applied to the orientation data. The computation of the latter is based on the estimated positions of the markers and the centroid of the robot.

For the position $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ of a given marker, the motion model will be:

$$\begin{cases} x_{k+1} = x_k + T\dot{x}_k + \dfrac{T^2}{2}\ddot{x}_k \\ y_{k+1} = y_k + T\dot{y}_k + \dfrac{T^2}{2}\ddot{y}_k \\ z_{k+1} = z_k + T\dot{z}_k + \dfrac{T^2}{2}\ddot{z}_k \end{cases} \tag{5.10}$$

The equations of the corresponding velocities $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\boldsymbol{z}})$ are:

$$\begin{cases} \dot{x}_{k+1} = \dot{x}_k + T\ddot{x}_k \\ \dot{y}_{k+1} = \dot{y}_k + T\ddot{y}_k \\ \dot{z}_{k+1} = \dot{z}_k + T\ddot{z}_k \end{cases} \tag{5.11}$$

The state-transition model is:

$$\begin{aligned} s_k &= F_k s_{k-1} + B_k u_k + w_k \\ t_k &= H_k s_k + \quad v_k \end{aligned} \tag{5.12}$$

Where:

$$\begin{aligned} s_k &= [x_k \ y_k \ z_k \ \dot{x}_k \ \dot{y}_k \ \dot{z}_k]^T \\ t_k &= [\breve{x}_k \ \breve{y}_k \ \breve{z}_k]^T \\ u_k &= [\ddot{x}_k \ \ddot{y}_k \ \ddot{z}_k]^T \\ H &= [I_3, 0_3] \end{aligned} \tag{5.13}$$

At every time-step $\boldsymbol{k}$; $\boldsymbol{s_k}$ is the estimated state of the vehicle (position and velocity); $\boldsymbol{t_k}$ is position measurement output by the sensor; $\boldsymbol{u_k}$: the acceleration of the vehicle along the three axes; $\boldsymbol{Q_k}$: the covariance of the noise process affecting the system ($\boldsymbol{w_k}$); $\boldsymbol{R_k}$: the covariance of the noise affecting the measurements ($\boldsymbol{v_k}$).

From Equations (5.10), (5.11); the state-transition matrix $\boldsymbol{F_k}$ becomes:

$$F_k = \begin{bmatrix} I_3 & TI_3 \\ 0_{3,3} & I_3 \end{bmatrix} \tag{5.14}$$

$$B_k = \begin{bmatrix} \frac{T^2}{2} I_3 \\ TI_3 \end{bmatrix} \tag{5.15}$$

5.5.2 Robust H_∞ Filter

In practice, an exact model of the system may not be available. The performance of such a system becomes an important issue. State/space

representation of the robust H_∞ filter [77] is given in Equation (5.16). In the present study, the matrices of these models are defined by Equation (5.14):

$$\begin{aligned} s_k &= (F_k + \Delta F_k)\tilde{s}_{k-1} + \ Bu_k + w_k \\ t_k &= (H_k + \Delta H_k)s_k + \ v_k \end{aligned} \tag{5.16}$$

At time-step $\boldsymbol{k}$, the two random variables $\boldsymbol{w_k}$ and $\boldsymbol{v_k}$ are uncorrelated zero-mean white noise processes with the covariance matrices $\boldsymbol{Q_k}$ and $\boldsymbol{R_k}$, respectively. The matrices $\boldsymbol{\Delta F_k}$ and $\boldsymbol{\Delta H_k}$ represent the uncertainties in system and measurements matrices. These uncertainties are assumed to be of the form:

$$\begin{bmatrix} \Delta F_k \\ \Delta H_k \end{bmatrix} = \begin{bmatrix} M_{1k} \\ M_{2k} \end{bmatrix} \ \Gamma_k N_k \tag{5.17}$$

$\boldsymbol{M_{1k}}$, $\boldsymbol{M_{2k}}$ and $\boldsymbol{N_k}$ are three known matrices, and $\boldsymbol{\Gamma_k}$ is an unknown matrix satisfying the bound:

$$\Gamma_k{}^T \Gamma_k \leq I \tag{5.18}$$

It is assumed that $\boldsymbol{F_k}$ is non-singular. This condition is verified with most physical systems. The purpose is to find an estimator of the form:

$$\tilde{s}_k = \tilde{F}_k \tilde{s}_{k-1} + \ K_k t_k \tag{5.19}$$

With the following characteristics:

- It should be stable (the eigenvalues of $\tilde{\boldsymbol{F}}_{\boldsymbol{k}}$ should be less than one in magnitude).

- Its error satisfies the following worst-case bound:

$$\max_{w_k, v_k} \frac{\|\tilde{s}_k\|_2}{\|w_k\|_2 + \|v_k\|_2 + \|\tilde{s}_0\|_{O_1^{-1}} + \|s_0\|_{O_2^{-1}}} \leq \frac{1}{\theta} \tag{5.20}$$

- Estimation error of $\tilde{\boldsymbol{s}}_{\boldsymbol{k}}$ satisfies the following RMS bound:

$$E(\tilde{s}_k \tilde{s}_k{}^T) < P_k \tag{5.21}$$

The solution of this problem can be determined with the following procedure:

1) Choose a scalar sequence $\boldsymbol{\alpha_k} > \mathbf{0}$ and a small $\boldsymbol{\varepsilon} > \mathbf{0}$.

2) Define the following matrices

$$\begin{aligned} R_{11k} &= Q_k + \alpha_k M_{1k} M_{1k}^T \\ R_{12k} &= \alpha_k M_{1k} M_{2k}^T \\ R_{22k} &= R_k + \alpha_k M_{2k} M_{2k}^T \end{aligned} \tag{5.22}$$

3) Initialise $\boldsymbol{P_k}$ and $\widetilde{\boldsymbol{P}}_{\boldsymbol{k}}$ as follows:

$$\begin{aligned} P_0 &= O_1 \\ \tilde{P}_0 &= O_2 \end{aligned} \tag{5.23}$$

$\boldsymbol{O_1}, \boldsymbol{O_2}$ are the initial values attributed to the estimation-error covariance matrices for the computation of $\boldsymbol{R_{1k}}$, $\boldsymbol{R_{2k}}$, $\boldsymbol{F_{1k}}$, $\boldsymbol{H_{1k}}$ and $\boldsymbol{T_k}$. Although these parameters have initially large values, the filter automatically tunes them within

a few iterations. As a result, the process reaches a steady state, and the error in estimation decreases to its lowest levels.

4) Find the positive definite solutions $\boldsymbol{P_k}$ and $\widetilde{\boldsymbol{P}}_{\boldsymbol{k}}$ satisfying the following Riccati equations:

$$\begin{aligned} P_{k+1} = {} & F_{1k}T_kF_{1k}^T + R_{11k} + R_{11k}R_{2k}R_{11k}^T - \\ & (F_{1k}T_kH_{1k}^T + R_{11k}R_{2k}R_{12k})R_k^{-1}(F_{1k}T_kH_{1k}^T + R_{11k}R_{2k}R_{12k})^T \\ & + \varepsilon I \end{aligned} \tag{5.24}$$

$$\begin{aligned} \tilde{P}_{k+1} = {} & F_k\tilde{P}_kF_k^T \\ & + F_k\tilde{P}_kN_k^T\left(\alpha_kI - N_k\tilde{P}_kN_k^T\right)^{-1} N_k\tilde{P}_kF_k^T + R_{11k} \\ & + \varepsilon I \end{aligned} \tag{5.25}$$

Where the matrices $\boldsymbol{R_{1k}}$, $\boldsymbol{R_{2k}}$, $\boldsymbol{F_{1k}}$, $\boldsymbol{H_{1k}}$ and $\boldsymbol{G_k}$ are defined as:

$$R_{1k} = (\tilde{P}_k^{-1} - N_k^TN_k/\alpha_k)^{-1}F_k^T \tag{5.26}$$

$$R_{2k} = R_{1k}^{-1}(\tilde{P}_k^{-1} - N_k^TN_k/\alpha_k)^{-1}R_{1k}^{-T} \tag{5.27}$$

$$F_{1k} = F_k + R_{11k}\ R_{1k}^{-1} \tag{5.28}$$

$$H_{1k} = H_k + R_{12k}^TR_{1k}^{-1} \tag{5.29}$$

$$T_k = (P_k^{-1} - \theta^2I)^{-1} \tag{5.30}$$

5) If Ricatti equation's solutions satisfy:

$$\frac{1}{\theta^2} I > P_k \tag{5.31}$$

$$\alpha_k I > N_k \tilde{P}_k N_k^T \tag{5.32}$$

Equation (5.19) solves the problem with:

$$K_k = (F_{1k} T_k H_{1k}^T + R_{11k} R_{2k} R_{12k}) \widetilde{R}_k^{-1} \tag{5.33}$$

$$\widetilde{R}_k = H_{1k} T_k H_{1k}^T + R_{12k}^T R_{2k} R_{12k} + R_{22k} \tag{5.34}$$

$$\tilde{F}_k = F_{1k} - K_k H_{1k} \tag{5.35}$$

The parameter $\boldsymbol{\varepsilon}$ is generally chosen as a very small positive number. In the present study it was set to $\boldsymbol{\varepsilon = 10^{-8}}$.
The parameter $\boldsymbol{\alpha_k}$ must be chosen large enough so that the conditions of Equations (5.31), (5.32) are satisfied. However, when $\boldsymbol{\alpha_k}$ increases, $\boldsymbol{P_k}$ also increases, which results in a looser bound for the RMS estimation error [77].

A steady-state of the robust filter can be obtained by letting the parameter $\boldsymbol{P_{k+1} = P_k}$ and $\boldsymbol{\tilde{P}_{k+1} = \tilde{P}_k}$ in Equation (5.25). Tracking results delivered by the Kalman filter are compared with the trajectory filtered with the Robust H_∞. The application of this filter results in a more robust tracker.

The adaptation of the Robust H_∞ filtering scheme is proven to be flexible and capable of producing an accurate state estimation based on uncertain system's parameters. This asset allows the tracking of vehicles without an exact knowledge of their motion model. The Robust H_∞ filter exhibited very interesting results in several automation and control applications [125] [126]. More importantly, the results obtained from the experiments and the error in estimation compared to ground truth measurements show the effectiveness of the current approach against the naïve filtering scheme (Kalman filter does not consider the uncertainties in the system). Nevertheless, if the exact model is

available, the Robust H_∞ filter performs even better. However, in many real cases the exact model remains hard to determine [127]. The Robust filter combines the robustness of H_∞ and the optimality of Kalman filtering.

5.5.3 Difference Between H_∞, Kalman and the Robust H_∞

H_∞ filter [77] minimises the error cost-function in the L_∞ norm sense. In other words, it bounds the worst case or the maximum possible steady state estimation-error $\boldsymbol{J}_\infty$ given by:

$$J_\infty = \lim_{N\to\infty} \quad \max_{s_0,w_k,v_k} \frac{\sum_{k=0}^{N}\|s_k - \tilde{s}_k\|^2}{\|s(0)-\tilde{s}(0)\|^2_{P_0^{-1}} + \sum_{k=0}^{N}\left(\|w_k\|^2_{Q_k^{-1}} + \|v_k\|^2_{R_k^{-1}}\right)} \tag{5.36}$$

$$J_\infty < \frac{1}{\theta} \tag{5.37}$$

The value of $\boldsymbol{\theta}$ determines how loose the bound is, Equation (5.37). Thus, how large one can tolerate the maximum magnitude of estimation-error.

On the other hand, Kalman filter optimises the L_2 (least squares norm or the RMS of estimation-error). Its error function $\boldsymbol{J_2}$ is given by:

$$J_2 = \lim_{N\to\infty} \quad \sum_{k=0}^{N} E(\|s_k - \tilde{s}_k\|_2) \tag{5.38}$$

In both filters, it is assumed that the parameters (states-transition matrix, state/measurements projection matrix) have been accurately determined in advance. However, this is very unlikely in practice. The Robust H_∞ filter combines the advantages of Kalman filter and H_∞ together; i.e. it has the ability to minimise the overall RMS, as well as to bound the worst case error. More importantly, the Robust H_∞ is also useful when the parameters of the system are imprecise. In the present study (tracking of moving objects), exact knowledge about the dynamics of the vehicle are not available. The native H_∞ (without uncertainty handling) is as good as a Kalman filter provided that the

noise processes are white Gaussian. Nevertheless, when the noise process is not Gaussian, H_∞ is better endowed to keep the magnitude of error below a particular threshold. The Robust H_∞ offers the advantages of H_∞, with the extra benefit of the minimisation of the global RMS error.

5.6 Tracking Data Fusion

The drop in precision of some tracking measurements can be significant, particularly when the target moves far from the cameras. In addition, when there are many targets moving around the obstacles in the same scene, occlusions can appear and consequently prohibit the correct recognition of the vehicles. At this level in the pipeline, the cooperation is established between multiple Kinects with the application of covariance intersection filter [115]. The estimated positions delivered by all cameras are merged (after being filtered by the robust H_∞) into one consistent estimate that precisely determines the pose of the vehicle in its space.

5.6.1 Covariance Intersection Filtering

Based on the estimates determined by the Robust H_∞ filter $\hat{\boldsymbol{x}}_{\boldsymbol{kn}}$ $(\mathbf{1} \leq \boldsymbol{n} \leq \boldsymbol{N})$ at time step $\boldsymbol{k}$, and their respective covariance matrices $\boldsymbol{P_{kn}}$; a joint estimate $\tilde{\boldsymbol{x}}$ is computed along with its error covariance $\boldsymbol{P}$ where the true state of the system is $\boldsymbol{x}$ (the true position of the vehicle).

If $\boldsymbol{N} > \mathbf{1}$ is the number of unbiased estimates delivered by all the cameras $\widehat{\boldsymbol{x}}_{\mathbf{1}}, \widehat{\boldsymbol{x}}_{\mathbf{2}}, \widehat{\boldsymbol{x}}_{\mathbf{3}} \ldots . \widehat{\boldsymbol{x}}_{\boldsymbol{N}}$ for the unknown state vector $\widetilde{\boldsymbol{x}}$:

$$\widetilde{x} = P \sum_{n=1}^{N} P_n^{-1} \widehat{x}_n \tag{5.39}$$

$$P^{-1} = \sum_{n=1}^{N} P_n^{-1} \tag{5.40}$$

When there exists some correlation between estimation errors, $\boldsymbol{P}$ may become far too optimistic and this may lead to a divergence in the sequential filtering. A conservative estimate can be given by applying covariance intersection according to Equations (5.41) and (5.42) :

$$\widetilde{x} = P \sum_{n=1}^{N} \omega_n P_n^{-1} \widehat{x}_n \tag{5.41}$$

$$P^{-1} = \sum_{n=1}^{N} \omega_n P_n^{-1} \tag{5.42}$$

With the nonnegative coefficients $\boldsymbol{\omega_n}$ verifying the following consistency condition:

$$\sum_{n=1}^{N} \omega_n = 1 \tag{5.43}$$

An estimate can always be obtained with:

$$P \geq P_0 := E[(\widetilde{x} - x)(\widetilde{x} - x)^T] \tag{5.44}$$

Where $\boldsymbol{P} \geq \boldsymbol{P_0}$ denotes the fact that $\boldsymbol{P} - \boldsymbol{P_0}$ is positive semi-definite. As a result, the coefficients $\boldsymbol{\omega_n}$ should minimise either the trace or the determinant of $\boldsymbol{P}$.

In order to avoid the possibly high numerical effort to find a solution to this nonlinear optimisation problem, Neihsen [115] proposed a fast approximate solution following this reasoning:

For $\boldsymbol{tr}(\boldsymbol{P_n}) \leq \boldsymbol{tr}(\boldsymbol{P_m}); \boldsymbol{1} \leq \boldsymbol{n}, \boldsymbol{m} \leq \boldsymbol{N}$; $\boldsymbol{tr}$ $(\boldsymbol{U})$ refers to the trace of the matrix $\boldsymbol{U}$, one would expect $\boldsymbol{\omega_n} \geq \boldsymbol{\omega_m}$. For the purpose of decreasing the computational load, instead of using the estimation uncertainty $\boldsymbol{P_n}$, the authors in [123] introduced estimation certainty by considering $\boldsymbol{S_n} = \boldsymbol{P_n^{-1}}$ resulting in:

$$\omega_n = \frac{tr(S_n)}{\sum_{i=1}^{N} tr(S_i)} \tag{5.45}$$

Equation (5.45) means that the greater $\boldsymbol{tr(S_n)}$ is: the more certain the estimate $\widehat{\boldsymbol{x}}_{\boldsymbol{n}}$, the higher the corresponding weighting $\boldsymbol{\omega_n}$. Conversely, the smaller $\boldsymbol{tr(S_n)}$, the lower the weighting $\boldsymbol{\omega_n}$ becomes. More importantly, consistency condition of Equation (5.43) remains satisfied:

$$\sum_{n=1}^{N} \omega_n = \frac{\sum_{n=1}^{N} tr(S_n)}{\sum_{i=1}^{N} tr(S_i)} = 1 \tag{5.46}$$

5.6.2 Covariance Intersection for Multikinect Tracking

As has been shown in Chapter 4, some Kinects may be worn. Hence, they produce erroneous measurements when the target is far away from the sensor. However, with a cooperative multiview setup, the global position and orientation estimate can be cooperatively corrected. The weighting coefficients lead this correction by attributing a greater weight to the more accurate single-camera estimates in the multiview setup covering the scene.

Another contribution of the present chapter is the adaptive weighting scheme based on the assessment of the quality regarding each estimate resulting from the Robust H_∞ filter and the confidence in the raw measurement delivered by the camera itself. Indeed, a quality factor is attributed to each camera participating in the capture process. This indicator is obtained from the residual error after applying the appropriate correction model. The mathematical formulation is as follows: For the processing thread corresponding to the $\boldsymbol{n}$-th camera:

- $\boldsymbol{P_n}$: is the covariance matrix of the error in the estimate delivered by the Robust H_∞ filter.

- $\boldsymbol{D_n}$: is the covariance matrix characterising the residual error after correction.

- $\boldsymbol{Z_n}$: is a positive scalar that represents the distance between the target and the camera.

The last assumption is inspired by the fact that the smaller the depth of the target, the more accurate the resulting measurement.

Figure 5.9 describes a situation where every sensor has its native hardware accuracy matrix $\boldsymbol{D_n}$ (Red circles). In addition, based on the distance separating the cameras from the target, weighting coefficient $\boldsymbol{Z_n}$ is introduced. For every pair of position-estimates $\widehat{\boldsymbol{x}}_{\boldsymbol{n}}, \widehat{\boldsymbol{x}}_{\boldsymbol{m}}$ the following condition should be satisfied:

$$\begin{aligned} &tr(D_n) + tr(P_n) + Z_n \leq tr(D_m) + tr(P_m) + Z_m \Rightarrow \ \omega_n \geq \omega_m \ ; \\ &1 \leq n, m \leq N \end{aligned} \tag{5.47}$$

Equation (5.47) represents the fact that $\widehat{\boldsymbol{x}}_{\boldsymbol{n}}$ affects the final estimate $\widetilde{\boldsymbol{x}}$ more than $\widehat{\boldsymbol{x}}_{\boldsymbol{m}}$ does. In addition, $\boldsymbol{P_n}$ affects the final error in estimation $\boldsymbol{P}$ more than $\boldsymbol{P_m}$ does. In the tracking algorithm, Niehsen's [115] findings about the fast covariance intersection are considered. In addition, the uncertainty characterising the quality of the measurements delivered by the sensor is defined. The weightings are given by the equation below:

$$\omega_n = \frac{1}{N-1} \ \frac{\sum_{\substack{i=1 \\ i \neq n}}^{N} (tr(D_i) + tr(P_i) + Z_i)}{\sum_{i=1}^{N} (tr(D_i) + tr(P_i) + Z_i)} \tag{5.48}$$

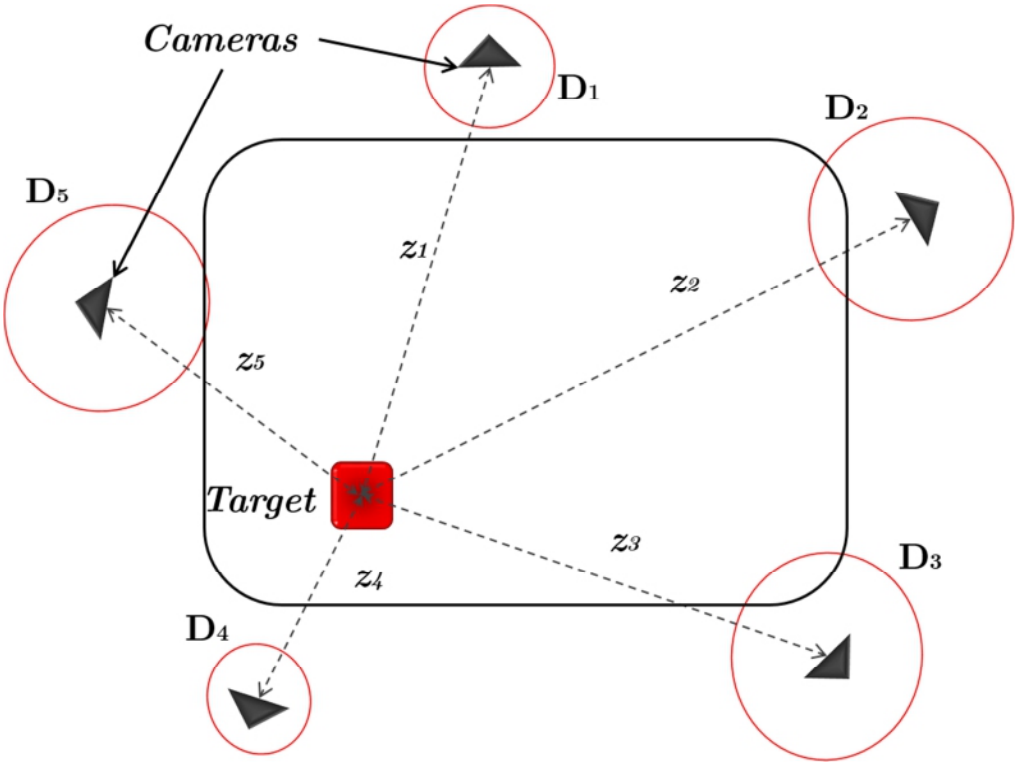


Figure 5.9 Covariance Intersection parameters

Another form of the same expression, which is more suitable to reduce the load of computation is given by:

$$\omega_n = \frac{1}{N-1} \frac{\sum_{i=1}^{N}(tr(D_i) + tr(P_i)+Z_i) - (tr(D_n) + tr(P_n)+Z_n)}{\sum_{i=1}^{N}(tr(D_i) + tr(P_i)+Z_i)} \tag{5.49}$$

At this level, it would be just necessary to compute the traces of the matrices once. The denominator $\sum_{i=1}^{N}(\boldsymbol{tr(D_i)} + \boldsymbol{tr(P_i)}+\boldsymbol{Z_i})$ stays the same for all the estimates. Hence, it is also computed once. Afterwards, the appropriate parameter $(\boldsymbol{tr(D_n)} + \boldsymbol{tr(P_n)}+\boldsymbol{Z_n})$ corresponding to each estimate is subtracted from the denominator. The condition of consistency of Equation (5.43) remains verified because $\sum_{n=1}^{N}\boldsymbol{\omega_n} = \mathbf{1}$. The conducted experiments using this approach proved that Equation (5.49) is the most realistic and suitable for the tracking scenario.

5.7 Orientation Computation

Until this point, the calculation of the direction towards which the robot is heading has not been involved in the estimation process. However, the current solution is already able to deliver the 6 DOF regarding the tracked entity without any need to process orientation data separately. In the published literature, the conventional way to compute the orientation during the motion of the vehicle is the well-known Least Squares algorithm. In this category of solutions, the optimiser iterates over all the elementary rotations and translations before reaching the correct pose of the robot. For this reason, they are not suitable for real-time solutions.

If the nine entries of the rotation matrix are added to the state vector, the size of the latter grows to accommodate 12 components (3 for translation and 9 for rotation). The corresponding matrices will scale to up to 12×12 elements, which could not be processed by dense alignment algorithms at more than 8 FPS for the five sensors. As a result, the little improvement in the accuracy of estimation is not worth the time taken by the processing. One may think of the rotation being a rank three matrix. However, the independent components (three $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ angles in the 3D space) cannot be applied directly on point data. In addition, they need to be converted into a rotation matrix to test how far the estimate is from the optimal value.

Some accuracy is traded for computation time as shown in the results. This choice is driven by the fact that when the accurate position data for the three markers is available, it becomes possible to efficiently compute the rotation matrix describing the pose of the vehicle (see Figure 5.10 (a), (b)) in the scene relative to the world frame, Figure 5.11.

On the other hand, the accurate estimate of the position of the vehicle can be used to compute the remaining three orientation components $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$. Such a procedure requires only some simple trigonometry to be applied on the accurate position of the centroid and the frontal marker, Figure 5.11 (a), (b). This

method is effective because it avoids complicating the filter with the additional load of computations that ultimately leads to almost the same result.

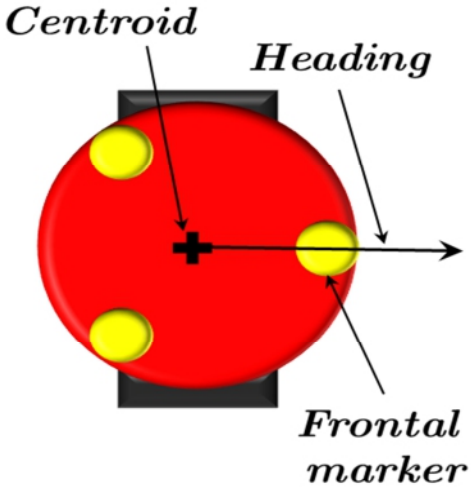
Figure 5.11 (c), (d) depicts how the determination of the orientations can be formulated. From the Equations (5.51) to (5.53), one can directly obtain the three angles that define the aspect of the vehicle in the scene. In addition, it is possible to compute the 3D rotation between two different orientations by only using the angles characterising the two poses.

$$Mag = \sqrt{(x - x')^2 + (y - y')^2 + (z - z')^2} \tag{5.50}$$

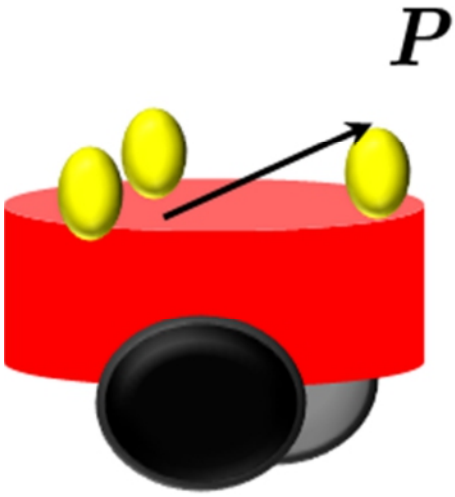
$$\alpha = arcos\left(\frac{x - x'}{Mag}\right) \tag{5.51}$$

$$\beta = arcos\left(\frac{y - y'}{Mag}\right) \tag{5.52}$$

$$\gamma = arcos\left(\frac{z - z'}{Mag}\right) \tag{5.53}$$

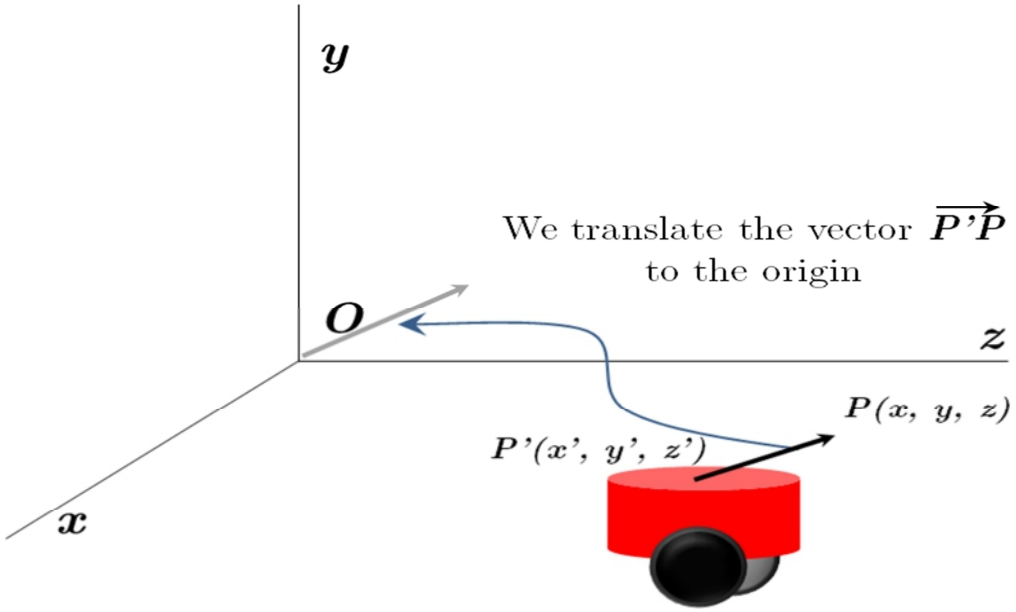


(a)

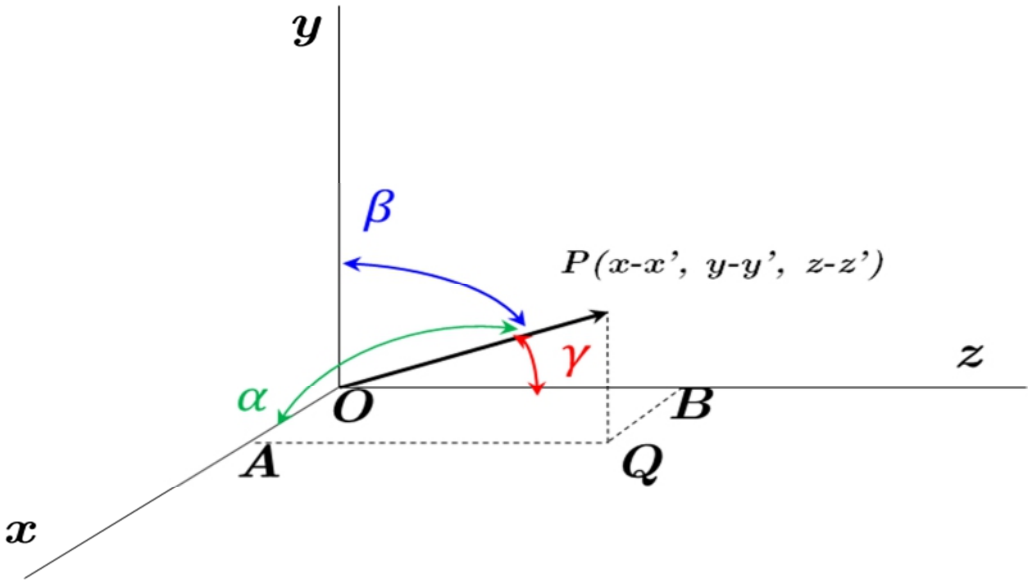


(b)

Figure 5.10 Markers' structure. **(a)** Heading vector (upper view). **(b)** Heading vector (side view)



(a)



(b)

Figure 5.11 Orientation computation. **(a)** Pose of the robot in the scene. **(b)** Orientation definition

5.7.1 Special Case

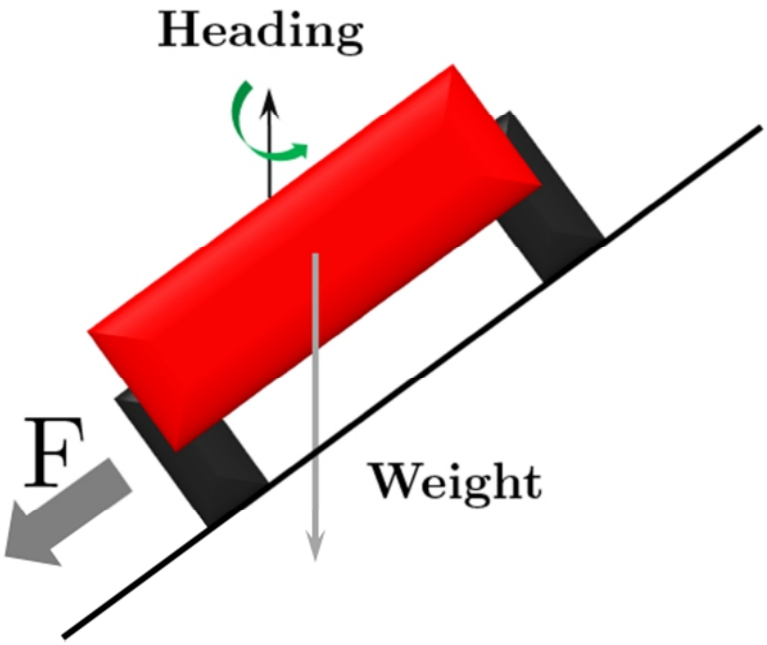
The presented approach to compute the pose of the vehicle in the scene is not yet complete. This is due to the fact that it gives the same orientation angles when the robot rotates around its heading vector. As a result, the poses taken by the vehicle are obviously different but their respective heading vectors are similar.

The use of the whole rotation matrix was avoided in the filtering scheme. This is because it would not be possible to track the vehicle at a high frame rate (author's experiments showed a maximum of 8 FPS with all coding optimisations taken into account). However, it has been already claimed in the introduction to this chapter that the purpose is the design of a real-time system running at 25 FPS, and the ability to describe the 6 DOF for the vehicle in the scene. The delivery of the rotational data in real-time can be possible if the vehicle is equipped with an IMU. The latter is able to stream the accelerations of the vehicle and its angular velocities at a high frame rate. The problem with this kind of sensor is the drift over time [128]. Thus, another source of pose estimation is necessary to eliminate the drift periodically.

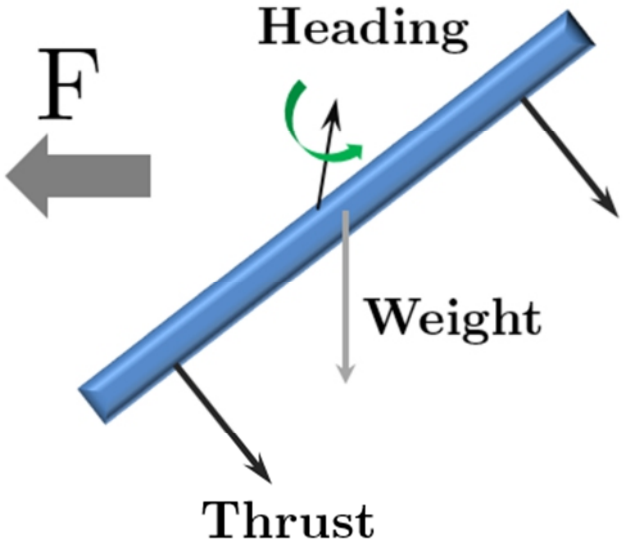
In practice, it is very unlikely for the vehicle to rotate around its heading vector. This fact is verified by the physical constraints on its dynamics and mechanical structure. For ground vehicles, there could be a high risk of slip if the robot is driven to rotate around the axis passing through its heading vector, Figure 5.12 (a). On the other hand, for aerial vehicles there will be some difficulty encountered by the vehicle in lifting itself when rotating around its heading, Figure 5.12 (b). The latter would lose equilibrium and consequently crash into the ground.

For this reason, another sparse filtering-based registration approach will be contributed in the next chapter to remedy this special case as well as processing time requirement. Based on some feature data taken from the scene, the last

mentioned computes successfully the rigid-body motion undergone between successive frames.



(a)



(b)

Figure 5.12 The behaviour of the vehicles under the effect of the different forces when they rotate around their heading vectors. **(a)** Ground vehicles (UGV). **(b)** Aerial vehicles (UAV)

5.8 Results & Discussions

All the following stages are based on the same hardware configuration presented in Chapter 3. For programming, C++ was used for the CPU and CUDA for the GPU/CPU heterogeneous coding.

5.8.1 Synchronisation

5.8.1.1 Kinects/Ground Truth

The key to synchronisation between ground truth measurements (OptiTrack[6]) and experimental data in the solution is the clock signal. In other words, the two threads (position estimation and the ground truth capture) run on the same machine. Thus, they share the same clock signal. In addition, the frequency of capture concerning the OptiTrack system is set to 120 FPS, which means it is more than four times higher than the native frame rate of the estimation pipeline (25 FPS). To handle concurrency between the two threads, a multithreaded solution synchronised by the clock signal is, therefore, proposed. For instance, at the reception of every new pair of positions (estimated, ground truth) the time of capture is saved. Afterwards, another algorithm matches the currently delivered position estimate with the set of the freshly acquired ground truth measurements. Knowing that the rate of capture is not the same, it results in at least four ground truth measurements for each estimated position. The closest in time is chosen (the frame with the smallest time difference with the instant of capture regarding the estimated position). Algorithm 5.1 provides the details of how the synchronisation works.

5.8.1.2 Between the Kinects

Although all the sensors are similar, their respective rates of capture are not synchronised. Algorithm 5.2 depicts the procedure of synchronisation between the flows of data streamed by the Kinects in the multiview setup. It also shows

[6] https://www.naturalpoint.com/optitrack/. 2015

the synchronisation between the capture, of all five Kinects, and the fusion module.

Algorithm 5.1 Synchronisation OptiTrack/Kinects

```
posData;
// an array of six pose data (x, y, z, ox, oy, oz)
f;
//an Optitrack frame containing the position and the orientation of
the rigid body (robot)
ft;
//an Optitrack frame to which is added the time information to find
the correspondence with tracker's data
OptiTrackThread()
{
...
    while (true)
     {
     ...
        EmptyArray (posData);
        for (i=0; i<4; i++)// for all five cameras
        {
            getFrame(f);
            addTimeToPoseData(ft,f);
            posData = <posData, ft>;
        }
        sendPoseDataToTracker(posData);
     ...
     }
...
}


TrackerThread()
{
...
    while (true)
     {
     ...
         if (newFrame)
         {
            bestMatchIndex = LookForTheBestMatch (posData,
            frameTime);
            //fetch the ground truth frame that better fits the one
            //delivered by the Kinects and discard the remaining
            //ones
            saveMeasureAndGt(posData, bestMatchIndex, frameTime);
            //save the pair <pose, ground truth>
         }
     ...
     }
...
}
```

Algorithm 5.2 Synchronisation between the Kinects

```
CamNum=5;// this variable indicates  the number of cameras
Buff;// array of 3D position data. It is a global variable pointing to the
memory space where is kept the position data issued by each camera.
NumIndex; // Index of a given thread corresponding to one camera
nbFilled=0;// nbFilled < CamNum; represents the number of the already ac-
quired frames
semaphore Mutex=1; //semaphore to protect the critical section when updat-
ing nbFilled
semaphore Empty=1;// indicates whether all the sensor-wise position data is
ready
semaphore Filled=0;// indicates if the fusion of the available data has
completed

CaptureThread (CamIndex)
{
....
    while (true)
    {
        // test whether a new frame has just arrived
        if ( NewFrame(CamIndex))
        {
          //lock the section of code which updates the variable
            nbFilled
            Down(Mutex);
            nbFilled++;
            if(nbFilled >= CamNum)
            {
                Down(Empty);
                Up(Filled);
            }
            //compute the 3d position of a given robot in
            //the scene from the depth image embedded in the current
            //frame
            buff(CamIndex) = poseData(NewFrame(CamIndex));
            Up(Mutex);

        }
    }
....
}

Fusion()
{
...
    while (true)
    {
    Down (Filled);
    ...
    //position data Fusion algorithm
    ...
    //Proceed through the actual fusion
    ...
    //after reading all necessary data the
    //acquisition is launched again for all five cameras
    nbFilled = 0;
    Up(Empty);
    }
...
}
```

The algorithm (Chapter 2) needs the following three semaphores:

- *Mutex* used to update the number of the available positions (**NbFilled**).
- *Empty* is set to one when the Fusion finishes.
- *Filled* is set to one when all the threads supervising the capture finish acquiring the image data (*NbFilled = 5*).

5.8.2 GPU Capture and Marker Extraction Algorithms

The subject images that are being processed have a VGA resolution (640×480) delivered at the same frame rate as the camera to allow the following applications to exploit fully the frame rate offered by the sensor. When the correction was first run on a regular CPU, the maximum achieved frame rate was 15 FPS. Hence, arises the need to implement the bottlenecks of the solution in the GPU. In addition, image data is more naturally organised to fit GPU blocks, where every element in the block (thread) processes a single pixel at a time [80]. Figure 5.13 illustrates how the depth image can be embedded in the GPU and the steps of processing in the kernel. The same architecture as that used in Chapter 3. The image is divided into blocks of a constant size (16×16 pixels; so 256 threads is the size of a single block). Pixels of the same block are processed simultaneously in the same thread block. As a result, for every pixel in the image is attributed a thread in the GPU. The latter is responsible for all the processing related to that pixel (capture, correction and markers extraction).

From a computational point of view, there are no iterations in the kernel of the filter. The optimisation applies to the three components of position ($\boldsymbol{x_k}, \boldsymbol{y_k}, \boldsymbol{z_k}$). Velocity components in the state vector are just needed to propagate the estimation to the next time-step. Hence, they are not considered in all the matrices. Most of these matrices are 3×3.

This solution does not have any problems in scaling up to a reasonable level. Processing bottlenecks were implemented in the GPU (image data capture and

markers extraction). Such an architecture allows a simultaneous processing of all the pixels. The results are validated with five sensors to show the real-time capability of the filtering algorithm with large images ($5 \times 2 \times 640 \times 480$ pixels processed at 25 FPS). Nevertheless, structured light cameras still suffer from interference when they are used altogether (Chapter 2).

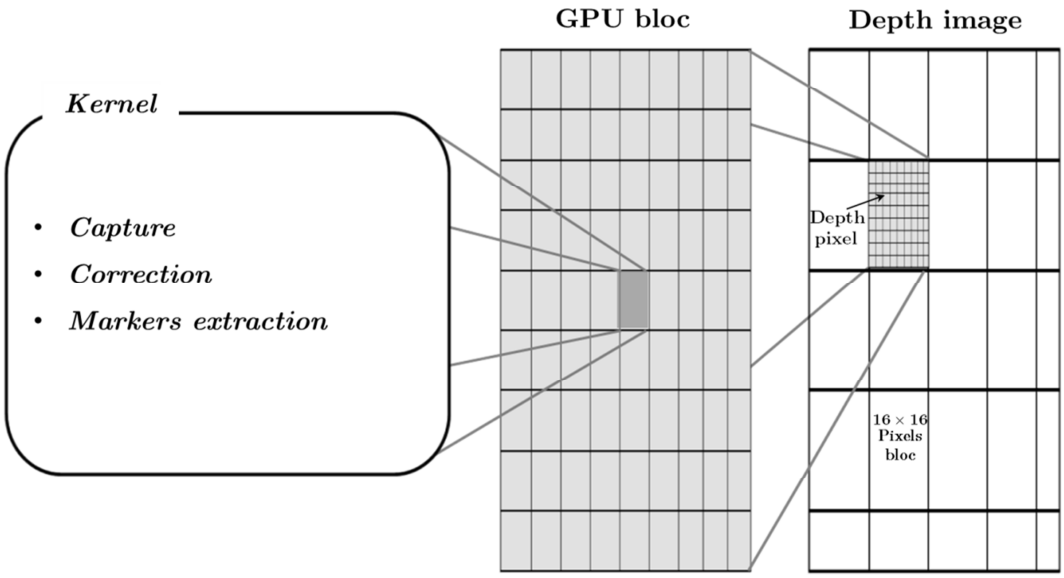


Figure 5.13 Kalman filter GPU implementation for depth map filtering

The CPU and the GPU are significantly different. A GPU can handle large amounts of data in many streams, performing relatively simple operations, but it is inadequate for massive processing on a single stream. A CPU is much faster for a per-core treatment and can, therefore, perform complex operations on a single stream of data more quickly. Consequently, the robust filter and the covariance intersection algorithms have not been implemented in the GPU. The reason is that the complexity of these algorithms is not proportional to the size of images. They just refine the 3D positions of the markers. It was possible to filter the five position data using a CPU-based multithreaded architecture where each thread handles the stream of a given camera. The fusion algorithm is then executed on the resulting estimates to find the correct pose of the vehicle.

5.8.3 Robust H_∞ Filter

The results discussed in the following sections are obtained from the setup shown in Figure 5.14.

Figure 5.15 and Figure 5.17 present respectively the best and the worst performances of both the Robust H_∞ (RF) tracking algorithm as well as the Kalman filter (KF). As shown in Section 5.5, the motion model of the robot is unknown. Hence, a generic Newtonian system is assumed to mimic the displacements regarding the vehicle. Uncertainties are controlled with RF.

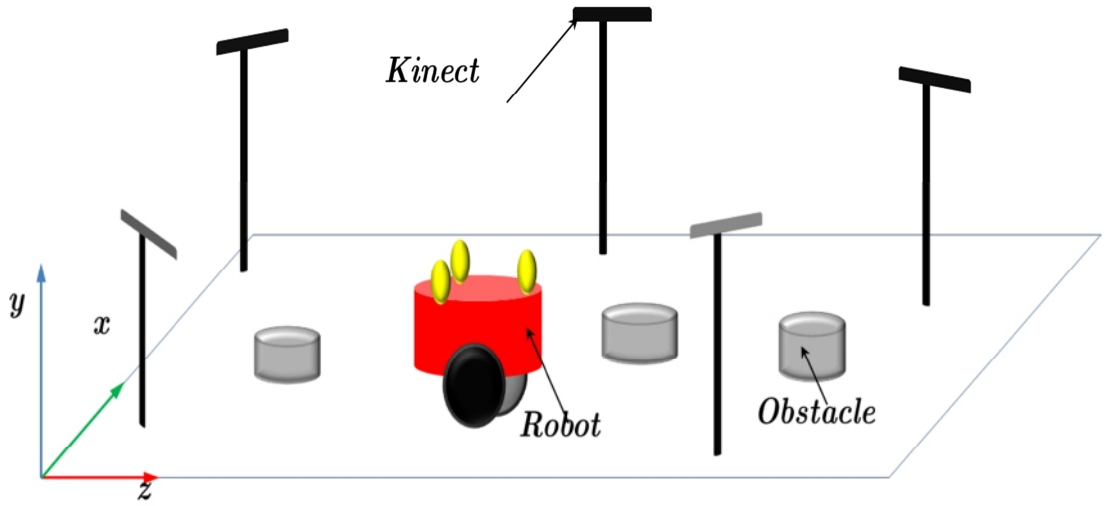


Figure 5.14 Experimental setup

$\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$-axes are shown in Figure 5.14. For $\boldsymbol{x}$ and $\boldsymbol{y}$ coordinates, Figure 5.15 and Figure 5.17 show almost similar shapes for the error graphs regarding the two filters. Such a resemblance appears because of the similarity of the model of motion for both algorithms. Nevertheless, tracking error of the Robust H_∞ is smaller. The smallest error among all five cameras for $\boldsymbol{x}$ coordinate was 0.0403m with RF against 0.0546m for KF, Figure 5.15 (a). The worst case in $\boldsymbol{x}$ was 0.0450m for RF against 0.0580m for KF, Figure 5.17 (a). For $\boldsymbol{y}$ coordinate, the best RMSE was 0.0252m with RF against 0.040m for the KF, Figure 5.15 (b). The worst case in $\boldsymbol{y}$ was 0.0253m with RF against 0.0429m for KF, Figure 5.17(b).

For z component, the best result with RF was 0.0493m against 0.064m for Kalman filter, Figure 5.15 (c). The worst result was 0.053m with RF against 0.0634m for KF, Figure 5.17 (c).

Throughout the experiments, the RF was the least affected by the inaccuracies in the parameters of the system. It always gives the best estimation. More importantly, it was capable of predicting the position of the moving robot even when no measurements were available. The detailed results for all the five sensors are given in Table 5.1, Table 5.2 and Table 5.3. The results shown in these tables are achieved after the correction of all cameras with their respective models. On the other hand, the effectiveness of some sensors against others is profoundly influenced by the shape of the trajectory followed by the vehicle. If all the cameras have the same precision of depth sensing, the closest one to the robot will be the best candidate to capture the position precisely.

x-RMSE(m)	Kinect_0	Kinect_1	Kinect_2	Kinect_3	Kinect_4
KF	0.0570	0.0575	0.0580	0.0552	0.0546
RF	0.0433	0.0435	0.0456	0.0440	0.0403
Difference(KF-RF)	0.0137	0.0140	0.0124	0.0112	0.0143

Table 5.1 Error in x component for all cameras

y-RMSE(m)	Kinect_0	Kinect_1	Kinect_2	Kinect_3	Kinect_4
KF	0.0392	0.0429	0.0414	0.0415	0.0415
RF	0.0253	0.0253	0.0253	0.0253	0.0252
Difference(KF-RF)	0.0139	0.0176	0.0161	0.0162	0.0163

Table 5.2 Error in y component for all cameras

z-RMSE(m)	Kinect_0	Kinect_1	Kinect_2	Kinect_3	Kinect_4
KF	0.0614	0.0594	0.0634	0.0590	0.0640
RF	0.0537	0.0524	0.0530	0.0534	0.0493
Difference(KF-RF)	0.0077	0.0070	0.0104	0.0056	0.0147

Table 5.3 Error in z component for all cameras

5.8.4 Covariance Intersection

At the final stage of the cooperative multiview tracking pipeline, the Covariance Intersection filter (CI) has been adopted to fuse the position data of the sensor-wise estimates. To validate the findings about the CI weighting coefficients, three different approaches of applying these weightings to the estimates have been compared. The weighting with only the error in estimation ($\boldsymbol{P_n}$) obtained from RF was first tested. Then the pair $\boldsymbol{P_n}$ with the uncertainty in the accuracy of the sensor ($\boldsymbol{P_n}, \boldsymbol{D_n}$). Finally, the combination of both previous parameters with the confidence in the depth measurement ($\boldsymbol{P_n}, \boldsymbol{D_n}, \boldsymbol{Z_n}$). After considering each new parameter in the weighting of data fusion, the quality of the estimation was improved. The CI algorithm has been tested on the estimates of the trajectory resulting from the Kalman filter (CI + KF) and the one obtained from the robust H_∞ filter (CI + RF). The results of the fused trajectories were as follows:

Firstly, for the $\boldsymbol{x}$ coordinate with $\boldsymbol{P_n}$ on its own, Figure 5.19 (a), Table 5.4 (col $\boldsymbol{x}$), gave an error of 0.028m with RF, whereas with KF it gave 0.0415m. After considering the accuracy of the sensor, Figure 5.21 (a), Table 5.5 (col $\boldsymbol{x}$), the error decreased for both filters to reach 0.0188m with RF, and 0.028m for KF. The introduction of $\boldsymbol{Z_n}$, Figure 5.23 (a), Table 5.6 (col $\boldsymbol{x}$), further approached the estimation to its ground truth value as the error reached 0.011m with RF and 0.016m for KF.

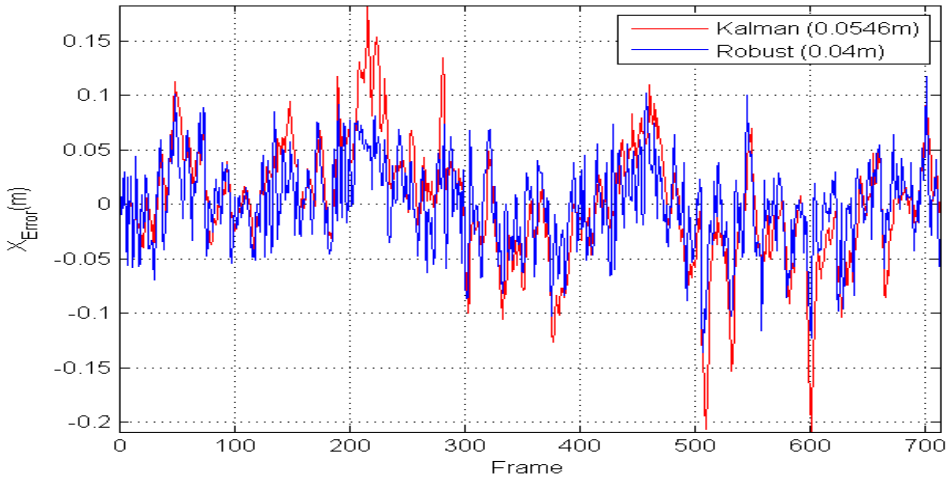
Secondly, for the $\boldsymbol{y}$ coordinate $\boldsymbol{P_n}$ on its own, Figure 5.19 (b), Table 5.4 (col $\boldsymbol{y}$), gave again an error of 0.0154m for RF, whereas with KF it gave 0.0247m. After

adding the accuracy parameter regarding the sensor, the error fell down to 0.011m with the RF. Likewise, the introduction of $\boldsymbol{D_n}$, Figure 5.21 (b), Table 5.5 (col $\boldsymbol{y}$), clearly improved the accuracy of KF estimation. The error decreased to 0.017m. Lastly, after including $\boldsymbol{Z_n}$ in the weighting, Figure 5.23 (b), Table 5.6 (col $\boldsymbol{y}$), the error in the estimation fell to 0.006m, at the same time the error with KF decreased to 0.0098m.

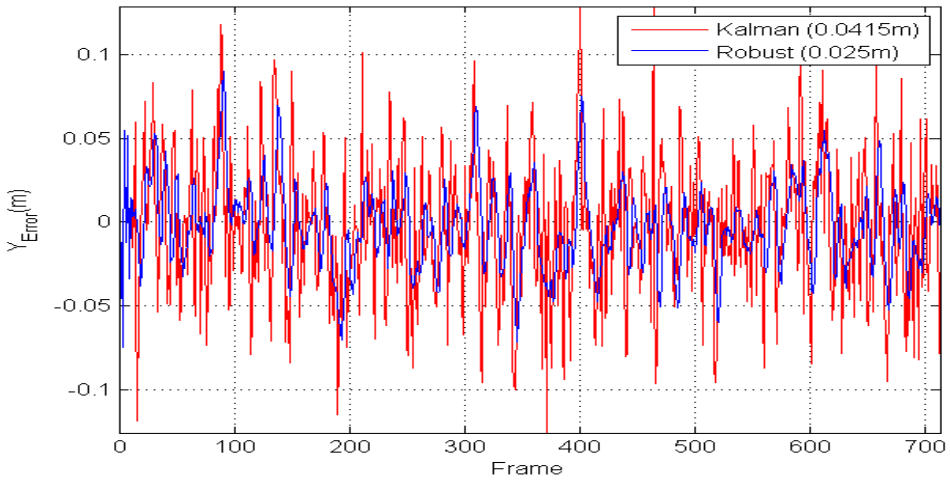
Thirdly, for the $\boldsymbol{z}$ component $\boldsymbol{P_n}$, Figure 5.19 (c), Table 5.4 (col $\boldsymbol{z}$), on its own gave again an error of 0.0323m for RF, whereas with KF it gave 0.0484m. After adding the accuracy parameter of the sensor, the error was slightly reduced to 0.0223m with RF. Similarly, the introduction of $\boldsymbol{D_n}$, Figure 5.21 (c), Table 5.5 (col $\boldsymbol{z}$), improved the accuracy of KF estimates as the error dropped to 0.0333m.

The introduction of $\boldsymbol{Z_n}$, Figure 5.23 (c), Table 5.6 (col $\boldsymbol{z}$), positively affected the error in the RF, which reached 0.013m. The error in KF estimation also decreased to 0.0195m.

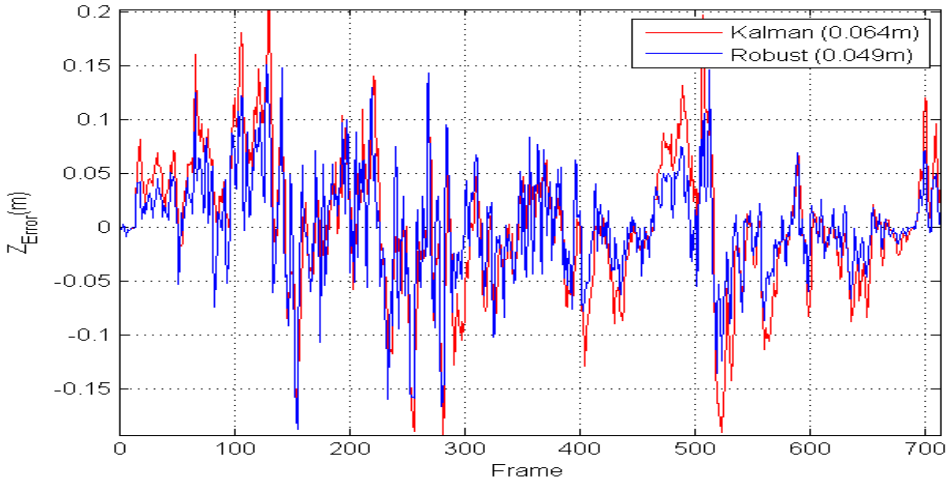
Based on the results obtained from these experiments, the quality of the estimation between RF and KF is not significantly different because of the compensating effect of the CI algorithm over all the sensors. In other words, each sensor contributes its best estimation. After correction, the most accurate measurement is used in both KF and RF to compute the next prediction. As a consequence, the difference in the fused output is not significant when a higher weighting is attributed to the most reliable measurement. In addition, given the limited space used for this indoor experiment, the RF theoretically performs as well as the KF when the robot moves linearly following the predefined motion model.



(a)

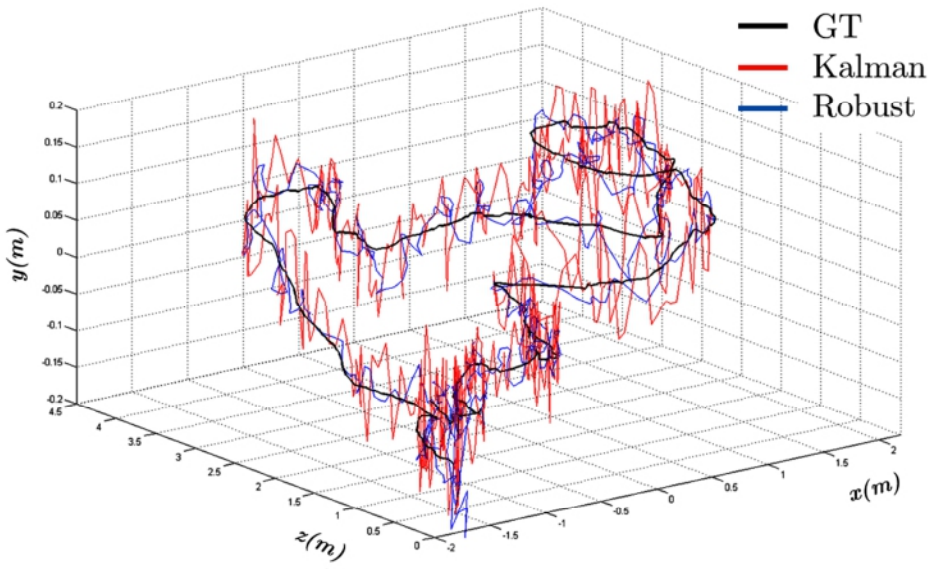


(b)

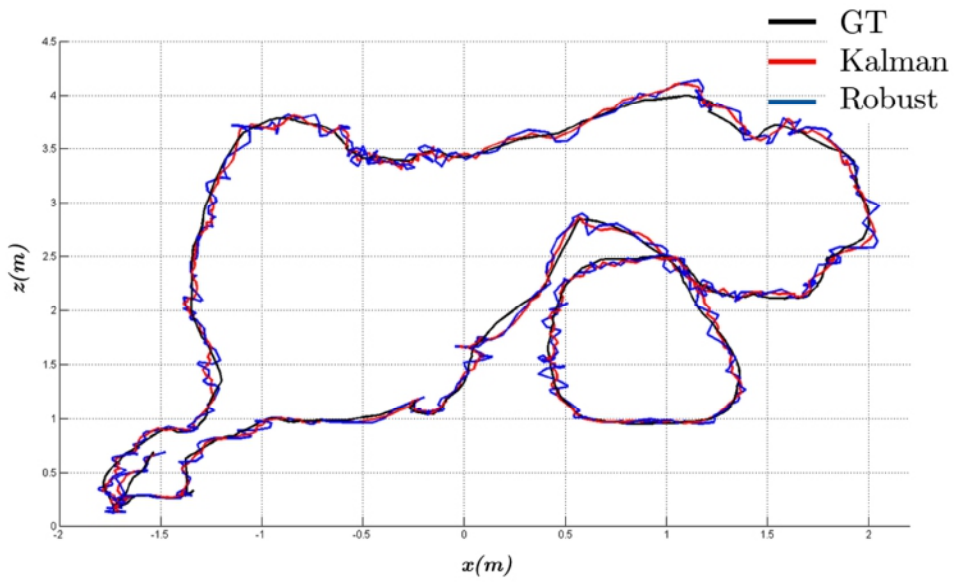


(c)

Figure 5.15 The best monoview tracking RMSE (a) $\boldsymbol{x}$. (b) $\boldsymbol{y}$. (c) $\boldsymbol{z}$.

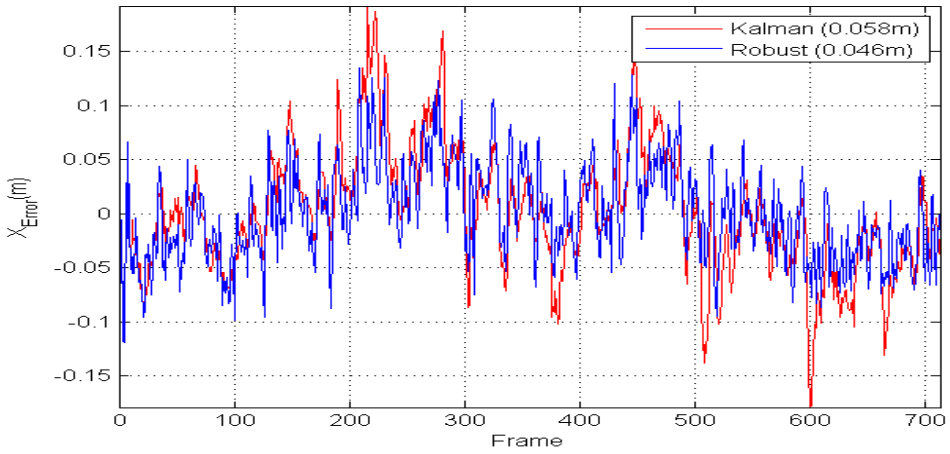


(a)

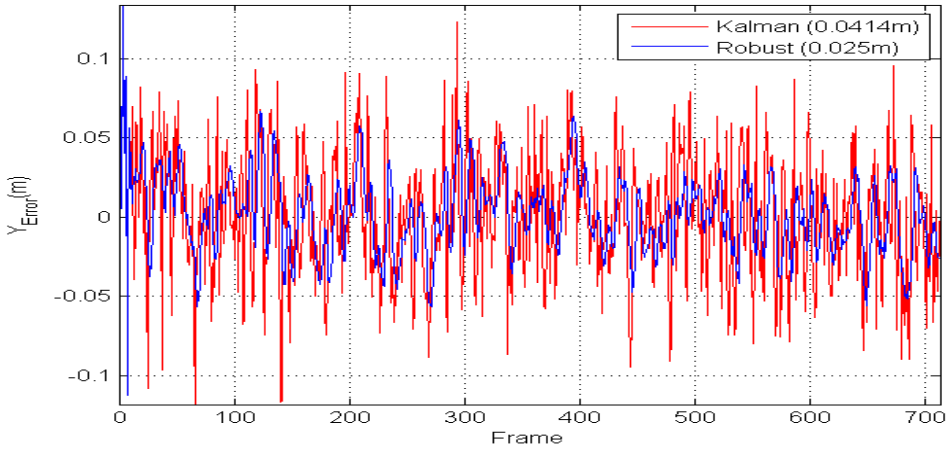


(b)

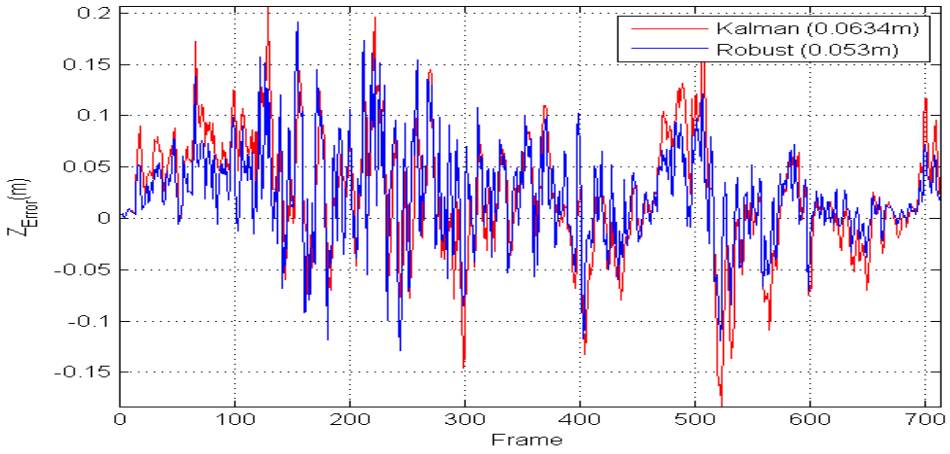
Figure 5.16 The best monoview tracking trajectory (a) 3D view of the trajectory. (b) xz view of the trajectory



(a)

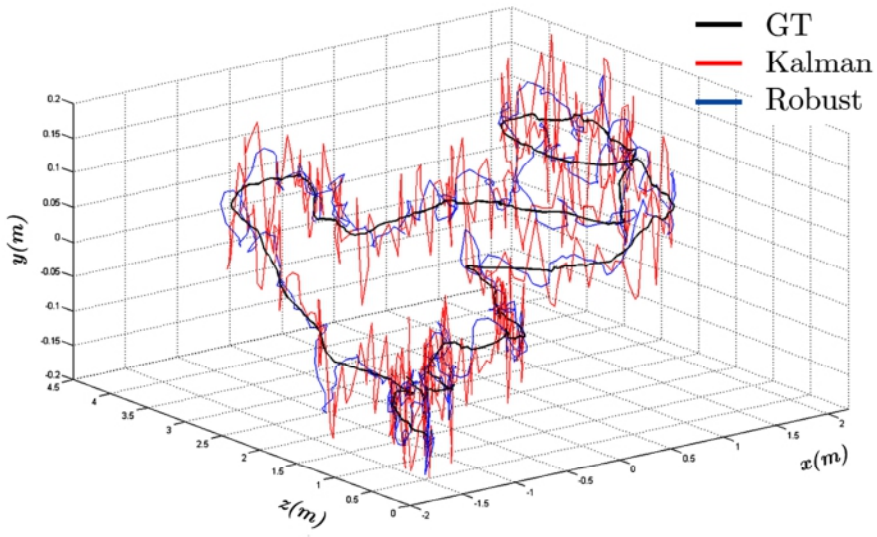


(b)

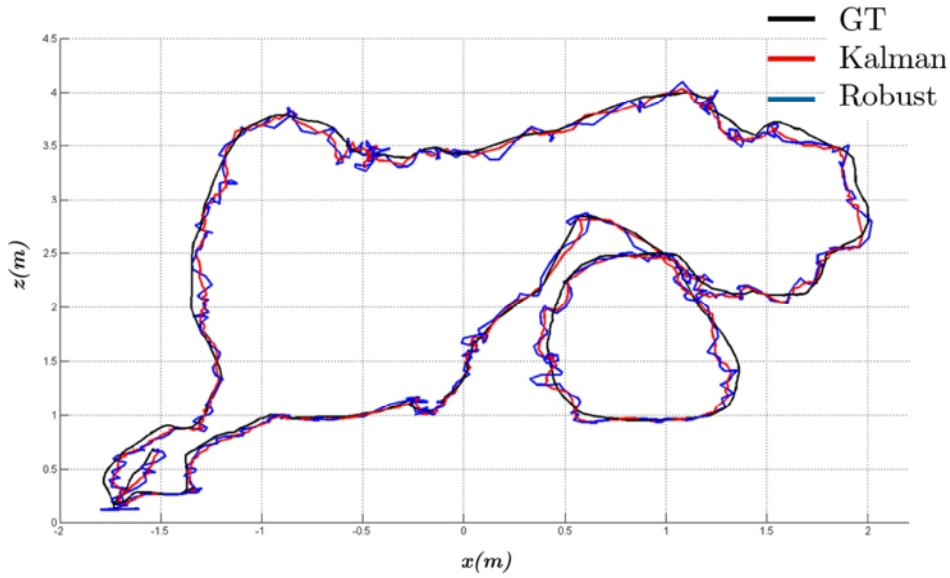


(c)

Figure 5.17 The worst monoview tracking RMSE (a) $\boldsymbol{x}$. (b) $\boldsymbol{y}$. (c) $\boldsymbol{z}$.

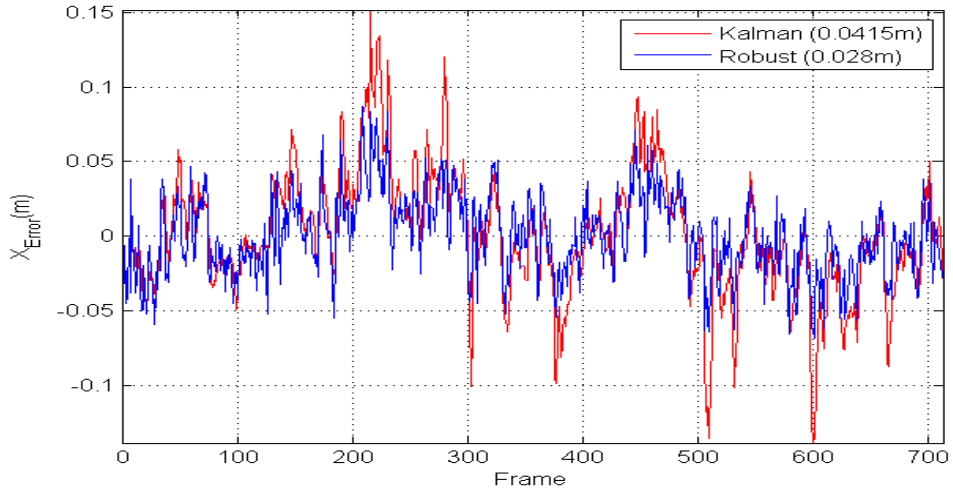


(a)

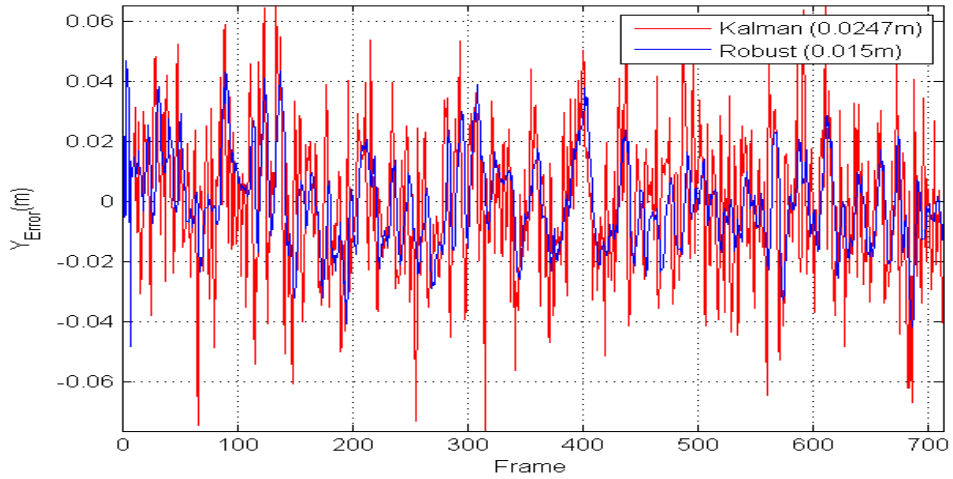


(b)

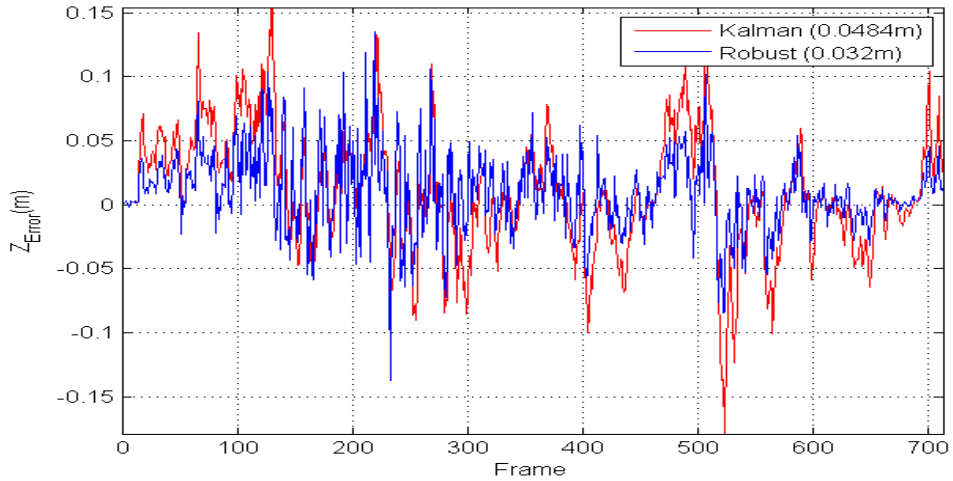
Figure 5.18 The worst monoview tracking results trajectory (a) 3D view of the trajectory. (b) xz view of the trajectory



(a)

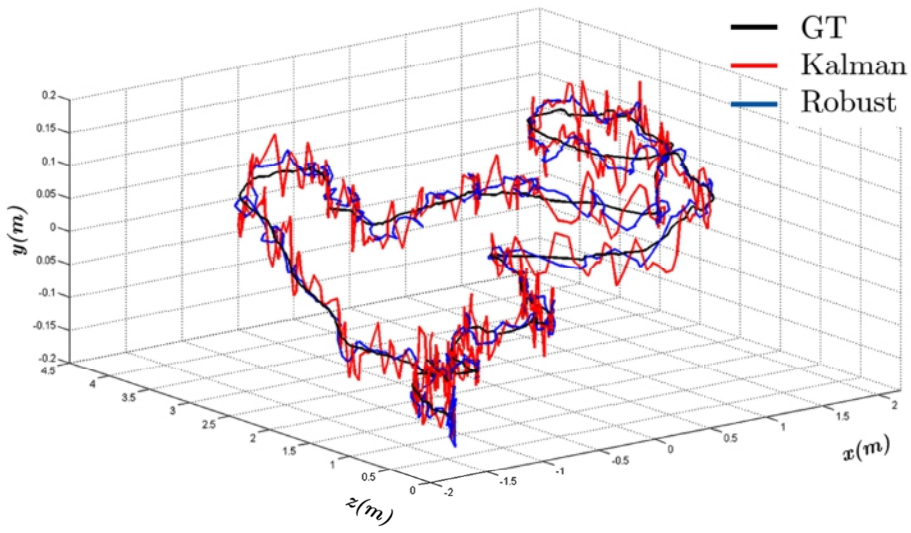


(b)

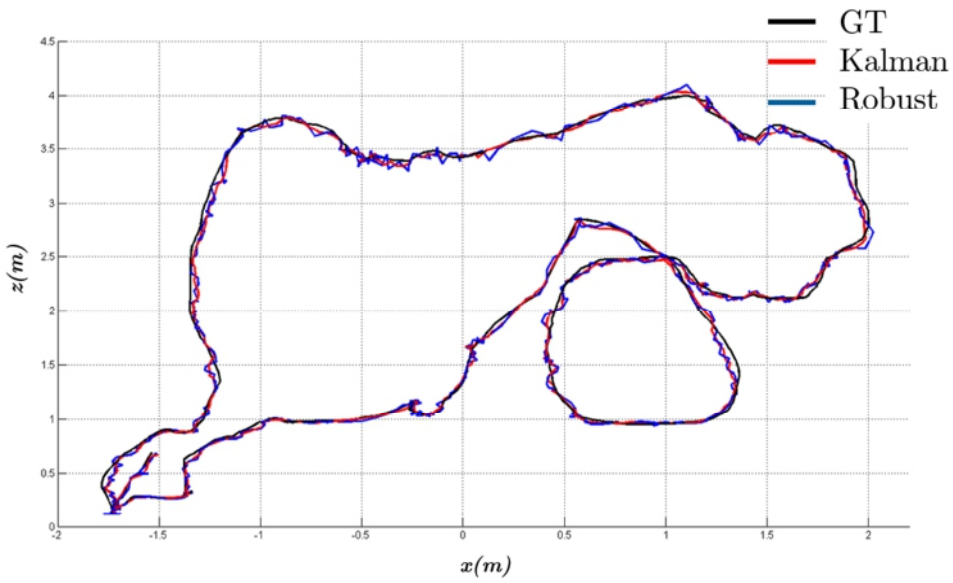


(c)

Figure 5.19 $\boldsymbol{P_n}$ multiview weighting RMSE (a) $\boldsymbol{x}$. (b) $\boldsymbol{y}$. (c) $\boldsymbol{z}$.

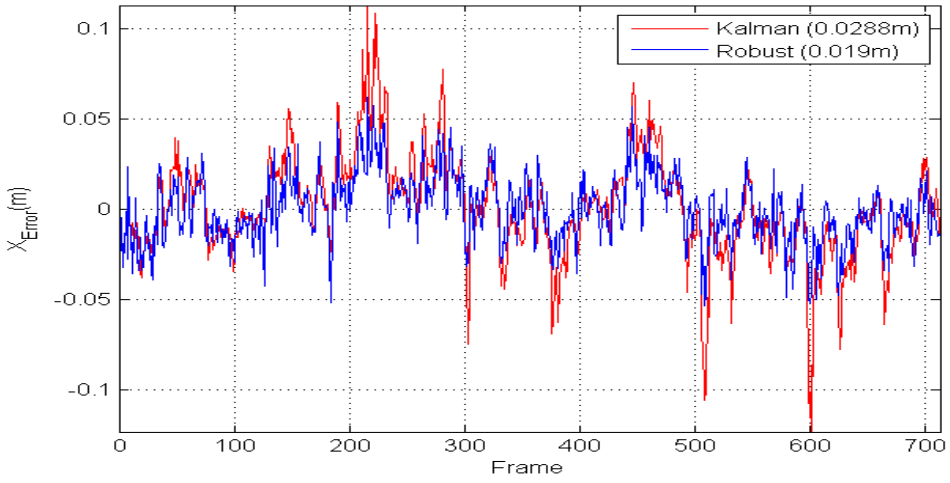


(a)

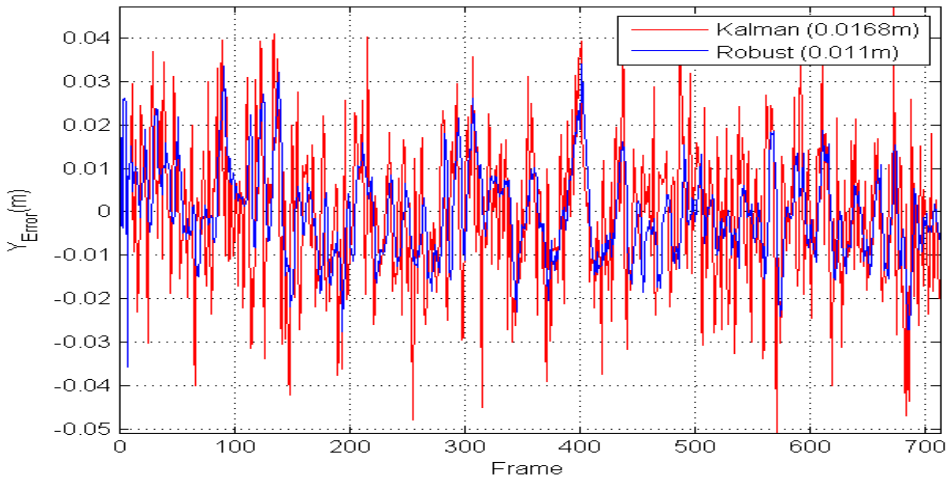


(b)

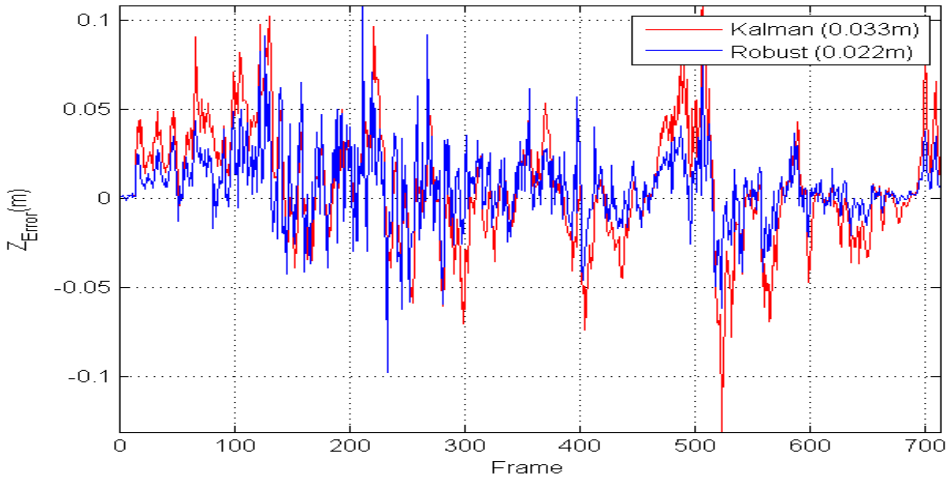
Figure 5.20 P_n multiview weighting trajectory. (a) 3D view of the trajectory. (b) xz view of the trajectory



(a)

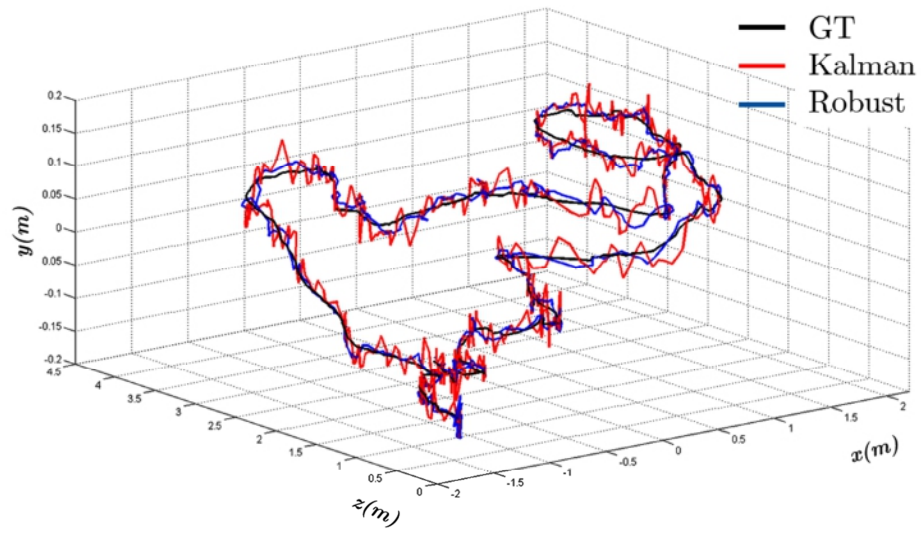


(b)

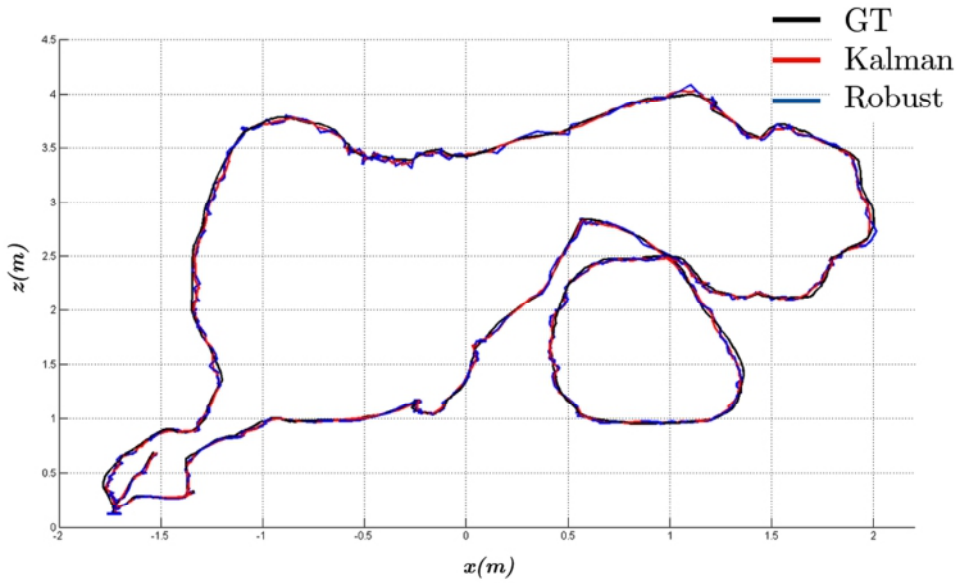


(c)

Figure 5.21 $\boldsymbol{P_n}$ and $\boldsymbol{D_n}$ multiview weighting RMSE (a) $\boldsymbol{x}$. (b) $\boldsymbol{y}$. (c) $\boldsymbol{z}$.

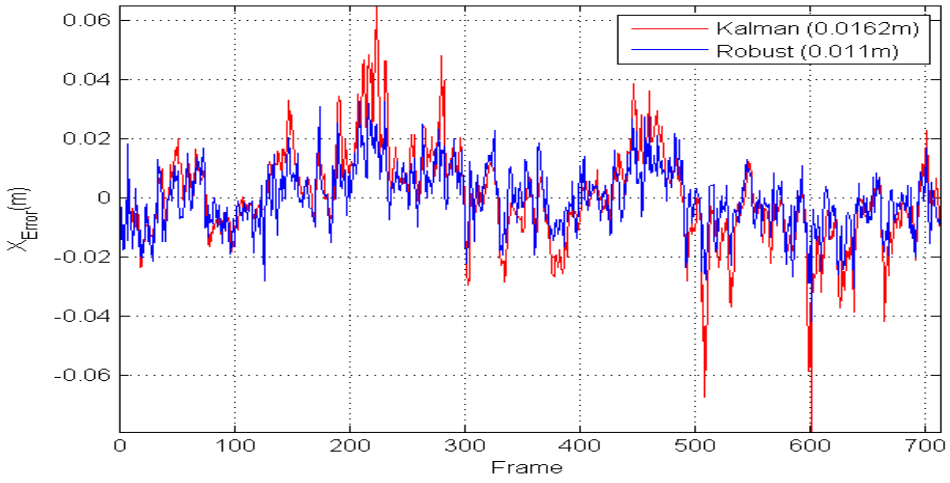


(a)

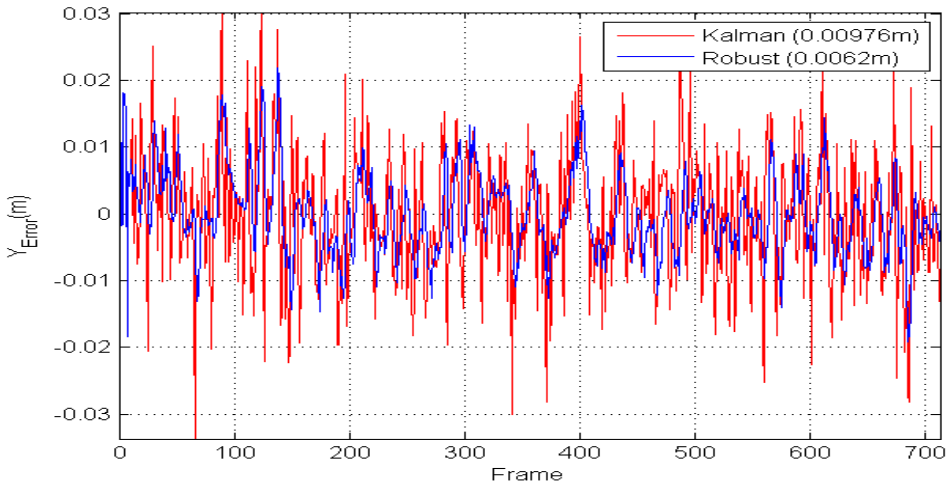


(b)

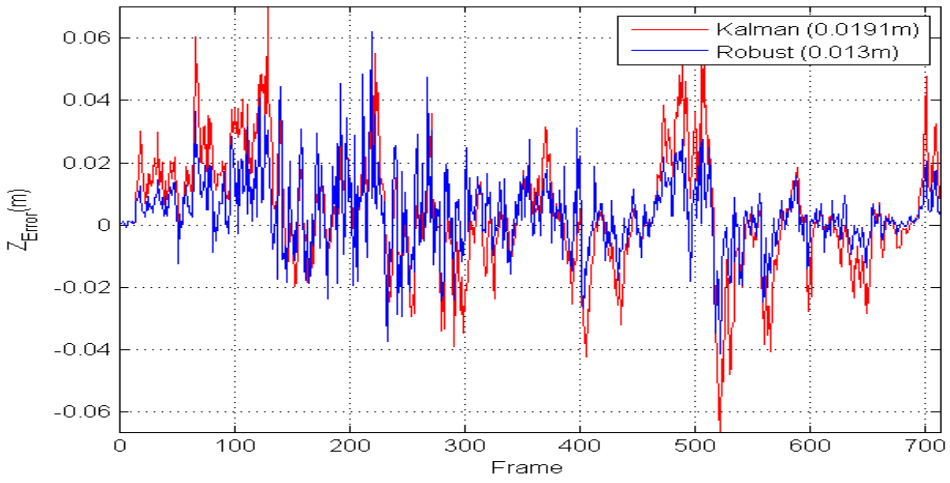
Figure 5.22 P_n and D_n multiview weighting trajectory. (a) 3D view of the trajectory. (b) xz view of the trajectory



(a)

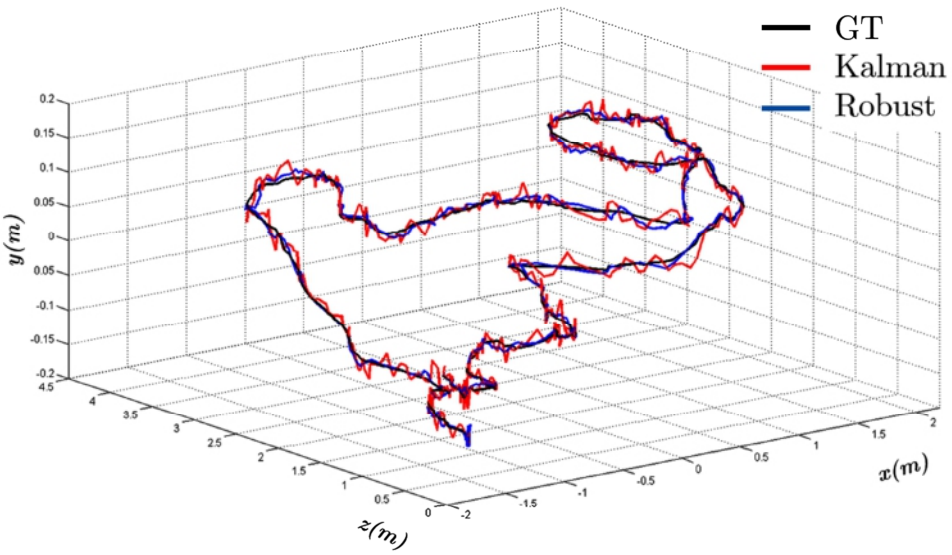


(b)

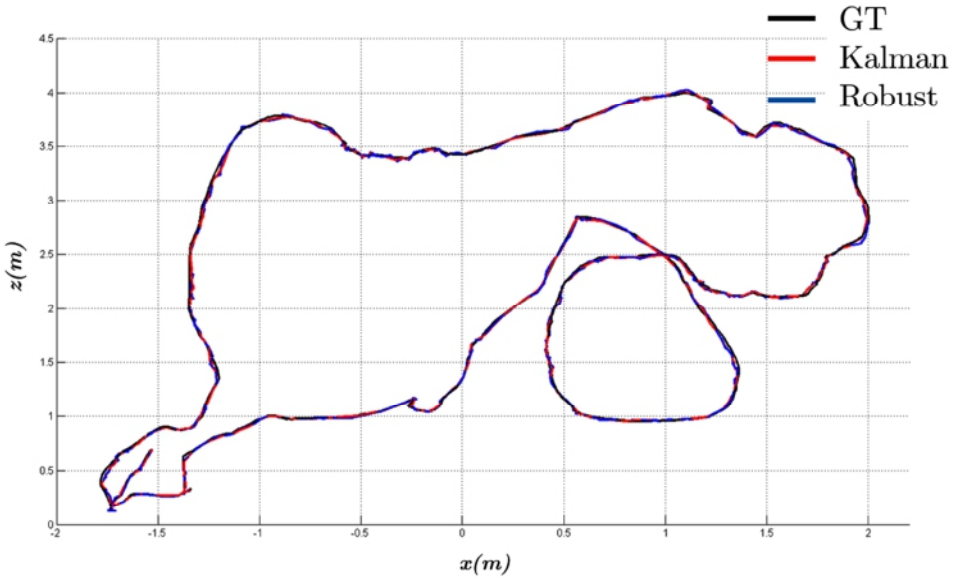


(c)

Figure 5.23 $\boldsymbol{P_n}$, $\boldsymbol{D_n}$ and $\boldsymbol{Z_n}$ multiview weighting RMSE (a) $\boldsymbol{x}$. (b) $\boldsymbol{y}$. (c) $\boldsymbol{z}$.



(a)



(b)

Figure 5.24 P_n, D_n and Z_n multiview weighting trajectory. (a) 3D view of the trajectory. (b) xz view of the trajectory

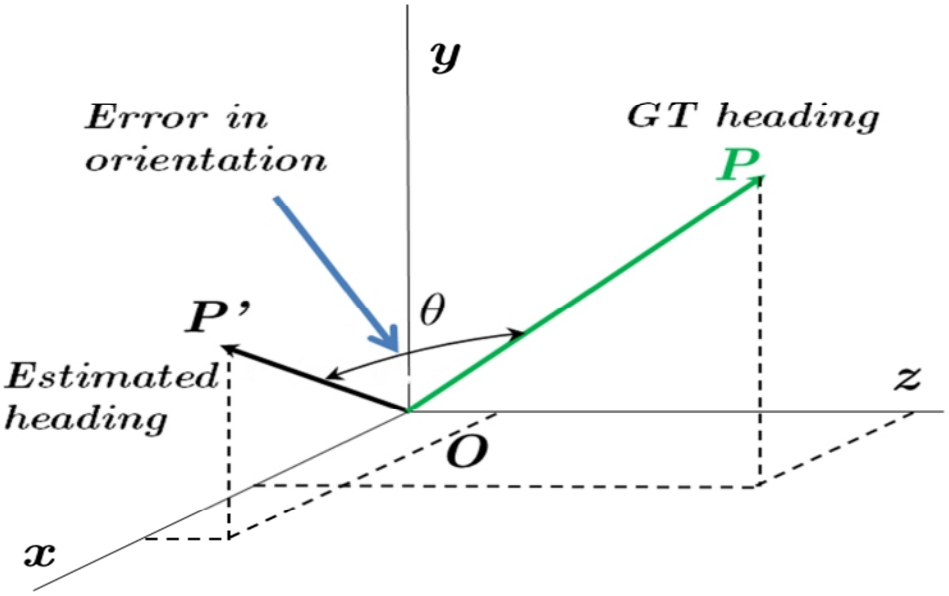


Figure 5.25 Angle between the GT and the estimated heading

5.8.5 Vehicle Orientation

To test the orientation obtained from the 6 DOF tracking solution, the error angles between the ground truth heading and the estimated one are computed. To this end, the dot product between the two vectors representing the ground truth and the expected direction of the robot is used. As one can see in Figure 5.25, the dot product between the two vectors $\vec{\mathbf{P}}, \vec{\mathbf{P}}'$ can be obtained from their magnitudes and the angle separating them as shown in Equation (5.54):

$$P \cdot P' = |P||P'|cos(\theta) \tag{5.54}$$

It results from Equation (5.54):

$$cos(\theta) = \frac{PP'}{|P||P'|} \tag{5.55}$$

Error angle between the two directions, therefore, becomes:

$$\theta = arcos\left(\frac{PP'}{|P||P'|}\right) \tag{5.56}$$

After applying Equation (5.56), the results illustrated in Figure 5.26 were obtained. The error in the orientation of the vehicle resulting from the RF was 11°. Whereas, KF's was 17°. With both filters, the error in the direction of the mobile robot was not significant. More importantly, the application of the CI improved the accuracy of the orientation angle as follows: with $\boldsymbol{P_n}$ on its own RF-RMSE was 7.1° and KF-RMSE was 12°. With $\boldsymbol{P_n}, \boldsymbol{D_n}$ the results is even better where RF-RMSE turned into 4.8° and KF-RMSE to 8.1°.

Finally, the introduction of $\boldsymbol{Z_n}$ significantly reduced the error to 2.7° for RF and 4.6° for KF. The result is very accurate given the fact that the three angles of orientation were not involved in the filtering algorithm. Consequently, the current approach to compute the heading of the vehicle has proven its effectiveness and high adequacy for real-time systems.

Filters	x-RMSE(m)	y-RMSE(m)	z-RMSE(m)
CI + KF	0.0415	0.0247	0.0484
CI + RF	0.0275	0.0154	0.0323
Difference CI+(KF-RF)	0.014	0.0093	0.0161

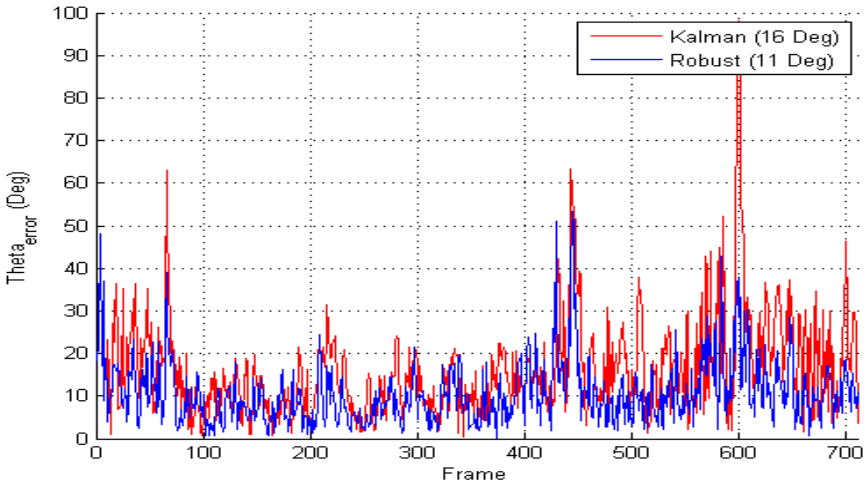
Table 5.4 Final tracking error after CI filtering with $\boldsymbol{P_n}$ weighting

Filters	x-RMSE(m)	y-RMSE(m)	z-RMSE(m)
CI + KF	0.0281	0.017	0.0333
CI + RF	0.0188	0.011	0.0223
Difference CI+(KF-RF)	0.0093	0.0065	0.011

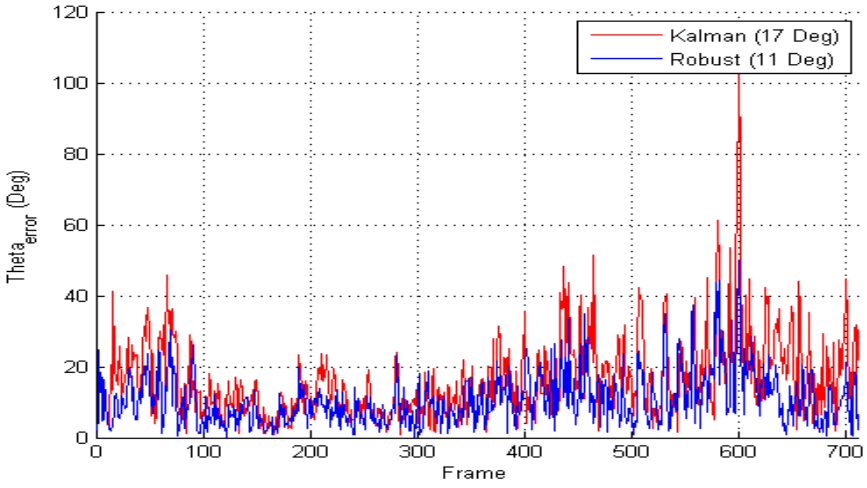
Table 5.5 Final tracking error after CI filtering with $\boldsymbol{P_n}$, $\boldsymbol{D_n}$ weighting

Filters	x-RMSE(m)	y-RMSE(m)	z-RMSE(m)
CI + KF	0.0165	0.0097	0.0195
CI + RF	0.011	0.006	0.0129
Difference CI+(KF-RF)	0.0055	0.0037	0.0066

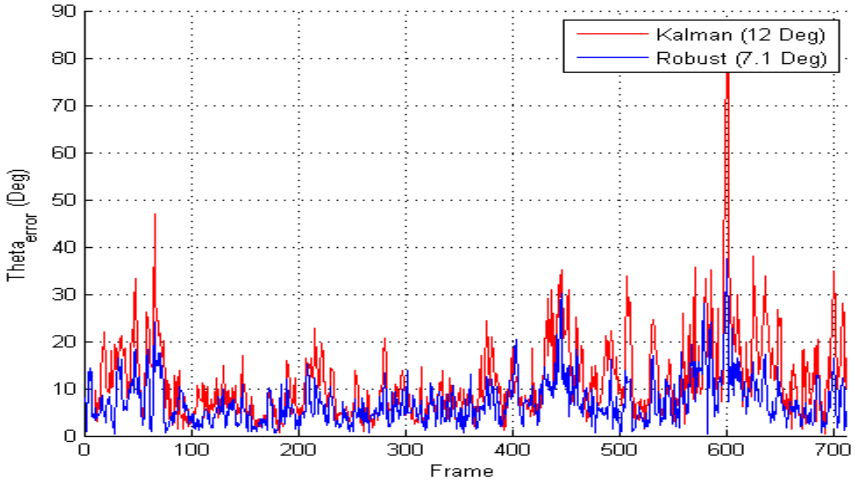
Table 5.6 Final tracking error after CI filtering with $\boldsymbol{P_n}$, $\boldsymbol{D_n}$ and $\boldsymbol{Z_n}$ weighting


Kalman (16 Deg)
Robust (11 Deg)
Theta_error (Deg)
Frame

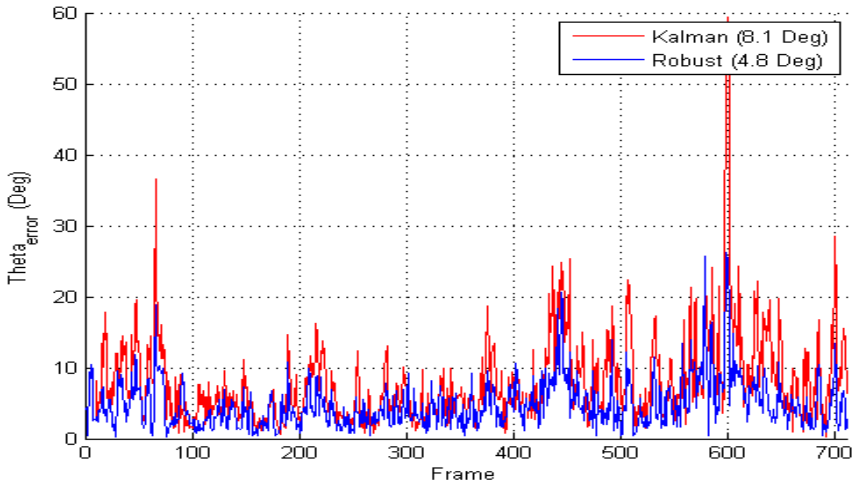
(a)


Kalman (17 Deg)
Robust (11 Deg)
Theta_error (Deg)
Frame

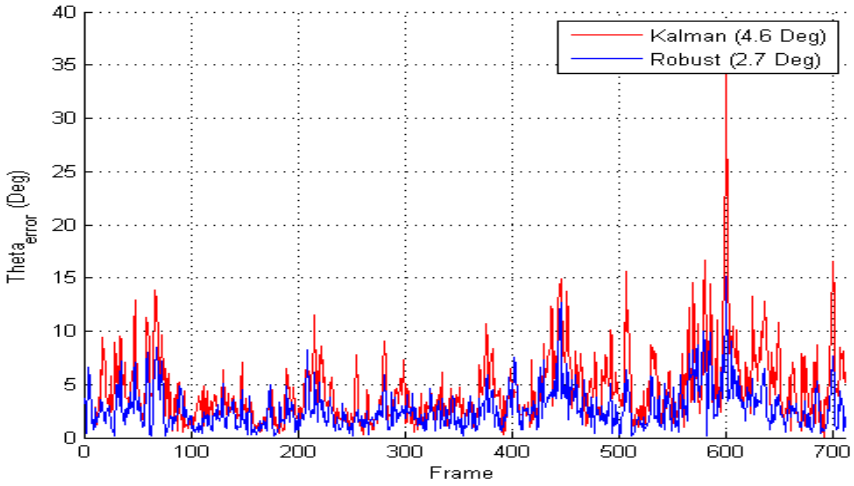
(b)


Kalman (12 Deg)
Robust (7.1 Deg)
Theta_error (Deg)
Frame

(c)



(d)



(e)

Figure 5.26 Error in the estimated orientation of the vehicle. **(a)** Best orientation estimation with RF. **(b)** Worst orientation estimation with RF. **(c)** CI with $\boldsymbol{P_n}$ weighting results. **(d)** CI with $\boldsymbol{P_n}$ and $\boldsymbol{D_n}$ weighting results. **(e)** CI with $\boldsymbol{P_n}$, $\boldsymbol{D_n}$ and $\boldsymbol{Z_n}$ weighting

5.9 Conclusion

A novel approach to the tracking the moving vehicles in indoor environments with a setup of multiple RGBD cameras was proposed. All the details about the methodology and the different constraints of implementation were described. Robust filtering was investigated in object tracking applications using multiple RGBD sensors. The power of the latter was demonstrated in overcoming the lack of knowledge about the system governing the motion of the vehicles. The quality of the measurements and the single estimates delivered by each sensor were then forwarded to a Covariance Intersection framework. The latter successfully combined all the individual contributions of the cameras in a single, consistent result.

The conducted experiments showed the achieved performance at a frame rate of 25 FPS with the five Kinects. The GPU implementation of the computationally demanding stages of processing (capture and markers extraction) helped significantly to accomplishing the real-time performance. However, no parallelisation was required for the robust filtering and the covariance intersection algorithms. The latter were just applied to the five sensor-wise position estimates (five 3D centroids). Their linear complexity of computation does not involve any parallel processing. Hence, they were implemented on the CPU, which indeed is more powerful in linear processing.

On the other hand, there are some limitations due to the active nature of the RGBD cameras that create interference problems, see Figure 5.27. A few solutions have been proposed in the literature to overcome such an inherent defect in this family of sensors [129]. However, these methods have other side effects such as the blurring of images that in turn causes substantial inaccuracies to the tracking. The current study does not deal with such an issue.

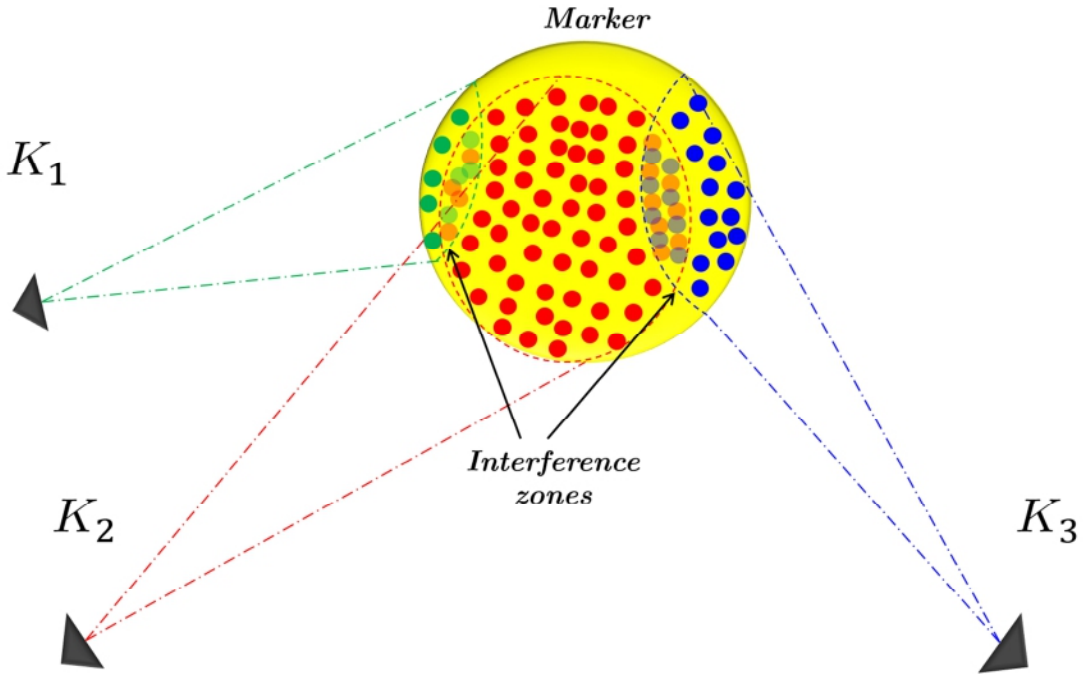


Figure 5.27 Interference between structured light patterns of the different RGBD cameras

6 A Recursive Robust Filtering Approach for 3D Registration

In this chapter, a recursive robust filtering approach for sparse 3D feature-based registration is proposed. Unlike the ordinary state of the art alignment algorithms, the proposed method has four advantages that have not cohabited altogether within any previous solution in the literature:

- It has the ability to deal with the inherent noise contaminating sensory data.
- It is robust to the uncertainties caused by uncertain feature localisation.
- It also combines the advantages of both L_∞ and L_2 norms for a higher performance and more prospective avoidance of local minima.
- It also provides an estimated rigid body transformation along with its error covariance. The latter enables a thorough control of the convergence regarding the alignment as well as a correct assessment of the quality of registration.

The mathematical rationale of the proposed approach is presented in detail. In addition, the results obtained from this 3D registration scheme are validated on various challenging scenarios with both synthetic and real data.

6.1 Overview

The widespread abundance of commercial 3D sensing devices for the general public and researchers at an affordable price has encouraged many enthusiasts to improve the quality of pose estimation algorithms. Many solutions and new algorithms have contributed to answering the growing need of the market. These algorithms leverage the raw sensor outputs and the high computational capability of multicore and graphic processors for a better human-machine interaction. Despite the high performance achieved with HD (High Definition) resolution images captured at 30 FPS in general, depth map in particular, the sensors still suffer from a significant measurement noise and a narrow field of view.

3D data registration is a very common tool that enables the recovery of the 6 DOF of the viewpoints from which the different scans were taken. For instance, the prime motivation of image registration returns back to the limited field of view regarding the real cameras. In other words, each viewpoint has its own coordinate system. Hence, emerges the need for the knowledge of transformations that map a given 3D dataset from one frame to another. To this end, several approaches have been proposed for point clouds, mesh and surface data registration. In most cases, objects of interest are assumed to be rigid. For this reason, their respective geometry remains unchanged over time. In addition, images taken from different viewpoints must share sufficiently large overlap regions. Based on the features observed in these areas, alignment algorithms compute a rigid body mapping that readjusts the images on each other. In practice, the alignment starts with the selection of some key points to be matched against each other in both source and target datasets [130]. Some of the matched pairs are systematically rejected when classified as outliers. Finally,

the minimisation of the distance between the pairs source/target key points yields the actual 3D transformation.

The computation of the 6 DOF transformations is achieved with a Least Squares (LS) minimisation algorithm. In practice, the output of the sensor is naturally contaminated with noise from potentially many different sources, each of which possibly has a different statistical nature (although the noise is assumed to be Gaussian) and amplitude. Assuming that a good initial guess may be available (e.g. human assisted alignment), most LS methods use Singular Value Decomposition (SVD) decomposition [131] or Horn's absolute rotation [34] [35] to estimate an approximate rigid body transformation. Moreover, these methods assume the whole source and target datasets being available before the processing takes place. However, in practice the data may be large, noisy, and streamed at a high frame rate, particularly when modern HD depth cameras are used. The latter are capable of delivering over two megapixel images at 30 FPS.

To cope with these shortcomings affecting the alignment process, a novel 3D registration solution is proposed in this thesis' chapter. The latter is capable of delivering 6 DOF transformations recursively, as well as of handling the noise and uncertainty seen in the position of 3D points. The link between ordinary LS registration methods such as: SVD [131] , Principal Component Analysis (PCA) [132], Iterative Closest Point (ICP) algorithm [133], and the recursive LS that is well known among optimal state estimation methods has been established. The relationship between the classic solutions and the recursive ones allows an efficient handling of uncertainties in parameters for a more robust registration.

The remainder of this chapter is organised as follows: In Section 6.2, the related works about 3D registration are discussed, and different alignment solutions that have been proposed in the literature are analysed. In Section 6.3, the target problem is formulated, and the elements of solution are clearly established. Two preliminary simplifications regarding translation and scale difference between two point clouds are examined to relax the problem. In Section 6.4, the

modelling behind the Weighted Least Squares is considered. Then, in Section 6.5, its recursive equivalent is deducted. In Section 6.6, the link between the Kalman filter and the RLS is settled. In Section 6.7, the registration model is fitted into a Kalman filter framework. In Section 6.8, the parametric uncertainty of the imaged 3D points is explained. Afterwards, in Section 6.9, the knowledge about uncertainty is used in the parameters of the Robust H_∞ filtering scheme. In section 6.10, the experimental results of the proposed registration approach applied to synthetic and real datasets including different noise levels are shown. In Section 6.11, the chapter is concluded and potential future works that can benefit from this modelling approach in alternative domains are debated.

6.2 Related Works

3D registration algorithms are useful for many applications such as 3D scanning, mapping, localisation, egomotion estimation and human body tracking. Mainly, this chapter delivers a more general solution to the estimation of the 6 DOF relating two states taken by the robot as seen in Chapter 5.

Schönemann was the first to publish a solution for the registration problem in 1966 [134]. Arun et al. [131] derived a closed-form solution to compute the absolute rotation with SVD. In the same year, Horn [34] proposed a similar solution based on Unit Quaternion, as well as another approach that uses the orthonormal matrices [35]. Rigid body alignment algorithms require an initialisation that can be achieved via several methods. The identification and indexation of features, scanner position tracking [135], principal axes of scans [136], exhaustive search for point correspondences [137]; are a few of the automatic solutions. On the other hand, a direct user action can also be useful to guess a good initial pose. Walker et al. [138], proposed an alternative solution to finding the absolute rotation using dual quaternions. Since its invention by Besl et al. [133], the Iterative Closest Point algorithm (ICP), has been considered as the state of the art point cloud alignment tool. Nevertheless, it

requires a good initial guess or some feature correspondences to avoid falling into a local minimum.

Nowadays, the newest variants such as EMICP [139] and SOFTASSIGN [140] are able to overcome some of the traditional ICP's limitations. Unlike the original algorithm, where each point in the source dataset has a single correspondent in the target one, the subsequent variants allow each point in the source to be checked against all the points belonging to the target dataset. To this end, the authors in [139] [140] introduced a weighting coefficient associated with every element. However, the computational effort to determine all possible combinations becomes a preventing factor when the size of the datasets grows beyond a thousand elements.

Other variants inspired from ICP were further proposed such as non-linear ICP [141], generalised ICP [142], and non-rigid ICP [143]. The latter have different levels of accuracy and convergence rates. What a given ICP variant can achieve may not be possible to accomplish with another variant. Hence, the decision about the usage of a particular registration algorithm depends on the nature of data and target application. Larusso et al. [144] showed that all the closed-form solutions are computationally similar. However, the performance can significantly differ. Thus, no single algorithm is exclusively optimal for all scenarios.

Umeyama [145] states in his work that Horn and Arun's algorithms fail when the datasets become highly corrupted with noise. He further proposed an alternative solution that uses Lagrange Multipliers [146]. Kanatani [147] simplified Umeyama's solution by fitting a rotation matrix to the 3D datasets using SVD. Granger et al. [148] reformulated the rigid body registration as a Maximum Likelihood problem with the Expectation Maximisation (EM) [149] approach to estimating transformation parameters. The authors update point correspondences between two datasets during the expectation step. Afterwards, they compute the parameters of the transformation using the derived correspondences.

A recursive solution, to sequentially estimate rigid body transformations with the Extended Kalman Filter (EKF) [24] was first proposed by Pennec et al. [150]. Ma et al. [151] followed the same strategy in order to align a dataset contaminated with isotropic Gaussian noise using the Unscented Particle Filter (UPF) [152]. This algorithm can accurately estimate the parameters for tiny datasets (less than one hundred elements). Ohta et al. [153] proposed the Generalised Total Least Squares (GTLS) method to compute a rotation matrix in the presence of anisotropic and inhomogeneous noise. An approximate algorithm for the 3D registration was later proposed by Balachandran et al. [154] in order to reduce the anisotropic noise.

Julier et al. also used an Unscented Kalman Filter (UKF) algorithm [155] to align two datasets following a sequential estimation. All these filter-based algorithms use the L_2 norm and consider the parameters being accurately determined beforehand. It is still impossible to completely eliminate the uncertainty from the parameters of the filter. The proposed method minimises the cost function in the L_2 norm sense as long as the provided parameters are assumed to be accurate enough. On the other hand, when the latter (parameters) have not been carefully determined, or in the case where alternative hardware precision limitations intervene, a non-negligible uncertainty amount must be properly included in the modelling for a more robust estimation. Such a carefulness yields a good maintenance of estimation error within a predefined bond by optimising in the L_∞ instead of L_2 norm.

This norm has been adopted as a standard in the community of mathematical optimisation and computer vision to solve a particular category of optimisation problems [156]. Micusik et al. [157] localise non-overlapping cameras using Second Order Cone Programming (SOCP) to minimise the L_∞ norm. They showed a good performance of SOCP for camera centre localisation with a fairly small error magnitude. Lee et al. [158] further claimed that by using L_∞ a number of computer vision problems such as homography estimation

(considered a quasi-convex problem) can be formulated and solved using the Bisection method.

Despite the latest developments in L_∞ based solutions, vision optimisation can provide accurate and globally optimal solutions, but the practical implementation is computationally demanding. The method proposed and implemented in this thesis on the other hand, uses a simpler and more computationally attractive L_∞ based filtering approach to constrain the worst-case error.

6.3 Problem Statement

Let us consider two sets of 3D points belonging to the source and the target point clouds $\boldsymbol{Q} = \{\boldsymbol{q_1}, \dots, \boldsymbol{q_n}\}$, $\boldsymbol{P} = \{\boldsymbol{p_1}, \dots, \boldsymbol{p_n}\}$, respectively. Each of the elements $\boldsymbol{p_i}, \boldsymbol{q_i}$ within the sets of points has three components $\boldsymbol{p_i} = (\boldsymbol{x_p}, \boldsymbol{y_p}, \boldsymbol{z_p})_i$ and $\boldsymbol{q_i} = (\boldsymbol{x_q}, \boldsymbol{y_q}, \boldsymbol{z_q})_i$. The $\boldsymbol{k}$-th point $\boldsymbol{q_k}$ in the source point cloud is matched a priori with the $\boldsymbol{k}$-th point in the target point cloud $\boldsymbol{p_k}$. The purpose of a 3D registration operation is finding a rigid body transformation ($\boldsymbol{t}$: translation, $\boldsymbol{R}$: rotation) that best maps the source point cloud $\boldsymbol{Q}$ onto the target one $\boldsymbol{P}$. The determination of such a mapping can be modelled as an optimisation problem. Nevertheless, due to the noisy outputs streamed by the sensor, an exact solution is very unlikely to be determined. Thus, a realistic model must take into account alignment error $\boldsymbol{e_i}$ as follows:

$$p_i = Rq_i + t + e_i \tag{6.1}$$

Equation (6.1) can be re-written in the form:

$$e_i = p_i - (Rq_i + t) \tag{6.2}$$

The rigid body transformation is optimal when the sum of the squares of errors ($\boldsymbol{e_i}$) becomes minimal:

$$e^2 = \underset{R,t}{argmin} \sum_{i=1}^{n} \|p_i - (Rq_i + t)\|^2 \tag{6.3}$$

Where:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{6.4}$$

$$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{6.5}$$

6.3.1 Preliminary Translation Estimation

It is possible to simplify the problem of Equation (6.3) by decoupling the translation vector $\boldsymbol{t}$ from the cost function following the steps below:

Let $\overline{\boldsymbol{p}}$ and $\bar{\boldsymbol{q}}$ be the centroids of the two point clouds,

$$\begin{aligned} \bar{p} &= \frac{1}{n}\sum_{i=1}^{n} p_i \\ \bar{q} &= \frac{1}{n}\sum_{i=1}^{n} q_i \end{aligned} \tag{6.6}$$

The two datasets are centred by translating the 3D points to the origin of the world frame,

$$\overline{P} = P - \bar{p}$$

$$\overline{Q} = Q - \bar{q} \tag{6.7}$$

Hence, Equation (6.2) becomes:

$$e_i = \bar{p}_i - R\bar{q}_i - (t + R\bar{q} - \bar{p}) \tag{6.8}$$

The decoupled translation $\bar{\boldsymbol{t}}$ is:

$$\bar{t} = t + R\bar{q} - \bar{p} \tag{6.9}$$

The error function to be minimised, therefore, becomes:

$$\begin{aligned} e^2 &= \sum_{i=1}^{n} \|(\bar{p}_i - R\bar{q}_i) - \bar{t}\|^2 \\ &= \sum_{i=1}^{n} \|\bar{p}_i - R\bar{q}_i\|^2 - 2\bar{t} \sum_{i=1}^{n} \|\bar{p}_i - R\bar{q}_i\| + n\|\bar{t}\|^2 \end{aligned} \tag{6.10}$$

It results from Equations (6.6) and (6.7):

$$\begin{aligned} \sum_{i=1}^{n} \bar{p}_i = \sum_{i=1}^{n} (p_i - \bar{p}) \Rightarrow \sum_{i=1}^{n} p_i - n\left(\frac{1}{n}\sum_{i=1}^{n} p_i\right) = 0 \\ \sum_{i=1}^{n} \bar{q}_i = \sum_{i=1}^{n} (q_i - \bar{q}) \Rightarrow \sum_{i=1}^{n} q_i - n\left(\frac{1}{n}\sum_{i=1}^{n} q_i\right) = 0 \end{aligned} \tag{6.11}$$

Consequently,

$$e^2 = \sum_{i=1}^{n} \|\bar{p}_i - R\bar{q}_i\|^2 + n\|\bar{t}\|^2 \tag{6.12}$$

Equation (6.12) represents a positive function of $\bar{\boldsymbol{t}}$. The latter attains its minimum value at $\bar{\boldsymbol{t}} = [\mathbf{0}, \mathbf{0}, \mathbf{0}]$ because the first term is constant. Equation (6.9) subsequently becomes:

$$t = \bar{p} - R\bar{q} \tag{6.13}$$

This equation represents an optimal translation between the source and the target centroids. In other words, a good initial guess at the translation can be obtained as shown in Equation (6.14). Instead of searching for an optimal solution in a space of 12 dimensions, i.e. 3 entries for translation $\boldsymbol{t}$ and 9 entries for rotation $\boldsymbol{R}$ which amounts to 12 total. Preliminary translation estimate reduces the dimensionality of the problem to 9. The value of the initial translation becomes:

$$\tilde{t} = \bar{p} - \bar{q} \tag{6.14}$$

6.3.2 Scale Difference Elimination

$$e^2 = \sum_{i=1}^{n} \|\bar{p}_i - sR\bar{q}_i\|^2 \tag{6.15}$$

Using Equation (6.15) and the fact that $\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \|\bar{\boldsymbol{q}}_{\boldsymbol{i}}\|^{\boldsymbol{2}} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \|\boldsymbol{R}\bar{\boldsymbol{q}}_{\boldsymbol{i}}\|^{\boldsymbol{2}}$, one can estimate the scale factor between the two sets of points from the following equation:

$$e^2 = \sum_{i=1}^{n} \|\bar{p}_i\|^2 - 2s \sum_{i=1}^{n} \bar{p}_i R\bar{q}_i + s^2 \sum_{i=1}^{n} \|\bar{q}_i\|^2 \tag{6.16}$$

The inner products between the source and the target vectors of data are replaced as follows:

$$G = \sum_{i=1}^{n} \|\bar{p}_i\|^2 \tag{6.17}$$

$$W = \sum_{i=1}^{n} \|\bar{q}_i\|^2 \tag{6.18}$$

$$D = \sum_{i=1}^{n} \bar{p}_i R \bar{q}_i \tag{6.19}$$

Substituting the new variables $\boldsymbol{G}$, $\boldsymbol{W}$ and $\boldsymbol{D}$ in Equation (6.16) results in:

$$e^2 = G - 2sD + s^2 W \tag{6.20}$$

Equation (6.20) is reformulated as follows:

$$e^2 = \left(s\sqrt{W} - \frac{D}{\sqrt{W}} \right)^2 + \frac{GW - D^2}{W} \tag{6.21}$$

Expression (6.21) is minimal when $\boldsymbol{s}$ takes the following value:

$$s = \sum_{i=1}^{n} \left(\frac{\bar{p}_i R \bar{q}_i}{\|\bar{q}_i\|^2} \right) \tag{6.22}$$

A good initial guess for the scale factor can, consequently, be:

$$\tilde{s} = \sum_{i=1}^{n} \left(\frac{\bar{p}_i \bar{q}_i}{\|\bar{q}_i\|^2} \right) \tag{6.23}$$

6.3.3 Optimal Rotation Estimation

Up until now, it has been possible to decouple the translation ($\boldsymbol{t}$) and eliminate the scale difference ($\boldsymbol{s}$) between the source and the target point clouds after the computation of the maximal value taken by the sum of the squared errors. Nevertheless, the determination of the rotation transform is more complicated and computationally challenging. The previous results $(\tilde{\boldsymbol{s}}, \tilde{\boldsymbol{t}})$ obtained from Equations (6.14), (6.23) are orientation-invariant. Thus, the registration problem (rigid-transformation estimation) is reduced to the estimation of the rotation between point clouds from Equation (6.3) to the following simplified form:

$$e^2 = \underset{R}{argmin} \sum_{i=1}^{n} \|\bar{p}_i - R\bar{q}_i\|^2 \tag{6.24}$$

Once the optimal rotation $\widehat{\boldsymbol{R}}$ is computed, the optimal translation $\hat{\boldsymbol{t}}$ can be directly deduced, as well as the best scale $\hat{\boldsymbol{s}}$ between the two sets of points using Equations (6.13) and (6.22).

The best rotation $\widehat{\boldsymbol{R}}$ can be obtained with Least Squares minimisation tools that are capable of providing a closed form solution for the orientation relating source and target point clouds. This solution is sufficient for most applications. However, if the inputs are significantly contaminated with measurement noise, the transformation becomes unstable, i.e. very sensitive to perturbations in the data to be aligned.

6.4 Weighted Least Squares (WLS) Estimation

The Least Squares registration is sufficient to estimate the rigid-body transformation between two sets of points. In the cost function of Equation (6.24) there are more equations than unknowns. The function is convex because it is quadratic in R. A single optimal solution is therefore expected to be found by the minimisation algorithm. The latter can either be the SVD [131], quaternions [34], orthonormal matrices [35] or dual quaternions [138]. On the other hand, the feasibility of the solution requires at least three pairs of correspondences (9 pairs of coordinates) that are linearly independent in order to solve each equation in the system.

Another class of solutions that is based on the non-linear methods, such as Levenberg-Marquardt, or the linearization of the cost function with the assumption of the incremental elementary rotations for infinitesimal changes in $\boldsymbol{R}$. The latter can then be approximated by a skew symmetric matrix where the entries are the actual rotation angles over the three main axes. A recent survey on 3D registration [159] cited several approaches aiming at the best

rotation matrix. However, the authors have not mentioned any time-varying solution, i.e. a solution based on iterative/recursive optimal filtering framework.

The Weighted Least Squares (WLS) estimation is a variant of the original Least Squares (LS) algorithm. Real world datasets contain different measurements with various confidence levels. WLS attributes a weighting coefficient to every measurement, computed with its respective accuracy. In the registration case, the elements of the 3D dataset (points in point clouds and triangles for 3D surfaces) are ordered according to their uncertainty, i.e. the inverse of confidence or certainty. The uncertainty quantifies the amount of noise in measurements. The noisier the measurements, the less their contribution in terms of useful information.

Suppose that one wants to estimate the best value ($\widehat{\boldsymbol{x}}$) of an $\boldsymbol{n}$-element vector $\boldsymbol{x}$ from a series of $\boldsymbol{k}$ noisy measurements $\boldsymbol{y_i}$. $\boldsymbol{v_i}$ are independent random variables representing measurement noise. The mathematical formulation of the WLS algorithm is based on the least squares estimation as shown in these equations:

$$\begin{aligned} y_1 &= H_{11}x_1 + \cdots + H_{1n}x_n + v_1 \\ &\vdots \\ y_k &= H_{k1}x_1 + \cdots + H_{kn}x_n + v_k \end{aligned} \tag{6.25}$$

The matrix form of Equation (6.25) with $\boldsymbol{H}$ a $\boldsymbol{k} \times \boldsymbol{n}$ projection matrix that maps an $\boldsymbol{n}$-dimensional vector into the $\boldsymbol{k}$-dimensional space, where the vector $\boldsymbol{y} = [\boldsymbol{y_1}, \ldots, \boldsymbol{y_k}]^T$ rests:

$$y = Hx + v \tag{6.26}$$

$$e = y - H\widehat{x} \tag{6.27}$$

The solution that minimises the error $\boldsymbol{e}$ [160] is given by:

$$\widehat{x} = (H^T H)^{-1} H^T y \tag{6.28}$$

Since it is assumed that the noise process for each measurement is non-zero and independent, the covariance matrix becomes, with $\boldsymbol{v} = [\boldsymbol{v_1}, \dots, \boldsymbol{v_k}]^T$:

$$R = E(vv^T) \tag{6.29}$$

The variance of each observation noise $\boldsymbol{v_i}$ is $\boldsymbol{E(v_i^2)} = \boldsymbol{\sigma_i^2}; \boldsymbol{1 \leq i \leq k}$.

The solution that minimises the error $\boldsymbol{e}$ [160] of the weighted observations is given by:

$$\widehat{x} = (H^T R^{-1} H)^{-1} H^T R^{-1} y \tag{6.30}$$

The asset of the WLS over its original counterpart LS lies in its ability to handle situations where the elements of the dataset are not similar in quality. In such a case, observation errors are not uniform. As a result, the weighted version of least squares estimation takes advantage from just the useful information provided by the data and yields an accurate estimate [161]. These weighting factors (covariance matrix) are assumed to have been precisely determined beforehand. However, in real scenarios another type of the estimated weightings is used instead. The latter are computed from the properties of observation noise. If redundant observations exist in the dataset, the result of the WLS estimation will be adversely affected. Such a drawback is more likely to occur in small datasets [162] [163].

Outliers are another performance limiting factor of the WLS. Captured measurements must be cleaned thoroughly with an appropriate outlier removal algorithm such as RANSAC [164]. Otherwise, outliers can have a significant effect that causes the WLS to promote them instead of the useful data. In this case, the result may become worse than the naive least squares estimation.

6.5 Recursive Least Squares (RLS) Estimation

In many real-world scenarios, observations are not entirely available. Instead, they are streamed progressively. This can be particularly noticed when the size of the outputs is important, or when their respective processing is time-consuming. The previous formulation of the WLS does not allow such a time-varying update of the already computed estimate $\hat{\boldsymbol{x}}$ to be performed. When a new sample $\boldsymbol{y_k}$ arrives, all the parameters of the filter are entirely re-evaluated. In addition, the matrices $\boldsymbol{R^{-1}}$ and $\boldsymbol{H}$ are augmented with the recently-delivered observation. In order to avoid re-computing the estimate $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$ from scratch, an alternative recursive version of the ordinary WLS algorithm is suggested. The latter takes advantage of the already available estimate $\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}$ to compute the current one $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$ without re-calculating the entire expression in Equation (6.30). The recursive estimator [77] can be written in the form:

$$y_k = H_k x_k + v_k \tag{6.31}$$

$$\widehat{x}_k = \widehat{x}_{k-1} + K_k(y_k - H_k \widehat{x}_{k-1}) \tag{6.32}$$

Equations (6.31), (6.32) are a particular case of the Kalman filter [165]. $\boldsymbol{K_k}$ is called the *gain parameter* and the difference $(\boldsymbol{y_k} - \boldsymbol{H_k}\hat{\boldsymbol{x}}_{\boldsymbol{k-1}})$ is called *innovation*. The latter represents how significant is the contribution of the last observation to the final estimate. In cases where either the gain or the innovation becomes zero, the last sample would not have any contribution to the estimation. $\boldsymbol{K_k}$ is accordingly determined as follows:

$$E(e_k) = E[x_k - \widehat{x}_k]$$

$$= E[x_k - \widehat{x}_{k-1} - K_k(y_k - H_k \widehat{x}_{k-1})]$$

$$
\begin{aligned}
&= E[e_{k-1} - K_k(H_k x_k + v_k - H_k \widehat{x}_{k-1})] \\
&= E[e_{k-1} - K_k H_k(x_k - \widehat{x}_{k-1}) - K_k v_k] \\
&= E[(I - K_k H_k)e_{k-1} - v_k] \\
&= (I - K_k H_k)E(e_{k-1}) - K_k E(v_k)
\end{aligned} \tag{6.33}
$$

Another condition should be taken into account to determine the best estimate. The last mentioned concerns the minimisation of estimation-error covariance:

$$
\begin{aligned}
S_k &= E\left[\textstyle\sum_{i=1}^{n}(x_i - \widehat{x}_i)^2\right] \\
&= E\left[\sum_{i=1}^{n} e_{i,k}^2\right] \\
&= E[e_k^T e_k] \\
&= E[tr(e_k e_k^T)] \\
&= tr(P_k)
\end{aligned} \tag{6.34}
$$

$\boldsymbol{P_k}$ is the covariance of the estimation-error. A recursive form for it can be obtained with the same rationale followed in Equations (6.31) to (6.33):

$$P_k = E[e_k e_k^T]$$

$$= E\left[\left((I - K_k H_k)e_{k-1} - K_k v_k\right)\left((I - K_k H_k)e_{k-1} - K_k v_k\right)^T\right]$$

$$= E[(I - K_k H_k)e_{k-1}e_{k-1}^T(I - K_k H_k)^T - K_k v_k e_{k-1}^T(I - K_k H_k)^T \\ - (I - K_k H_k)e_{k-1}v_k^T K_k^T + K_k v_k v_k^T K_k^T]$$

$$= (I - K_k H_k)E(e_{k-1}e_{k-1}^T)(I - K_k H_k)^T - K_k E(v_k e_{k-1}^T)\ (I - K_k H_k)^T \\ -(I - K_k H_k)E(e_{k-1}v_k^T)K_k^T + K_k E(v_k v_k^T)K_k^T \tag{6.35}$$

With measurements noise $\boldsymbol{v_k}$ being independent from estimation-error $\boldsymbol{e_k}$ one gets:

$$E(v_k e_{k-1}^T) = E(v_k)E(e_{k-1}^T); \\ E(e_{k-1}v_k^T) = E(e_{k-1})E(v_k^T) \tag{6.36}$$

$$E(v_k) = E(v_k^T) = 0 \xrightarrow{yields} E(v_k e_{k-1}^T) = E(e_{k-1}v_k^T) = 0 \tag{6.37}$$

Equation (6.37) is verified because $\boldsymbol{v_k}$ is a white noise process. Equation (6.35) therefore becomes:

$$P_k = (I - K_k H_k)P_{k-1}(I - K_k H_k)^T + K_k R_k K_k^T \tag{6.38}$$

$\boldsymbol{R_k}$ is the covariance of the measurement noise $\boldsymbol{v_k}$. The variable $\boldsymbol{P_k}$ of Equation (6.38) represents the $time - varying$ form of the estimation error covariance resulting from the RLS.

From the same Equation (6.38), it is plausible that $\boldsymbol{P_k}$ is proportional to $\boldsymbol{R_k}$. The best result in the estimation of $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$ is consequently reached at the minimum of $\boldsymbol{E}[\boldsymbol{tr}(\boldsymbol{P_k})]$.

In order to find the best value of the gain $\boldsymbol{K_k}$, the result of Equation (6.38) is applied with the chain rule of Equations (6.39) and (6.40):

$$\frac{\partial tr(ABA^T)}{\partial A} = 2AB \ if \ B = B^T \tag{6.39}$$

$$\frac{\partial P_k}{\partial K_k} = 2(I - K_k H_k) P_{k-1}(-H_k^T) + 2K_k R_k \tag{6.40}$$

$\boldsymbol{K_k}$ is therefore minimised by setting the derivative of Equation (6.40) to zero:

$$\begin{aligned} K_k R_k &= (I - K_k H_k) P_{k-1} H_k^T \\ K_k R_k &= P_{k-1} H_k^T - K_k H_k P_{k-1} H_k^T \\ K_k (R_k + H_k P_{k-1} H_k^T) &= P_{k-1} H_k^T \\ K_k &= P_{k-1} H_k^T (H_k P_{k-1} H_k^T + R_k)^{-1} \end{aligned} \tag{6.41}$$

The global least squares estimation algorithm is summarised in Algorithm 6.1.

Algorithm 6.1 Recursive Least Squares Algorithm

Initialisation

$$\hat{x}_0 = E(x)$$
$$P_0 = E[(x - \hat{x}_0)(x - \hat{x}_0)^T]$$

for *every iteration* $(k = 1, n)$

Read a new measurement:

$$y_k = H_k x + v_k$$

Then, carry out through the estimation:

$$K_k = P_{k-1} H_k^T (H_k P_{k-1} H_k^T + R_k)^{-1}$$
$$\hat{x}_k = \hat{x}_{k-1} + K_k (y_k - H_k \hat{x}_{k-1})$$
$$P_k = (I - K_k H_k) P_{k-1} (I - K_k H_k)^T + K_k R_k K_k^T$$

end

6.6 Kalman Filter and RLS

The process of recursive registration starts with the capture of the 3D data Figure 6.1 (a). The captured data is sent through a 3D feature extraction algorithm. The selection of a relatively small subset of features instead of the

whole point cloud is advantageous in two aspects: the first is the significant decrease of processing time as the size of feature's subset is around 1/80 smaller than the one of the entire 3D map; on the other hand, feature points are more representative, stable and almost outlier-free. For instance, outliers can considerably mislead the search for the optimal 3D transformation. Correspondence computation algorithms are responsible for the matching of key points (features). The result of this stage is a list of source/target pairs of features.

The Data Recasting Module embeds the coordinates of a source feature within a given pair in a matrix called the Projection matrix $\boldsymbol{H_k}$, Figure 6.1 (b). Target feature point coordinates are also inserted in a vector $\boldsymbol{z_k}$. $\boldsymbol{H_k}$ and $\boldsymbol{z_k}$ serve as parameters for the ultimate Recursive Registration solution.

Figure 6.2 (a) illustrates a general architecture characterising the state-transition filters. The equations are taken from Kalman filter definition in Chapter 3. The choice of this framework is motivated by the fact that the Kalman filter is the basis for all the subsequent alternatives in the same family (recursive optimal state estimators). All the filters belonging to this family are divided into two stages: *Prediction* and *Correction*.

Recursive filtering is a branch of the prediction/correction paradigm. The last mentioned enables the computation of the a priori estimation based on the dynamic nature of the target system, followed by a correction that associates with the actual measurements of the a priori estimate. The algorithm progressively refines the estimated transformation at the reception of new pairs of feature points.

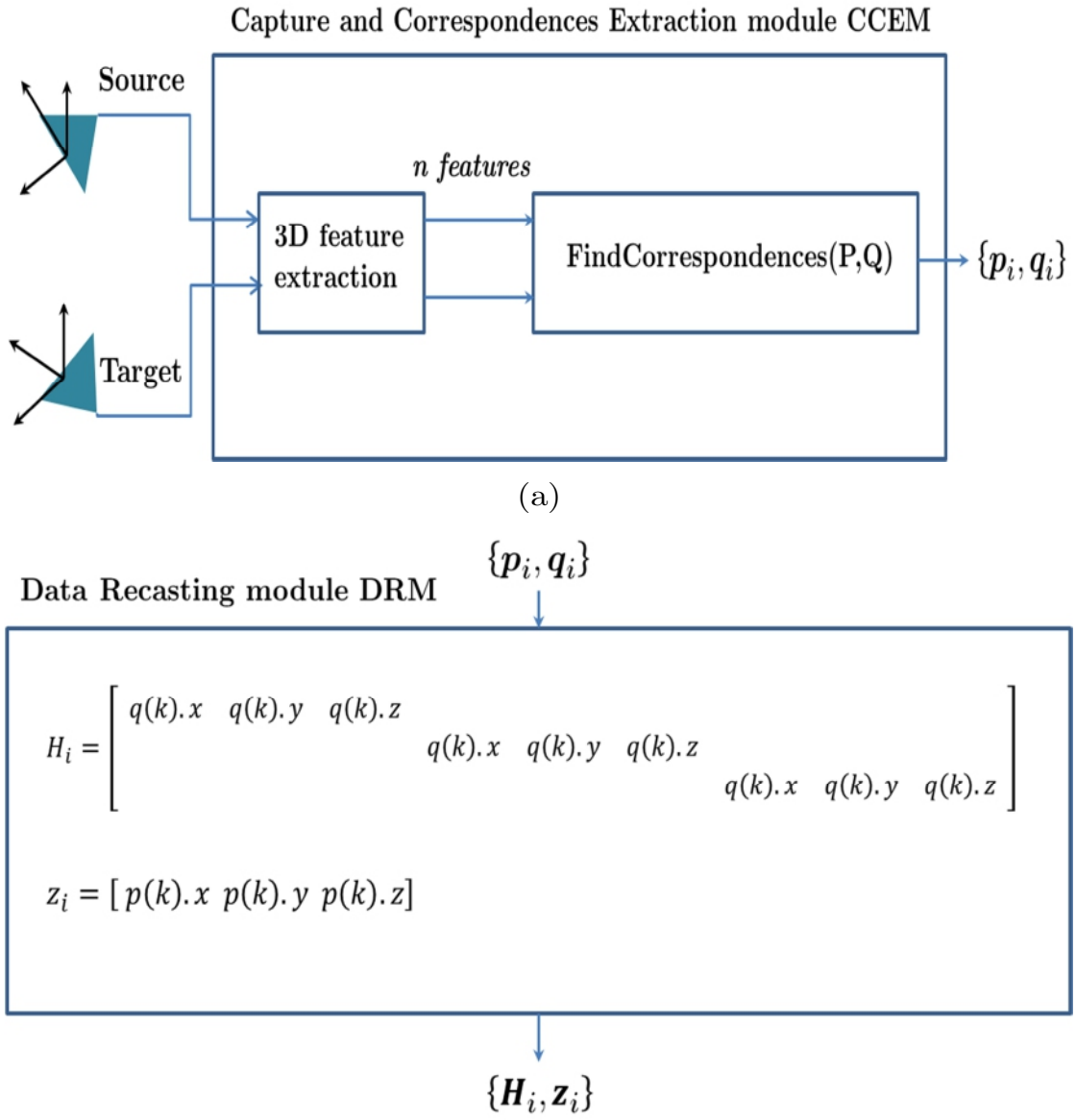
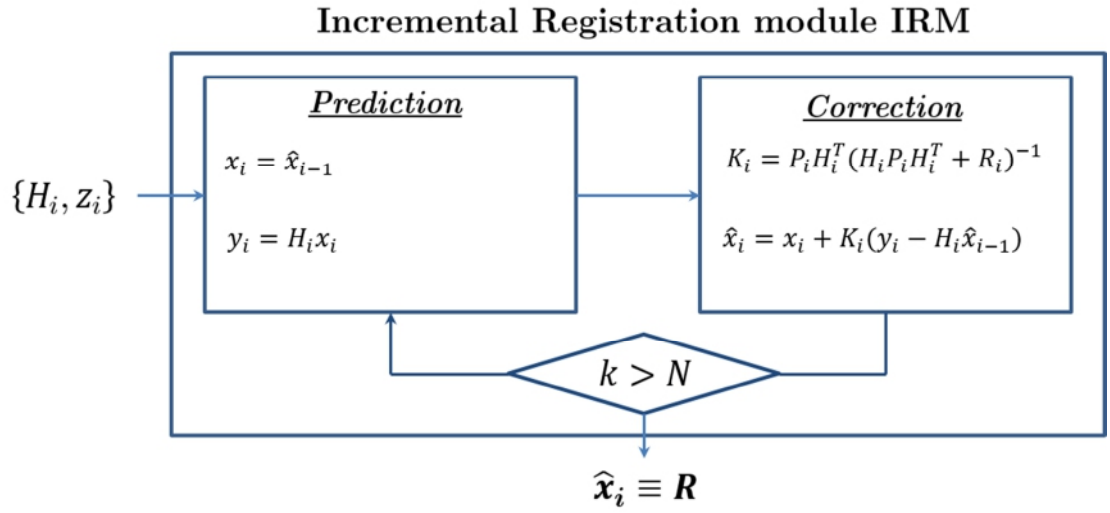
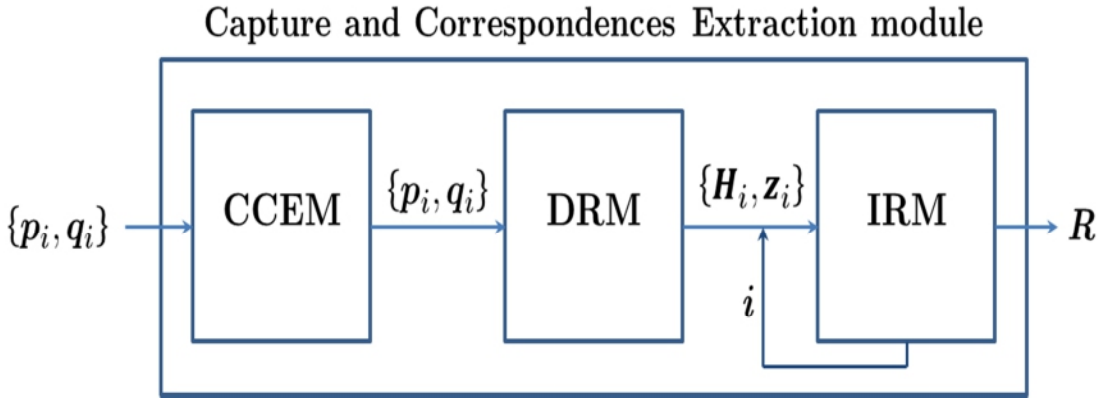


Figure 6.1 Capture and recasting module. **(a)** Capture and correspondence extraction module. **(b)** $\{\boldsymbol{p_i}, \boldsymbol{q_i}\}$ to $\{\boldsymbol{H_k}, \boldsymbol{z_k}\}$ recasting module.



(a)



(b)

Figure 6.2 3D recursive registration with Kalman filter. **(a)** Kalman filter for registration. **(b)** Global registration pipeline.

Recursive processing is convenient for problems where the datasets are substantially large. On the other hand, regular registration algorithms are proven to be very poor in coping with such scenarios [166].

The global diagram explaining the flow of data and the different stages of processing is depicted in Figure 6.2 (b).

The Kalman filter (KF) is a more general form of Recursive Least Squares optimisation. All of its parameters can be time-varying. KF was first introduced by Rudolf Kalman [167]. Since then, it has become a standard for optimal linear filtering as well as a source of inspiration for many modern, more sophisticated approaches to optimal state estimation.

In KF's scheme, see Chapter 3, $\boldsymbol{x_k} \in \mathfrak{R}^n$ represents the state vector, $\boldsymbol{y_k} \in \mathfrak{R}^m$ is the measurement or observation vector. In addition to these vectors that have been already introduced in the RLS method, the KF provides the possibility to further include an external control variable $\boldsymbol{u_k}$ that enables the filter to maintain a more general affine form (Equation (6.42)). KF equations generalise the original RLS as follow:

$$x_k = A_k \widehat{x}_{k-1} + B_k u_k + w_k \tag{6.42}$$

$$y_k = H_k x_k + v_k \tag{6.43}$$

In addition to the measurement noise $\boldsymbol{v_k}$, the KF takes into account another noise process $\boldsymbol{w_k}$, that disturbs the process of the prediction regarding the estimate $\boldsymbol{x_k}$. Both noise processes handled by the KF are assumed to be white with normally distributed random variables $\boldsymbol{v_k} \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \boldsymbol{R_k}), \boldsymbol{w_k} \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \boldsymbol{Q_k})$.
Using the recursive Equation (6.32) relating $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$ and the prior $\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}$ along with the initial value of the state $\boldsymbol{x_0}$ and the respective error in estimation $\boldsymbol{P_0}$, KF estimates optimal state vector $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$ by minimising the norm of the mean squared error $(\boldsymbol{x_i} - \hat{\boldsymbol{x}}_{\boldsymbol{i}})^{\mathbf{2}}$ recursively.

The recursive nature of the filter requires an initial value for the state $\boldsymbol{x_0}$ and its corresponding error $\boldsymbol{P_0}$. By replacing the variables $\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}$, $\hat{\boldsymbol{P}}_{\boldsymbol{k-1}}$ of Equations (6.32), (6.38) and (6.41), with their Kalman corresponding variables $\boldsymbol{x_k}$,$\boldsymbol{P_k}$. KF equations become as Equation (3.1) to (3.6).

The simplification of Equation (6.38) yields (3.6), as follows:

$$\begin{aligned}
\hat{P}_k &= (I - K_k H_k) P_k (I - K_k H_k)^T + K_k R_k K_k^T \\
&= P_k - K_k H_k P_k + K_k (H_k P_k H_k^T + R_k) K_k^T - P_k H_k^T K_k^T \\
&= P_k - K_k H_k P_k + P_k H_k^T K_k^T - P_k H_k^T K_k^T \\
&= (I - K_k H_k) P_k
\end{aligned} \tag{6.44}$$

KF is divided into:

Prediction: the previously obtained state estimate is projected forward along with its covariance matrix using filter's parameters $\boldsymbol{A_k}$,$\boldsymbol{B_k}$ and $\boldsymbol{H_k}$. The purpose of the prediction stage is the estimation of an a priori value of the state variable. The latter will be further corrected with the actual noisy measurements. Moreover, the prediction is capable of providing a relatively good estimation of the state and its covariance matrix even when no measurements are provided.

Correction: the correction is performed on the predicted estimates at the reception of the newly acquired measurements. The result is the a posteriori estimate that is theoretically the most accurate among all the results of alternative filters, provided that the system is linear and precisely modelled.

6.7 3D Registration with RLS

RLS has been established as a very useful tool for the resolution of many engineering problems. This usefulness results from its ability to fit noisy data to a known model, in both time-invariant and time-varying filtering problems. Shortly after the KF was invented, many subsequent optimal state estimation techniques were developed with more sophisticated formulations. The latter have become a standard for several applications. Nevertheless, 3D data registration solutions have profited very poorly from the assets of time-varying filters; even though the close form solutions were experiencing several weaknesses. Moreover, the authors of a number of recent image registration surveys did not allude to the possibility of solving the 3D alignment problem with any recursive filtering tools [31].

The idea of aligning 3D data by means of a recursive filtering scheme is motivated by some advantages that promote time-varying registration solutions against alternative methods. The first advantage is that recursive filtering based registration does not require all the data to be entirely available at the beginning of processing. The advantage of such a property can be particularly seen in scenarios where the amount of data is important. It can also be seen when feature extraction or matching algorithms deliver the pairs of correspondences progressively over time. Another advantage of time-varying registration is the possibility for cooperation between different active registration units working in parallel. The latter can instantaneously share their respective most updated estimates. This asset allows for various instances of registration to take advantage of each other's contributions, since some features can be more accurately seen from another viewpoint. This cooperation helps in reducing the probability of the alignment algorithm being trapped in a local minimum or being sensitive to perturbations.

6.7.1 Recursive 3D Registration Modelling

Intuitively, 3D registration is clearer to write in the native LS form that has been illustrated in Equation (6.24). To express the same problem with a recursive framework, that can fit the KF, in particular, as well as any other time-varying filter, the original problem of Equation (6.24) should be rewritten as follows:

$$p = Rq + e \tag{6.45}$$

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} + e \tag{6.46}$$

After distributing, Equation (6.46) is rewritten as a system of three equations:

$$\begin{cases} p_x = r_{11}q_x + r_{12}q_y + r_{13}q_z + e_x \\ p_y = r_{21}q_x + r_{22}q_y + r_{23}q_z + e_y \\ p_z = r_{31}q_x + r_{32}q_y + r_{33}q_z + e_z \end{cases}$$

$$\begin{cases} p_x = r_{11}q_x + r_{12}q_y + r_{13}q_z \\ p_y = \quad\quad\quad\quad\quad\quad r_{21}q_x + r_{22}q_y + r_{23}q_z \\ p_z = \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad r_{31}q_x + r_{32}q_y + r_{33}q_z \end{cases} \tag{6.47}$$

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} q_x & q_y & q_z & & & & & & \\ & & & q_x & q_y & q_z & & & \\ & & & & & & q_x & q_y & q_z \end{bmatrix} \begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \\ r_{21} \\ r_{22} \\ r_{23} \\ r_{31} \\ r_{32} \\ r_{33} \end{bmatrix} + \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} \tag{6.48}$$

$$p = H_R \mathcal{R}_9 + e \tag{6.49}$$

Equation (6.49) expresses the target feature point coordinates as a transformation $\boldsymbol{H_R}$ applied to the new vector that holds the elements of the rotation matrix $\boldsymbol{\mathcal{R}_9}$.

The advantage of this formulation is the possibility for fitting the 3D registration problem into the recursive least squares framework. In other words, instead of applying the ordinary rotation matrix $\boldsymbol{R}$ on source feature points; the matrix built upon the source data is applied on the vector containing the nine entries of the rotation matrix. The latter is estimated recursively, using the pairs of matched features. In other words, the state variable $\boldsymbol{x_k}$ represents the rotation matrix $\boldsymbol{R}$ of Equation (6.24). Subsequently, the filter uses the pairs of corresponding points, $\boldsymbol{y_k}$, embedded in the projection matrix $\boldsymbol{H_k}$ to refine the entries of the subject rotation matrix.

The formulation of the registration problem for each time-step $\boldsymbol{k}$ is modelled as follows:

- $\boldsymbol{x_k} = [\boldsymbol{r_{11}}\ \boldsymbol{r_{12}}\ \boldsymbol{r_{13}}\ \boldsymbol{r_{21}}\ \boldsymbol{r_{22}}\ \boldsymbol{r_{23}}\ \boldsymbol{r_{31}}\ \boldsymbol{r_{32}}\ \boldsymbol{r_{33}}]^{\boldsymbol{T}} \in \boldsymbol{\Re^9}$: State vector containing the nine components of the rotation matrix.

- $\boldsymbol{A_k} = \boldsymbol{I_9}, \forall \boldsymbol{k}$: Theoretically, the estimated state (rotation transformation) remains unchanged for all pairs of corresponding feature points to be aligned, as long as the latter is assumed to be rigid. In other words, the state at the current time-step should not be propagated at the reception of the new source and target feature-points linked by the same rigid body transformation.

- $\boldsymbol{B_k} = \boldsymbol{0_9}$; $\forall \boldsymbol{k}$: No control variable is required.

- $\boldsymbol{w_k} \sim \boldsymbol{\mathcal{N}}(\boldsymbol{0}, \boldsymbol{Q_k})$: Random variable representing the uncorrelated zero-mean noise process, for which $\boldsymbol{Q_k} = \boldsymbol{\sigma_k^2 I_9}$: $\boldsymbol{\sigma_k}$ should remain small as long as the noise process perturbing the system is assumed to have been accurately characterised. The latter can be anisotropic, i.e. can have different magnitudes in the different directions of the space where the state variables belong.

- $\boldsymbol{z_k} \in \boldsymbol{\mathfrak{R}^3}$: Actual noisy measurement vector, whose elements are the coordinates of the target feature point obtained a priori from feature matching module.

- $\boldsymbol{y_k} \in \boldsymbol{\mathfrak{R}^3}$: Predicted observation vector that contains the 3D position of the target feature point.

- $\boldsymbol{v_k} \sim \boldsymbol{\mathcal{N}}(\boldsymbol{0}, \boldsymbol{R_k})$: Random variable representing the uncorrelated zero-mean measurement noise, for which $\boldsymbol{R_k} = [\boldsymbol{\sigma_x^2}\ \boldsymbol{\sigma_y^2}\ \boldsymbol{\sigma_z^2}]\boldsymbol{I_3}$, the three diagonal elements represent the standard deviations regarding the three coordinates. The latter are obtained from the analysis of the properties of noise contaminating the outputs streamed by the sensor.

- $\boldsymbol{P_k}$: Estimation error covariance matrix.

- $\boldsymbol{K_k}$: Kalman gain matrix.

- $\boldsymbol{\hat{x}_k}$: Corrected estimate at time-step $\boldsymbol{k}$.

- $\hat{\boldsymbol{P}}_{\boldsymbol{k}}$: Corrected covariance matrix of error in $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$.

Both KF-based registration and state of the art ICP algorithms compute a rigid body transformation that aligns the source point cloud with the target one by the minimisation of the L_2 norm of alignment error. However, the time-varying solution has the advantage of working progressively at the pace of data delivery as well as the ability to handle errors corrupting 3D point's positions in an effective manner. The complete algorithm for Kalman-based registration is explained in Algorithm 6.2.

The algorithm works as follows:

- Initialise the state vector (rotation matrix elements) with the identity matrix entries. Alternatively, if available, a preliminary guess at its value as well as the covariance matrix. The latter allows one to weight feature points.

- Iterate over feature points; acquire the next measurement $\boldsymbol{z}_{\boldsymbol{k}}$ from the target features and build the projection matrix $\boldsymbol{H}_{\boldsymbol{k}}$ from the coordinates of the source features.

- KF starts with the prediction of the next estimate along with its covariance matrix. The system equation regarding the expected transformation expresses the fact that the correct transformation should not evolve during the scanning of the list of the feature points. For instance, they are all related by the same transformation.

- Afterwards, comes the correction stage where the first step is the computation of the Kalman gain $\boldsymbol{K}_{\boldsymbol{k}}$ from the predicted covariance matrix $\boldsymbol{P}_{\boldsymbol{k}}$, $\boldsymbol{H}_{\boldsymbol{k}}$ and the covariance matrix of measurement noise $\boldsymbol{R}_{\boldsymbol{k}}$.

- The estimate $\boldsymbol{x}_{\boldsymbol{k}}$ and the covariance of error in estimation $\boldsymbol{P}_{\boldsymbol{k}}$ are therefore corrected with $\boldsymbol{K}_{\boldsymbol{k}}$.

The innovation term $(\boldsymbol{y_k} - \boldsymbol{H_k}\hat{\boldsymbol{x}}_{\boldsymbol{k-1}})$ is computed upon the difference between the measured feature position $\boldsymbol{z_k}$ and the predicted position resulting from the current state estimate (transformation) as well as the projection matrix formed by the entries of source feature points coordinates $\boldsymbol{H_k x_k}$. In addition, the prediction covariance matrix of the error in estimation, $\boldsymbol{P_k}$, is computed from the previous covariance matrix and the constant one of noise process disturbing the system $\boldsymbol{Q_k}$.

The computational complexity of the KF-based registration is proportional to $\boldsymbol{O(n \times 9^3) = O(n \times 729)}$ in the worst case. $\boldsymbol{n}$ is the number of feature points used to compute the optimal registration and 12 is the maximum size of the filter's state vector. This relatively high computational load can be reduced with matrix-computation optimisation approaches such as CW-like algorithms [168].

By doing so, the previous complexity of computation can be reduced to $\boldsymbol{O(n \times 9^{2.37}) \cong O(n \times 182.622)}$. On the other hand, the complexity regarding the state of the art registration approaches is proportional to $\boldsymbol{O(n^2 \times 9^{2.37}) \cong O(n^2 \times 182.622)}$. In other words, due to the quadratic complexity of the classical registration methods, they can only be suitable for reduced size alignment problems. This consideration is important to enable the algorithm to output a good-quality rigid body transformation in a reasonable time. The larger datasets are more effectively aligned with linear-complexity algorithms such as the proposed KF-based solution.

From Algorithm 6.2, one can easily append the three components of the translation vector to the projection matrix $\boldsymbol{H_k}$ along with the scale factor between the source and the target point clouds in the following manner:

$$\begin{bmatrix} p \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} q \\ 1 \end{bmatrix} + \begin{bmatrix} e \\ 1 \end{bmatrix} \tag{6.50}$$

$$\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix} + \begin{bmatrix} e \\ 1 \end{bmatrix} \tag{6.51}$$

In the formulation of Equations (6.50), (6.51), the complexity of the filter becomes proportional to $\boldsymbol{O(n \times 12^3) = O(n \times 1728)}$ in the worst case. After applying the computational optimisations, the previous complexity decreases down to $\boldsymbol{O(n \times 12^{2.37}) \cong O(n \times 361.126)}$. When compared to KF-based registration, ordinary registration approaches achieve the same result with a complexity proportional to $\boldsymbol{O(n^2 \times 12^{2.37}) \cong O(n^2 \times 361.126)}$.

Algorithm 6.2 Kalman-based registration

Source and target point clouds

$\boldsymbol{P}$, $\boldsymbol{Q}$*: 3D feature points;*

$[\boldsymbol{p_i}, \boldsymbol{q_i}] = \emptyset$*: Correspondences list;*

Find 3d correspondences in the two point clouds

$[\boldsymbol{p_i}, \boldsymbol{q_i}]$ = *FindCorrespondences(P,Q);*

$\hat{\boldsymbol{x}}_{\mathbf{0}} = [1.0\ 0.0\ 0.0\ 0.0\ 1.0\ 0.0\ 0.0\ 0.0\ 1.0]^T$

$\boldsymbol{Q} = \sigma^2 I_9$

$\hat{\boldsymbol{P}}_{\mathbf{0}} = \boldsymbol{Q}$

$\boldsymbol{R}_{\mathbf{0}} = [\boldsymbol{\sigma}_x^2\ \boldsymbol{\sigma}_y^2\ \boldsymbol{\sigma}_z^2] I_3$

for *each pair of correspondences* ***(k=1, n)***

$z_k = [\ p(k).x\ \ p(k).y\ \ p(k).z]$

$$H_k = \begin{bmatrix} q(k).x & q(k).y & q(k).z & & & & & & \\ & & & q(k).x & q(k).y & q(k).z & & & \\ & & & & & & q(k).x & q(k).y & q(k).z \end{bmatrix}$$

Prediction

$x_k = \hat{x}_{k-1}$

$y_k = H_k x_k$

$P_k = \hat{P}_{k-1} + Q$

Correction

$K_k = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1}$

$\hat{x}_k = x_k + K_k(z_k - y_k)$

$\hat{P}_k = (I - K_k H_k) P_k$

end

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} q_x & q_y & q_z & & & & & & & 1 & & \\ & & & q_x & q_y & q_z & & & & & 1 & \\ & & & & & & q_x & q_y & q_z & & & 1 \end{bmatrix} \begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \\ r_{21} \\ r_{22} \\ r_{23} \\ r_{31} \\ r_{32} \\ r_{33} \\ t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} \tag{6.52}$$

$$p = H_{R,t}\mathcal{R}_{12} + e \tag{6.53}$$

Equation (6.53) expresses the target feature point coordinates as a transformation $\boldsymbol{H_{R,t}}$ applied to the new vector that holds the twelve elements of the rigid body $[\boldsymbol{R}, \boldsymbol{T}]$ matrix $\boldsymbol{\mathcal{R}_{12}}$.

Although the KF can be very useful, its projection matrices $\boldsymbol{H_R}, \boldsymbol{H_{R,t}}$ are defined by the noisy inputs of source point cloud. This noise factor negatively impacts the stability of estimation. The filter therefore responds reciprocally to the quality of the captured feature points. In other words, if the points are accurately captured, the respective projection matrix regarding the filter will be accurate. Alternatively, if the 3D input data is noisy, then the corresponding projection matrix becomes inaccurate.

The robust time-varying filters are able to guarantee a certain level of robustness of estimation in order to prevent unreliable inputs from significantly biasing resulting estimates. To this end, such a family of filters uses knowledge of how uncertain the parameters are in order to estimate a stable and reliable mean value for the state of the system with a reduced estimation error covariance. The next approach proposed to overcome the instability in the 3D registration with time-varying filters is based on the adaptation of the Robust H_∞ filter to the 3D registration problem. Nonetheless, it is necessary to

quantify the amount of the global uncertainty characterising RGBD sensor's outputs before proceeding through the actual registration.

6.8 3D Points Uncertainty

It has been shown in the previous section that the 3D registration problem is easier to express with measurement noise metrics alone (covariance matrices). The latter can be straightforwardly embedded in the covariance matrix of time-varying filters. Nevertheless, the parameters of the filter are computed from noisy point cloud data, which itself changes from one frame to another. For this reason, the filter becomes very sensitive and highly dependable on the accuracy of sensory outputs.

To handle the instability of the resulting estimation, the intervals of uncertainties affecting the parameters of the filter should be confined. These parameters are recast in the projection matrix $\boldsymbol{H_R}$. In other words, the behaviour of the noisy outputs delivered by the sensor (Microsoft Kinect) should be thoroughly studied. Yet, the same concepts remain applicable for alternative 3D sensors.

The uncertainties are modelled empirically by looking at how the 3D points are distributed, and how the camera senses the real world. Up until now, several noise cancellation approaches have been proposed to smooth RGBD data. Likewise, the appropriate smoothing technique that has been developed in Chapter 3 enables a precise tracking of the 3D features to produce an optimal 3D structure of the scene.

6.8.1 RGBD Camera z-Resolution

Before building a model for uncertainties affecting the outputs, the resolution of the depth map generated by the camera should be closely studied. The camera was pointed parallel to a large flat wall as has been shown in Figure 3.4 (a). That setup permits the capture of a point cloud covering the whole operating range regarding the sensor. The depth resolution is inversely proportional to the

distance from the device. More importantly, the points within the captured data are dispersed over parallel clusters that were named $Z-Levels$. Every level constitutes a partition in the whole point cloud.

6.8.2 Depth Noise Statistics

It has already been demonstrated in Chapter 3 that noise process in the Kinect has a Gaussian distribution with varying standard deviations. These standard deviations rely on the range between the sensor and the scene. The standard deviation $\boldsymbol{\sigma_k}$ of a given Z-level $\boldsymbol{z_k}$ is defined with how far the level $\boldsymbol{z_k}$ is from the camera plane. These statistical parameters can be obtained from Equation (3.11).

$\boldsymbol{I_k}$ is a set of indices used to identify the different Z-levels. $\boldsymbol{\sigma_{z_k}}$ of Equation (3.11) represents the average distance separating the two boundaries of the interval $[\boldsymbol{i-1}, \boldsymbol{i+1}]$ and the central level $\boldsymbol{z_k}$ to which belongs to the sampled point. Empirically, the best results are reached with $\boldsymbol{i=3}$. That is, the true depth $\hat{\boldsymbol{Z}}_{\boldsymbol{k}}$ at every pixel is expected to be equal to $\boldsymbol{Z_k} \pm ((\boldsymbol{Z_{k+3}} - \boldsymbol{Z_{k-3}}) / \boldsymbol{2})$.

The standard deviations concerning the remaining two coordinates $(\boldsymbol{x_k}, \boldsymbol{y_k})$ are deduced from the intrinsic parameters of the camera $(\boldsymbol{f_x}, \boldsymbol{f_y}, \boldsymbol{c_x}, \boldsymbol{c_y})$ and the standard deviation of the depth measurements $\boldsymbol{\sigma_{z_k}}$. Their respective equations are as follows:

$$\begin{cases} u_i = (f_x / z_i) x_i + c_x \\ v_i = (f_y / z_i) y_i + c_y \end{cases} \tag{6.54}$$

$$\begin{cases} x_i = (z_i / f_x)(u_i - c_x) \\ y_i = (z_i / f_y)(v_i - c_y) \end{cases} \tag{6.55}$$

$$\begin{cases} \sigma_{z_k} = 0.5\ (\ Z_{k+i} - \ Z_{k-i}) \\ \sigma_{x_k} = (\sigma_{z_k} / f_x)(u_k - c_x) \\ \sigma_{y_k} = (\sigma_{z_k} / f_y)(v_k - c_y) \end{cases} \tag{6.56}$$

From Algorithm 6.1, it is plausible that the coordinates of feature points are the basis for the computation of the projection matrix $\boldsymbol{H_k}$. However, every point is affected by a certain amount of noise characterised by the standard deviations $\boldsymbol{\sigma_{x_k}}, \boldsymbol{\sigma_{y_k}}, \boldsymbol{\sigma_{z_k}}$ towards the directions of the axes $\boldsymbol{x}, \boldsymbol{y}$ and $\boldsymbol{z}$, respectively. Taking this into account, a covariance matrix is associated with every feature in order to describe its uncertainty. As can be seen in Figure 6.3 (a), each covariance matrix (equation (6.56)) can be represented by an ellipsoid whose principal axes' lengths are the respective standard deviations.

Knowledge about the quality of measurements is used to feed a robust filtering scheme that has the ability to deal with the reduced accuracy to deliver a more resilient estimation. The robustness is therefore seen in the stability of the resulting estimated 3D transformation over different feature data with varying levels of accuracy. The result consequently becomes less sensitive to the quality of inputs.

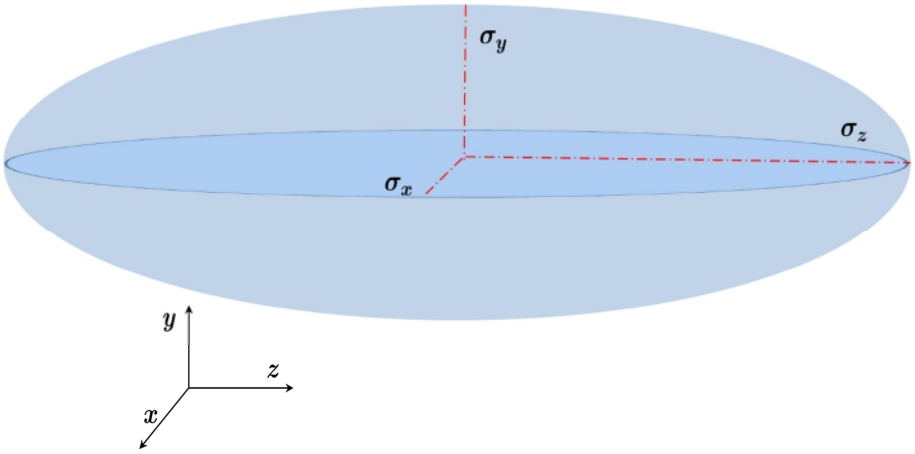
Using the standard deviations $\boldsymbol{\sigma_{x_k}}, \boldsymbol{\sigma_{y_k}}, \boldsymbol{\sigma_{z_k}}$, the covariance matrix can be quantified as follows:

$$C(x, y, z) = \begin{bmatrix} \sigma_x^2 & \sigma_x\sigma_y & \sigma_x\sigma_z \\ \sigma_y\sigma_x & \sigma_y^2 & \sigma_y\sigma_z \\ \sigma_z\sigma_x & \sigma_z\sigma_y & \sigma_z^2 \end{bmatrix} \tag{6.57}$$

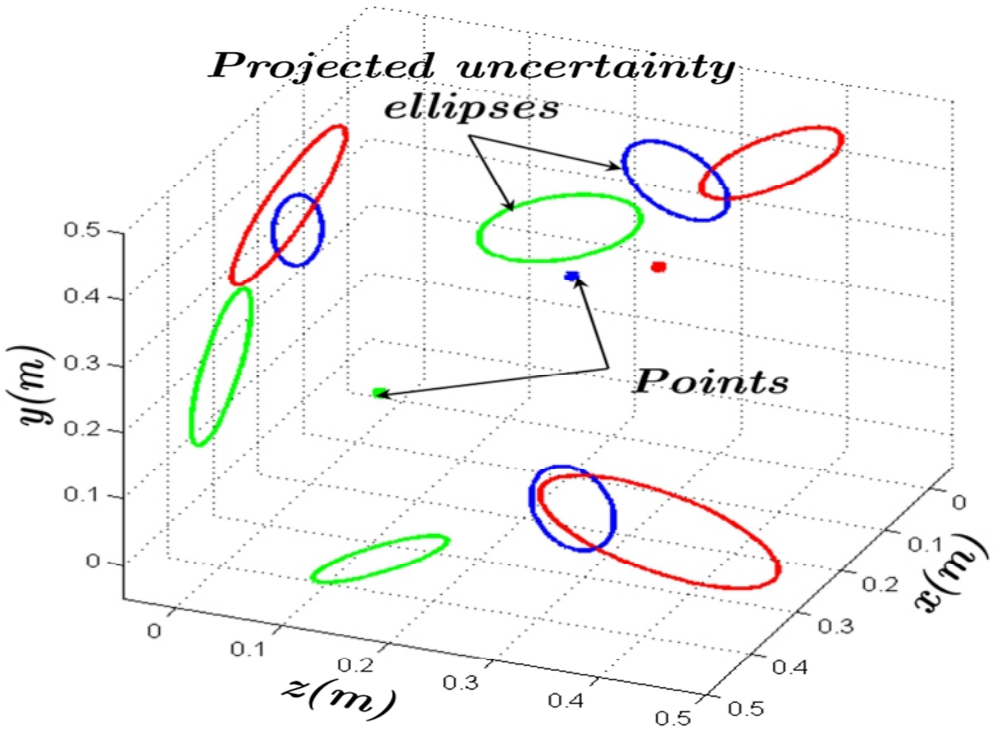
Every point $\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ has a covariance matrix $\boldsymbol{C}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ representing the spread of uncertainty in the three axial directions. An example of three points in space is illustrated in Figure 6.3 (b). The projection of covariance ellipsoids on the planes $\boldsymbol{zx}, \boldsymbol{zy}, \boldsymbol{yx}$ produces three ellipses, Figure 6.4 (a), (b), (c). The volume of the ellipsoid is proportional to the norm of the covariance matrix, which in turn, is proportional to the uncertainty itself. Ellipses lying in the three principal planes explain the diffusion of the uncertainty in their respective directions. The more accurately a feature point is captured, the smaller the norm of its covariance matrix will be (blue point in Figure 6.3 (b)). Likewise,

the less accurate the capture of a given feature, the larger the norm of its covariance matrix (red point in Figure 6.3 (b)).

Kanazawa et al. [169] claimed that the incorporation of the error in the estimation of pose between features does not contribute any further improvements to the final result. On the other hand, Brooks et al. [170] as well as the author of this manuscript in a previous work, both noticed a reduced error in estimation after considering the uncertainty. Based on the conducted experiments with the registration algorithms and the fact that WICP (Weighted ICP) outperforms ICP, as the results will show, it is evident that the incorporation of uncertainty in a feature's location improves the estimation of the relative pose between point clouds. In addition, there are practical tools in optimal state estimation theory that can be used to reduce the perturbations caused by the uncertain data. The proposed registration scheme follows Brooks' findings and uses the knowledge about feature point's uncertainty to recast the problem of the point cloud alignment into a time-varying robust H_∞ filter. The latter is able to guarantee a more precise and robust registration that cannot be otherwise ensured with most of the known approaches in the 3D registration literature. As will be shown in the results, the robust algorithm will be tested against the naive Kalman filter based registration algorithm as well as several other registration methods such as WICP [171], EMICP [172] and Horn's absolute rotation with quaternions.

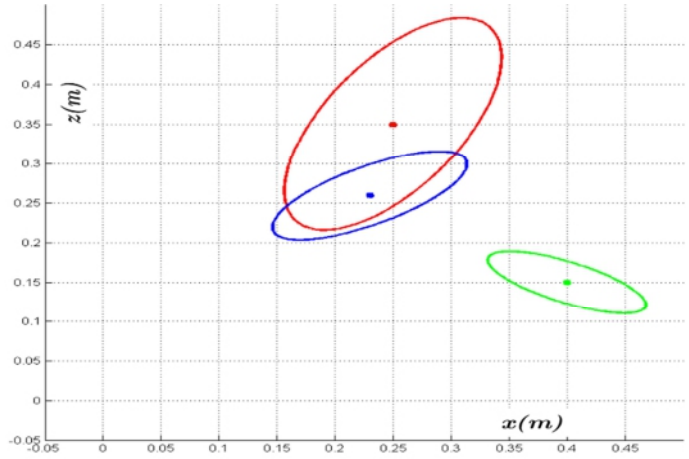


(a)

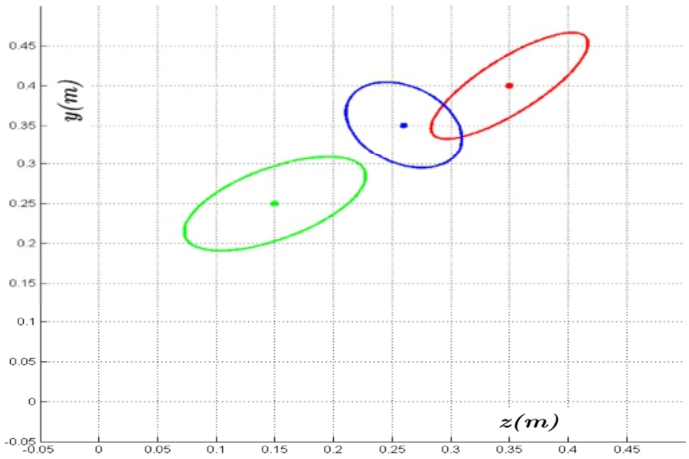


(b)

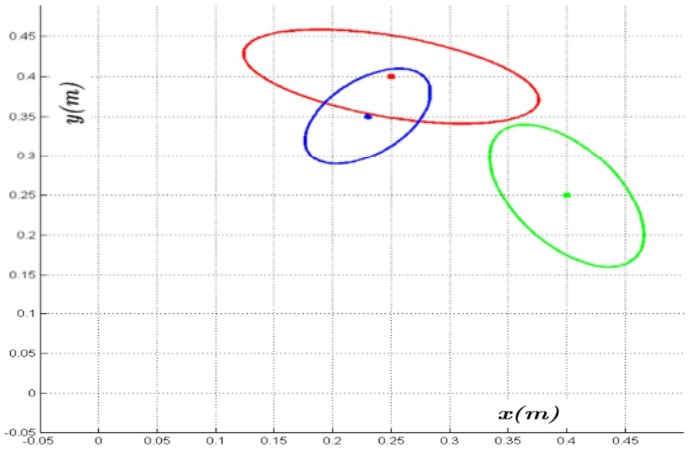
Figure 6.3 3D point's uncertainty**. (a)** Uncertainty ellipsoid. **(b)** Projections of uncertainty ellipsoids on the three principal planes (***xy***, ***yz***, ***zx***), where the three red ellipses correspond to the red point, and vice versa for the blue and the green points.



(a)



(b)



(c)

Figure 6.4 2D projections on the cardinal planes. **(a)** xz. **(b)** zy. **(c)** xy

6.9 Robust H_∞ Filter for 3D Registration

In Section 6.7, the possibility of solving the registration problem with the recursive least squares algorithm was proven. After the study of measurement uncertainties, it is now possible to estimate changing rigid body motions over time. The last estimated motion is supposed to align the source and the target point clouds robustly.

The proposed registration framework is a variant of the formulation of time-varying recursive registration. It incorporates modelling and measurement uncertainties. This formulation begins from Equation (6.48) as follows:

$$
\begin{aligned}
H_R = & \\
& \begin{bmatrix} q_x+\sigma_x & q_y+\sigma_y & q_z+\sigma_z & & & & & & \\ & & & q_x+\sigma_x & q_y+\sigma_y & q_z+\sigma_z & & & \\ & & & & & & q_x+\sigma_x & q_y+\sigma_y & q_z+\sigma_z \end{bmatrix} \\
& = \begin{bmatrix} q_x & q_y & q_z & & & & & & \\ & & & q_x & q_y & q_z & & & \\ & & & & & & q_x & q_y & q_z \end{bmatrix} + \\
& \qquad \begin{bmatrix} \sigma_x & \sigma_y & \sigma_z & & & & & & \\ & & & \sigma_x & \sigma_y & \sigma_z & & & \\ & & & & & & \sigma_x & \sigma_y & \sigma_z \end{bmatrix}
\end{aligned}
\tag{6.58}
$$

The compact form, therefore, becomes:

$$
y_k = (H_R + \Delta H_R) x_k \tag{6.59}
$$

$\boldsymbol{\Delta H_R}$ is the uncertainty impinging on the model, and $\boldsymbol{y_k}$ is the predicted measurement. The difference between the latter and the measured target feature location $(\boldsymbol{z_k} - \boldsymbol{y_k})$ is the actual innovation contributed by the filter.

In addition to measurement uncertainty, $\boldsymbol{\Delta H_R}$, the robust H_∞ filter takes into account process-modelling uncertainties as well. The equations with the same state and measurement variables $\boldsymbol{x_k}, \boldsymbol{y_k}$ as in Equations (6.42), (6.43) are:

$$\begin{aligned} x_k &= (A_k + \Delta A_k)\widehat{x}_{k-1} + B_k u_k + w_k \\ y_k &= (H_k + \Delta H_k)x_k + v_k \\ \sigma_A &= [\sigma_{r_{11}}\sigma_{r_{12}}\sigma_{r_{13}}\sigma_{r_{21}}\sigma_{r_{22}}\sigma_{r_{23}}\sigma_{r_{31}}\sigma_{r_{32}}\sigma_{r_{33}}] \end{aligned} \tag{6.60}$$

$$\Delta A_k = \sigma_A I \tag{6.61}$$

$\boldsymbol{\Delta A_k}$ is a diagonal matrix (see Equation (6.61)) whose entries are the standard deviations of the incremental alignment error that represents the spread of error in the directions of the available degrees of freedom.

If one is only interested in the estimation of the rotation matrix, diagonal elements $\boldsymbol{\sigma_{r_{ii}}}$ are initially set to one. The latter are subject to change when the algorithm progresses in time. The most optimal transformation is attained when $\boldsymbol{\sigma_A} \approx \mathbf{0}$.
The two matrices $\boldsymbol{\Delta A_k}$ and $\boldsymbol{\Delta H_k}$ are, therefore, system and measurements uncertainties. If these two matrices are not available they can be assumed to have the form of Equation (5.17).

The adaptation of the Robust H_∞ filtering scheme is shown to be flexible and capable of delivering accurate state estimations even with uncertain system parameters. The estimation error compared to the ground truth measurements will show the effectiveness of the contributed approach against alternative recursive filters such as Kalman and the more established registration solutions available in the literature. These tools and many others do not consider the uncertainties in the parameters of their respective systems. If the parameters are accurate, then the robust registration performs as efficiently as KF. However,

when the system is not precisely characterised, the naive algorithms do not possess sufficient resilience to reliably handle the unstable parameters. In real scenarios, the exact model is very unlikely to be determined [127]. The robust registration combines the robustness of H_∞ (it is less affected by the accuracy of system's parameters) and the optimality of Kalman filtering on linear systems.

6.10 Results & Discussions

In this section, the results regarding the recursive filter-based registration solution are validated with tests on synthetic and real 3D data. For instance, the algorithms that have been compared to the implementation of the Kalman and the Robust H_∞ solutions for registration are:

- The Weighted-ICP [171] (WICP): Several approaches have been contributed to assign the weightings to the 3D points. The two main perspectives are: a constant weighting based on the distance separating the two points belonging to the same pair of features; alternatively, a varying weighting based on sensor's noise. Point's measurement uncertainty is converted into a covariance matrix as has been shown in Section 6.8. The latter will be used to rank the quality of points.

- Expectation Maximisation ICP algorithm [172] (EMICP): The authors of this algorithm investigated the registration problem on 3D points sampled from surface data. They suggested a general Maximum-Likelihood (ML) estimation of the rigid body transformation between the sets of points. They claimed, that if the Gaussian noise is assumed, the ML estimator is equivalent to ICP with the Mahalanobis distance. After they had considered the matches as a hidden variable, they obtained a marginally more compound criterion that could be adequately resolved using Expectation Maximisation (EM) tools. If Gaussian noise is assumed, their proposed method is equivalent to ICP with various matches biased by the normal-

ised Gaussian weightings. The last mentioned is computed from covariance matrices that define the spread of uncertainty around the sample.

- Horn's closed form solution based on quaternions [34] (Horn): The author proposed a closed-form solution to the least squares problem of Equation (6.3). The closed form property is due to the fact that no iterations are required. Its advantage is therefore the possibility of obtaining an optimal estimation in a single step. The triviality of the solution is due to the easiness in computing the eigenvector of a symmetric matrix associated with the most positive eigenvalue. The entries in this matrix result from the summation of the products of the corresponding points coordinates. This operation is computationally proportional to $\boldsymbol{O}(\boldsymbol{n})$ where $\boldsymbol{n}$ is the number of points. Another advantage is that no initial guess is necessary for the algorithm to work. Nevertheless, in the presence of measurement noise these advantages will no longer be verified and additional iterations are necessary after every estimation to refine the final result. The author used the unit quaternion vector instead of the normal rotation matrix to represent the rotation. Such a representation is robust against the Gimbal lock problem [173].

Since the level of accuracy reached with any given algorithm may not be achievable by another one, the behaviours of the different algorithms are analysed thoroughly by computing the RMSE and the time taken to deliver the result.

RMSE measurement resides in the distance separating the target and the transformed source point clouds. The new set of points is the outcome of the application of transformation obtained from alignment on the initial source. $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ coordinate's distance between the components of the two point clouds is calculated, as well as the overall distance separating all the points together.

In order to fairly assess every solution, processing time elapsed to find the best pose between the two sets of points is also studied. Throughout the

experiments, it is noticeable that the plotted metrics (RMSE and processing time) are not homogeneous. For this reason, a logarithmic scale has been used to cope with the difference of scale on the same graph. The error graphs regarding Kalman and the Robust H_∞ have a higher tendency to stay smaller in magnitude than EMICP, WICP and Horn. However, EMICP, WICP and the Robust H_∞ are the most time-consuming algorithms. Whereas, Kalman and Horn tend to be quicker.

Only the RMSE for 30 samples from the whole 1000 samples that have been tested will be plotted to avoid overloading the graphs with too many results. The number of features extracted from every point cloud is about 400 points. This choice is motivated by the fact that after removing outliers and false matches, average sized point cloud in a single frame contains up to 400 useful features. In addition, the implication of every single element within the cloud alignment process will significantly increase processing time without a remarkable improvement in the quality of the registration. The computation time has been computed for the five algorithms running in an $i7-2670QM$ working at $2.2GHz$, with 12.0 GB of RAM.

6.10.1 Synthetic Data

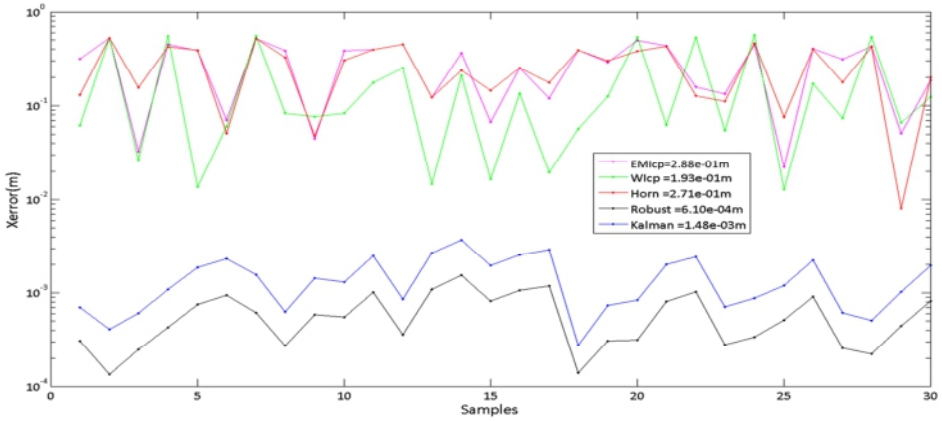
Different sets of 3D points have been generated, where $\boldsymbol{Q_i}$ (source) as well as a random 3D rigid transformations $[\boldsymbol{R_i}, \boldsymbol{T_i}]$. Based on the latter, a set of target 3D points, i.e. $\boldsymbol{P_i} = \boldsymbol{R_i}\,\boldsymbol{Q_i} + \boldsymbol{T_i}$ is built. To realistically simulate the physical data, a normally distributed anisotropic white noise is added to the clean datasets. The latter has different magnitudes $\boldsymbol{\sigma_i}$, large ($20\ mm \leq \boldsymbol{\sigma_i} \leq 80\ mm$), average ($10\ mm \leq \boldsymbol{\sigma_i} \leq 20\ mm$) and small ($0.1\ mm \leq \boldsymbol{\sigma_i} \leq 10\ mm$). The datasets therefore become $\boldsymbol{Q_i} \leftarrow \boldsymbol{Q_i} + \boldsymbol{e_i'}$ and $\boldsymbol{P_i} \leftarrow \boldsymbol{P_i} + \boldsymbol{e_i}$.

For each one of the three noise levels, 1000 point clouds in a volume ranging from $0.5{\times}0.5{\times}0.5\ \boldsymbol{m^3}$ to $10{\times}10{\times}10\ \boldsymbol{m^3}$ are generated to test the performance of algorithms. The latter (algorithms) have been tested on every single sample in order to compare their respective performances.

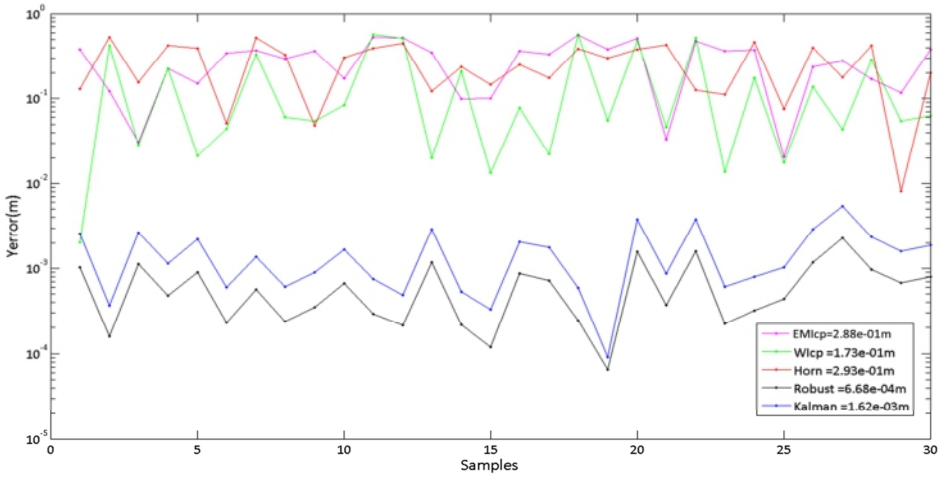
6.10.1.1 Scenario 1: Small Noise Magnitude

RMSE

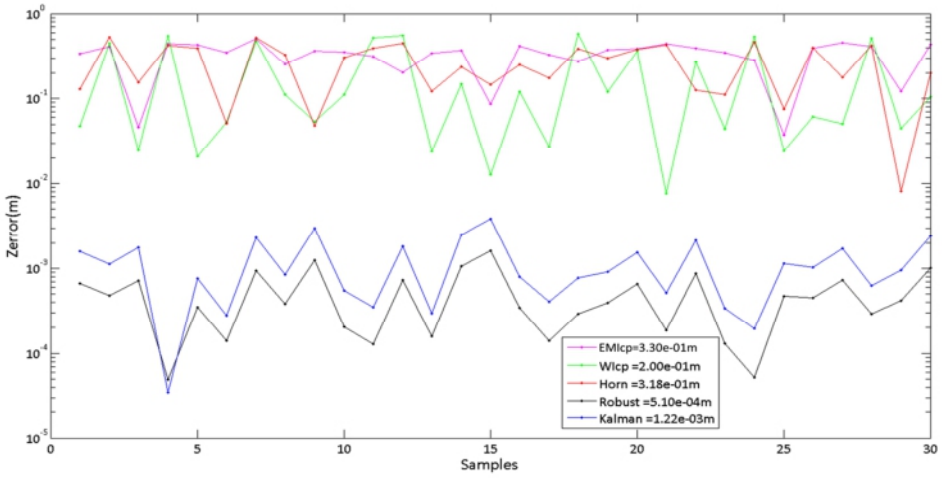
The average RMSE for EMICP (pink line in Figure 6.5 (a, b, c, d)) was $(288, 288, 330)$mm for the single components of the triplet $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. The overall error was 302mm. WICP RMSE (green line in Figure 6.5 (a, b, c, d)) was $(193, 173, 200)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. The overall error reached 189mm. Horn RMSE (red line in Figure 6.5 (a, b, c, d)) was $(271, 293, 318)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ separately. The overall error was 294mm. Robust H_∞ RMSE (black line in Figure 6.5 (a, b, c, d)) was $(0.61, 0.668, 0.51)$mm for the triplet $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. The total error was 0.596mm. Kalman RMSE (blue line in Figure 6.5 (a, b, c, d)) was $(1.48, 1.62, 1.22)$mm for the triplet $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, and its global error was 1.44mm.


Xerror(m)
Samples
EMIcp=2.88e-01m
Wlcp =1.93e-01m
Horn =2.71e-01m
Robust =6.10e-04m
Kalman =1.48e-03m

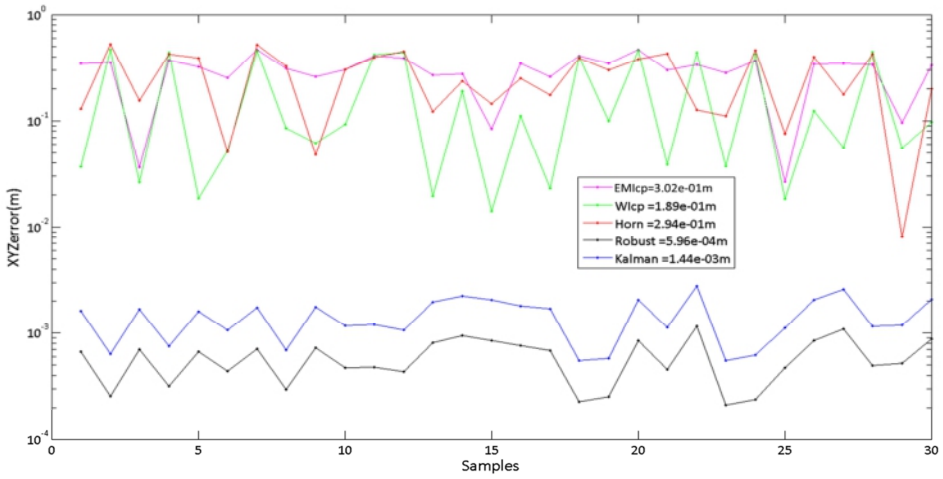
(a)


Yerror(m)
Samples
EMIcp=2.88e-01m
Wlcp =1.73e-01m
Horn =2.93e-01m
Robust =6.68e-04m
Kalman =1.62e-03m

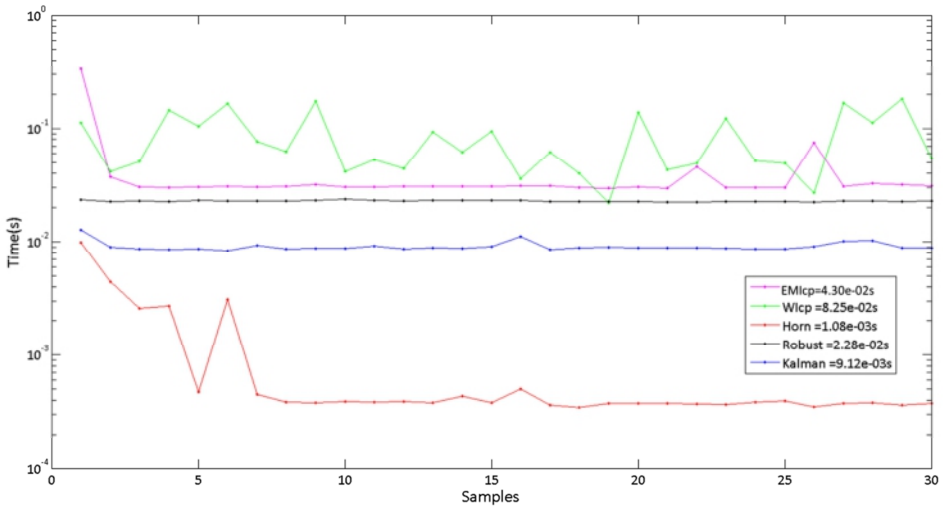
(b)


Zerror(m)
Samples
EMIcp=3.30e-01m
Wlcp =2.00e-01m
Horn =3.18e-01m
Robust =5.10e-04m
Kalman =1.22e-03m

(c)



(d)



(e)

Figure 6.5 Synthetic data with *small* noise magnitude. **(a)** $\boldsymbol{x}$ RMSE. **(b)** $\boldsymbol{y}$ RMSE. **(c)** $\boldsymbol{z}$ RMSE. **(d)** $\boldsymbol{xyz}$ RMSE. **(e)** Processing time

Although the knowledge of the correspondences between features helps enormously in preventing local minima, the disparity in performance between the proposed filtering based approach and state of the art algorithms is noticeable. As feature localisation is inherently contaminated with noise, the resulting alignment is not very well refined with EMICP, Horn and less significantly WICP. On the other hand, filter-based solutions have the ability to

cope with the noisy measurements; for this reason they have marked a higher accuracy, with the best result observed with the Robust H_∞ filter.

Processing time

As shown in Figure 6.5 (e), the average processing time taken by EMICP (pink line) is 43ms (23 FPS). This frequency is steady for most samples. With WICP (green line) the processing takes on average 82.5ms (12 FPS). This duration is fluctuating in the interval [50,120]ms. For Horn (red line) the processing takes on average 1.08ms (925 FPS). Eventually, both solutions have a very stable processing time of $(22.8, 9.12)$ms or $(43, 109)$ FPS for Robust H_∞ and Kalman respectively.

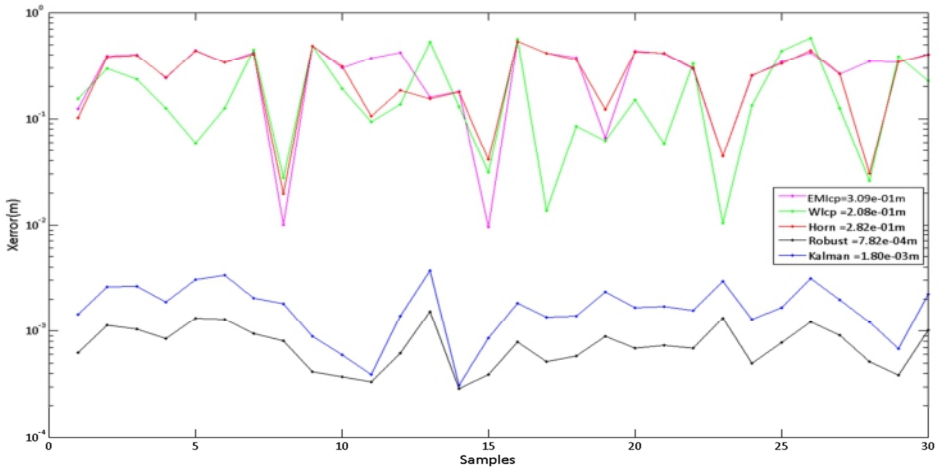
6.10.1.2 Scenario 2: Average Noise Magnitude

RMSE

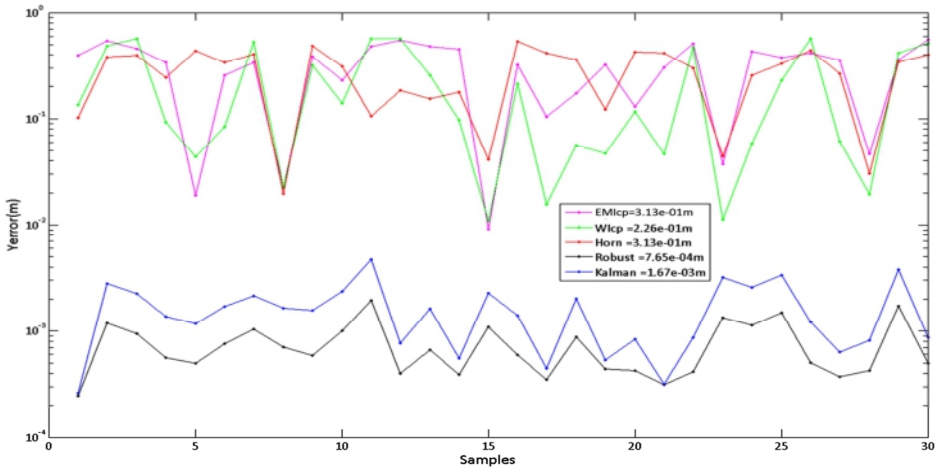
The average RMSE for EMICP (pink in Figure 6.6 (a, b, c, d)) was $(309, 313, 335)$mm for the triplet $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. The overall error increased slightly from 302mm (small magnitude scenario) to 319mm. WICP RMSE (green in Figure 6.6 (a, b, c, d)) was $(208, 226, 246)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, respectively. The overall error increased with about 30mm to reach 227mm. Horn RMSE (red in Figure 6.6 (a, b, c, d)) was $(282, 313, 330)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. The total error increased from 294mm to 309mm. Robust H_∞ RMSE (black in Figure 6.6 (a, b, c, d)) was $(0.78, 0.765, 0.66)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, respectively. The overall error increased mildly with less than 0.15mm to reach 0.73mm. Kalman RMSE (blue in Figure 6.6 (a, b, c, d)) was $(1.8, 1.67, 1.43)$mm for the triplet $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. Just like the Robust H_∞, overall Kalman RMSE increased also by less than 0.2mm to reach 1.63mm.

Processing time

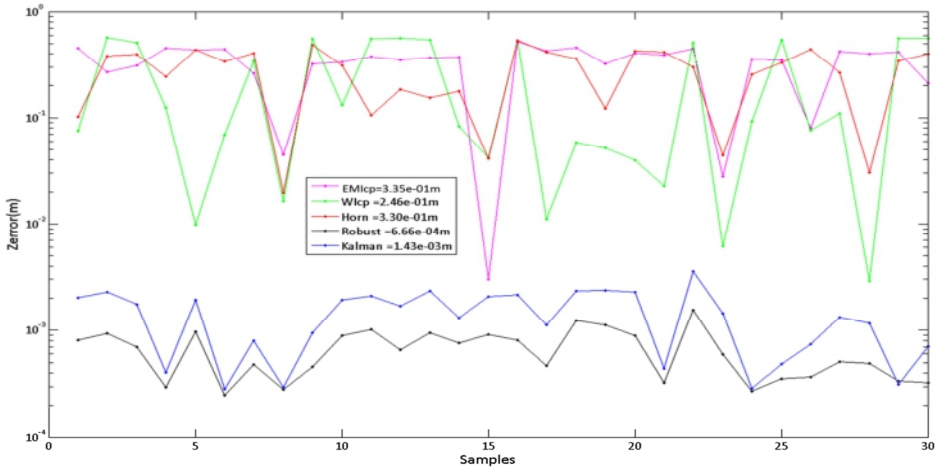
As shown in Figure 6.6 (e), the average processing time taken by EMICP (pink) is 44ms (23 FPS). This frequency is steady for most samples. For WICP (green) the processing takes an average period of 72.5ms (13 FPS). The latter is


Xerror(m)
Samples
EMIcp=3.09e-01m
Wlcp =2.08e-01m
Horn =2.82e-01m
Robust =7.82e-04m
Kalman =1.80e-03m

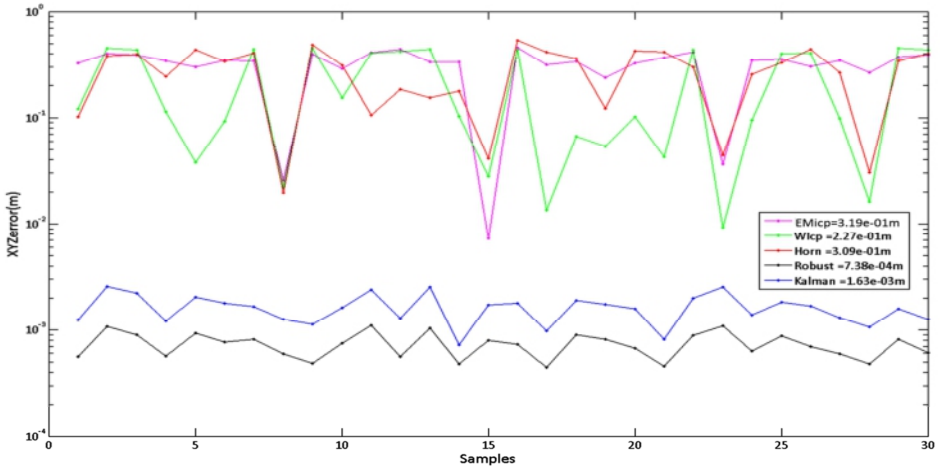
(a)


Yerror(m)
Samples
EMIcp=3.13e-01m
Wlcp =2.26e-01m
Horn =3.13e-01m
Robust =7.65e-04m
Kalman =1.67e-03m

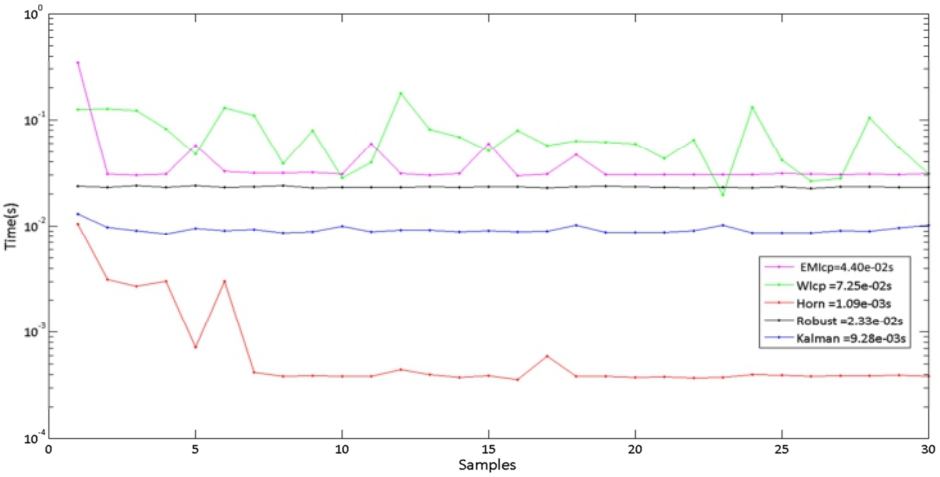
(b)


Zerror(m)
Samples
EMIcp=3.35e-01m
Wlcp =2.46e-01m
Horn =3.30e-01m
Robust =6.66e-04m
Kalman =1.43e-03m

(c)



(d)



(e)

Figure 6.6 Synthetic data with *average* noise magnitude. **(a)** $\boldsymbol{x}$ RMSE. **(b)** $\boldsymbol{y}$ RMSE. **(c)** $\boldsymbol{z}$ RMSE. **(d)** $\boldsymbol{xyz}$ RMSE. **(e)** Processing time

significantly fluctuating from one sample to another. For Horn (red) the processing takes on average 1.09ms (917 FPS). Finally, both proposed solutions still preserve their stable processing time of (23.3, 9.28)ms or (43, 107) FPS for the Robust H_∞ and Kalman respectively.

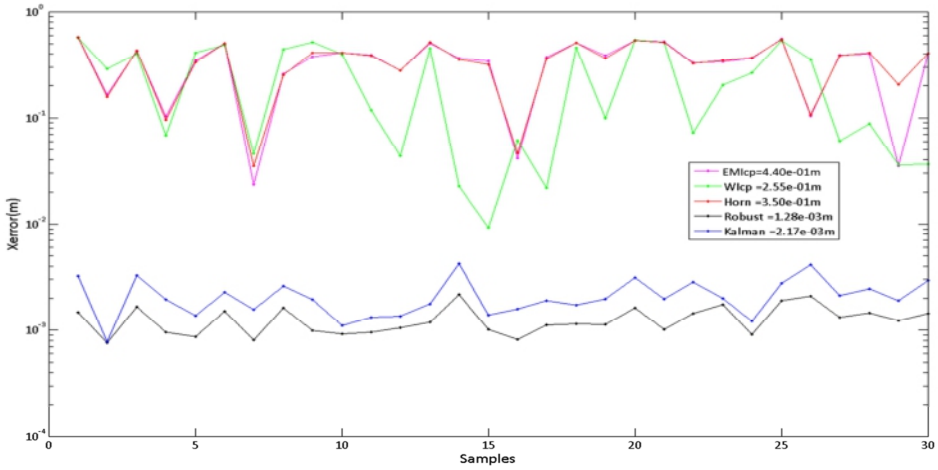
6.10.1.3 Scenario 3: Large Noise Magnitude

RMSE

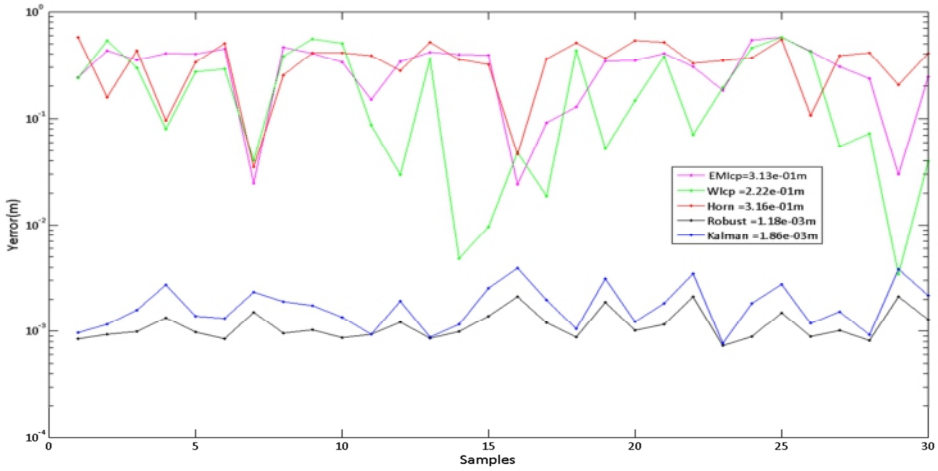
The average RMSE for EMICP (pink in Figure 6.7 (a, b, c, d)) was $(440, 313, 322)$mm for the triplet $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. Its RMSE increased again by 7mm from 319mm to 327mm. WICP RMSE (green in Figure 6.7 (a, b, c, d)) was $(255, 222, 273)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. Its total RMSE increased from 219mm in the scenario of average noise magnitude to 250mm. WICP is still significantly affected by the important amount of noise as the difference in RMSE reached 30mm. Horn RMSE (red in Figure 6.7 (a, b, c, d)) was $(350, 316, 326)$mm for the triplet $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. The overall error increased from 309mm to 331mm. Robust H_∞ RMSE (black in Figure 6.7 (a, b, c, d)) was $(1.28, 1.18, 1.16)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. Whereas, the overall RMSE increased from 0.73mm to 1.21mm. Kalman RMSE (blue in Figure 6.7 (a, b, c, d)) was $(2.17, 1.86, 1.98)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, respectively. The total error increased again by about 0.4mm to reach 2.00mm in this scenario.

Processing time

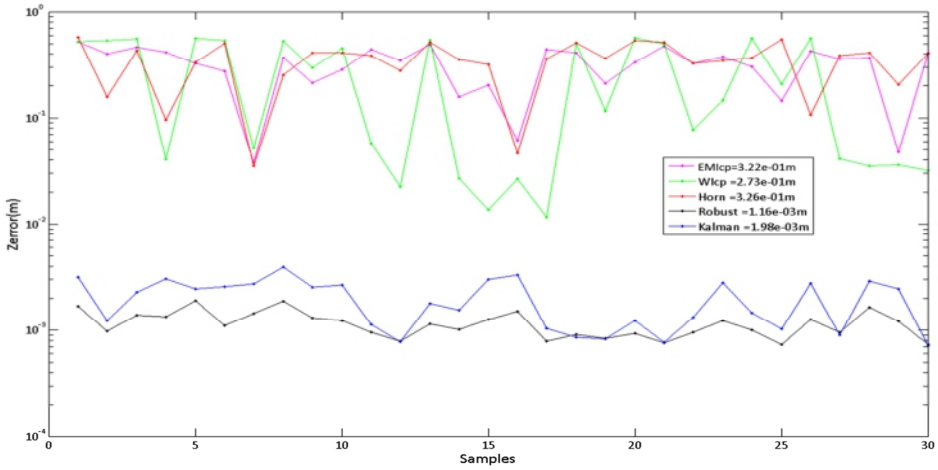
As shown in Figure 6.7 (e), the average processing time taken by EMICP (pink line) was 41ms (24 FPS). This frequency is steady for most samples. With WICP (green line) the processing takes an average period of 71.4ms (14 FPS). The latter is still continually fluctuating. For Horn (red line), the processing takes an average of 1.07ms (934 FPS). The contributed solutions have a very stable processing time of $(23.1, 9.28)$ms or $(43, 107)$ FPS for Robust H_∞ and Kalman respectively.


EMlcp=4.40e-01m
Wlcp =2.55e-01m
Horn =3.50e-01m
Robust =1.28e-03m
Kalman =2.17e-03m
Xerror(m)
Samples

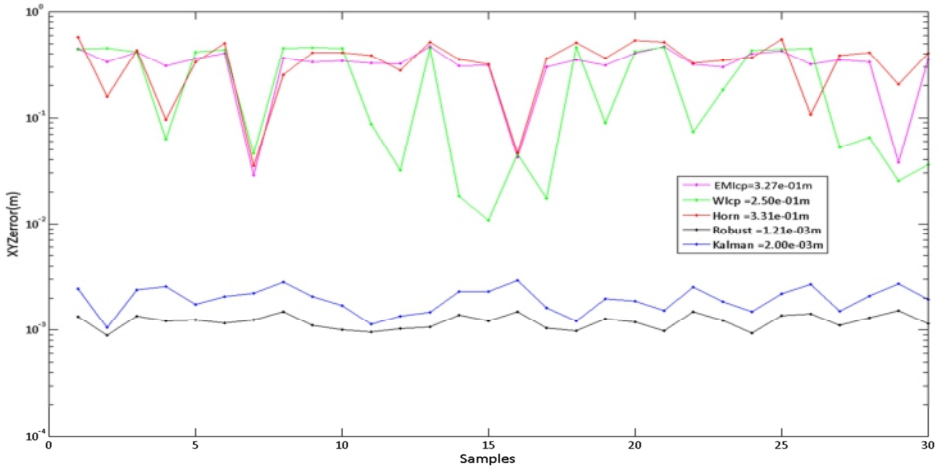
(a)


EMlcp=3.13e-01m
Wlcp =2.22e-01m
Horn =3.16e-01m
Robust =1.18e-03m
Kalman =1.86e-03m
Yerror(m)
Samples

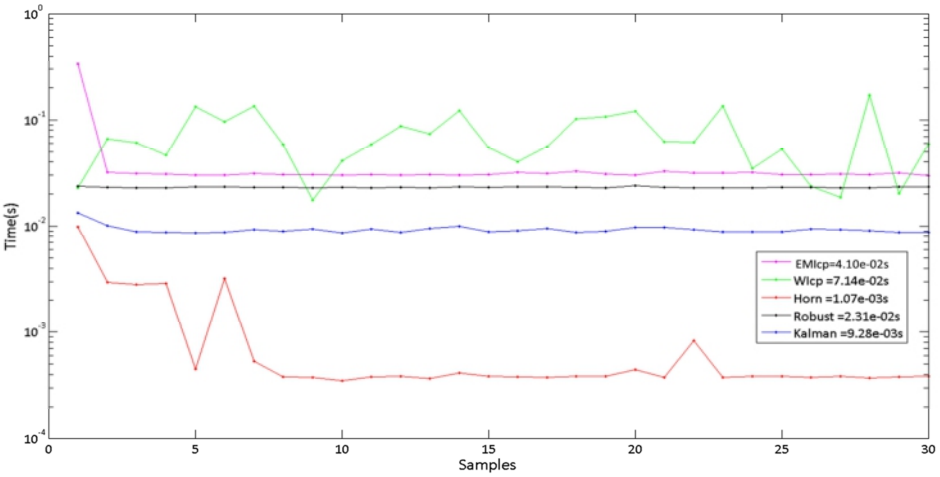
(b)


EMlcp=3.22e-01m
Wlcp =2.73e-01m
Horn =3.26e-01m
Robust =1.16e-03m
Kalman =1.98e-03m
Zerror(m)
Samples

(c)

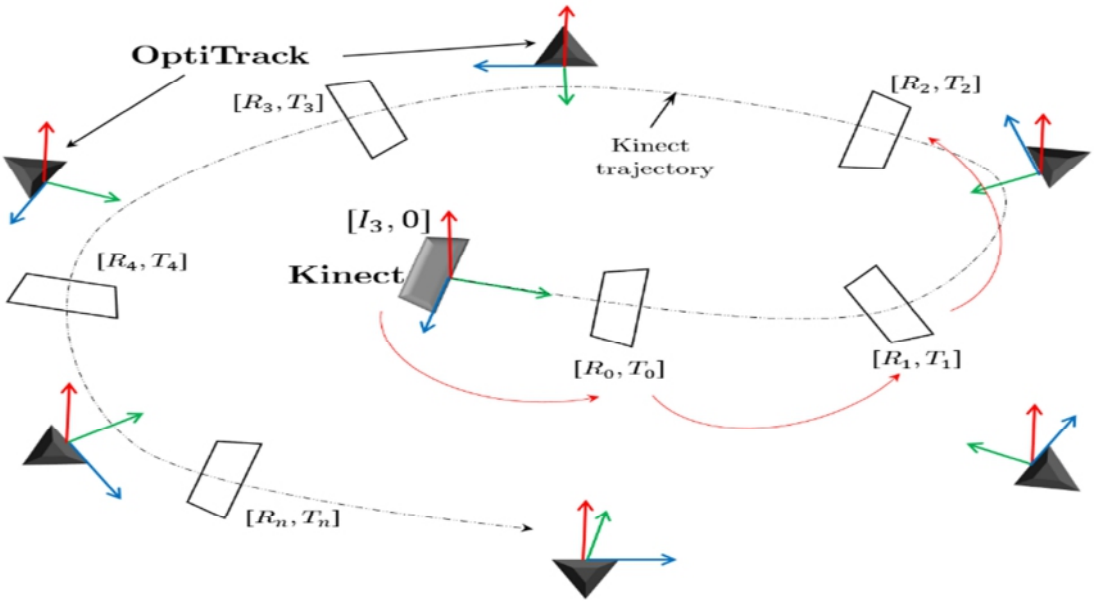


(d)

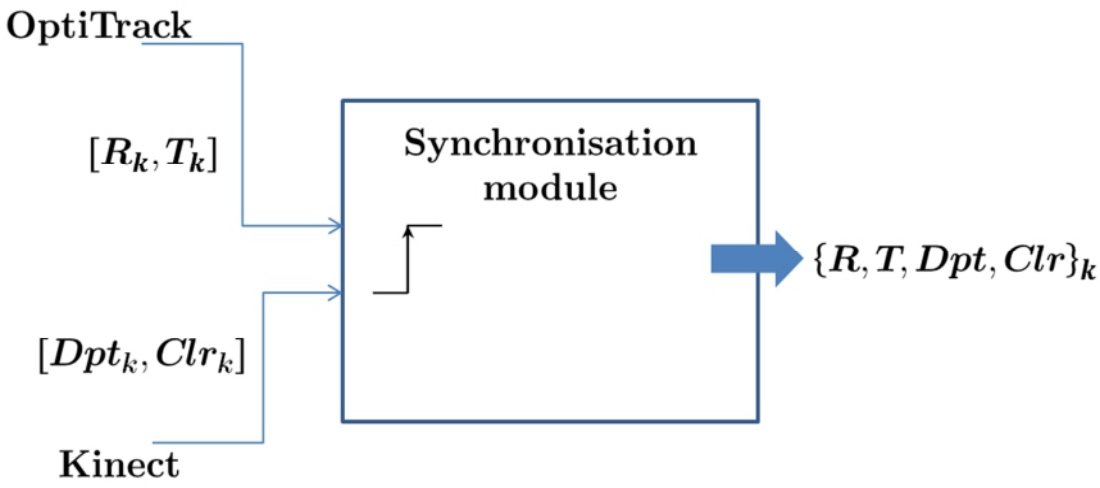


(e)

Figure 6.7 Synthetic data with *large* noise magnitude. **(a)** $\boldsymbol{x}$ RMSE. **(b)** $\boldsymbol{y}$ RMSE. **(c)** $\boldsymbol{z}$ RMSE. **(d)** $\boldsymbol{xyz}$ RMSE. **(e)** Processing time



(a)



(b)

Figure 6.8 Kinect data collection. **(a)** Experimental setup. **(b)** Synchronisation module

6.10.2 Real Data

The proposed solutions have been also validated in a real scenario where image data was delivered by two versions of a consumer RGBD sensor (Microsoft Kinect[1]). The oldest version of the sensor has a VGA resolution of 640×480 pixels for both depth and colour cameras working at 30 FPS. The latter uses a structured light principle [96] to compute the distance separating objects from the camera plane. Whereas, the newest Kinect sensor can stream HD images (1280×1024 pixels) at 30 FPS. 3D point clouds delivered by the old version of the sensor are noisier than the ones streamed by the latest one (Chapter 2).

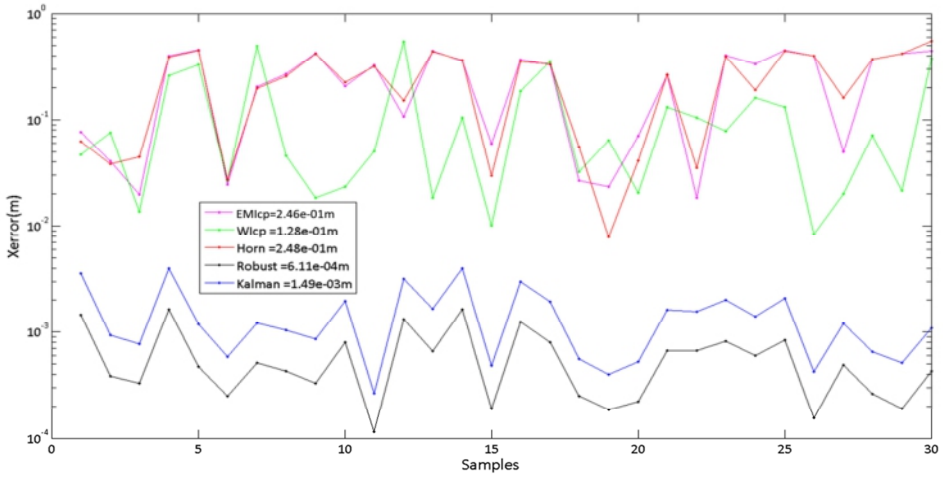
To collect test data, the camera was moved in an indoor environment (Kinect works only indoors). The module captures pairs of colour and depth images. The latter constitute the raw data for 3D point clouds building. In order to precisely assess the performance of registration, a high-quality tracking system (OptiTrack[2]) is required to track instantly the pose of the camera (ground truth), Figure 6.8 (a). Nevertheless, another synchronisation module is needed to associate the actual pose of the sensor $[\boldsymbol{R_k}, \boldsymbol{T_k}]$ with its corresponding pair of depth/colour images $[\boldsymbol{Dpt_k}, \boldsymbol{Clr_k}]$ respectively at time-step $\boldsymbol{k}$, Figure 6.8 (b). Just 400 key points were selected from each acquired point cloud to test the five algorithms.

6.10.2.1 Scenario 4: New Kinect

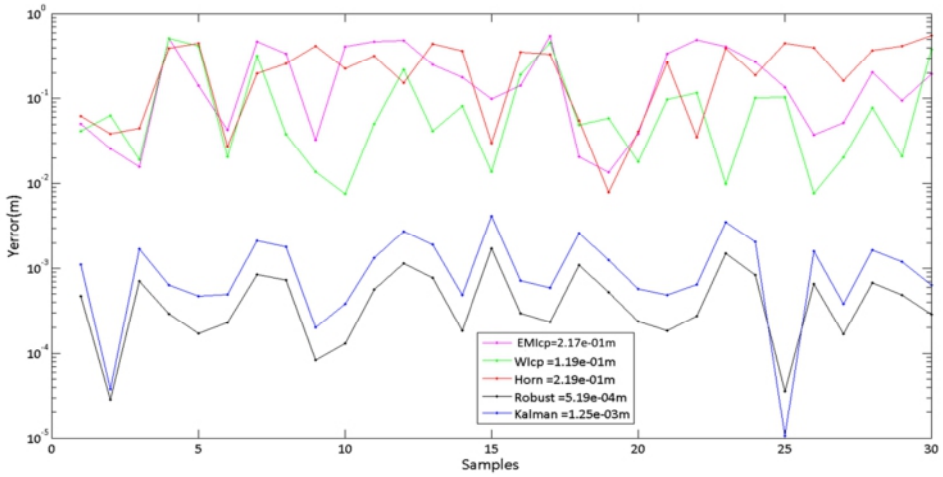
RMSE

The average RMSE observed with EMICP (pink in Figure 6.9 (a, b, c, d)) was $(246, 217, 285)$mm for the triplet $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. The overall error was 285mm. WICP

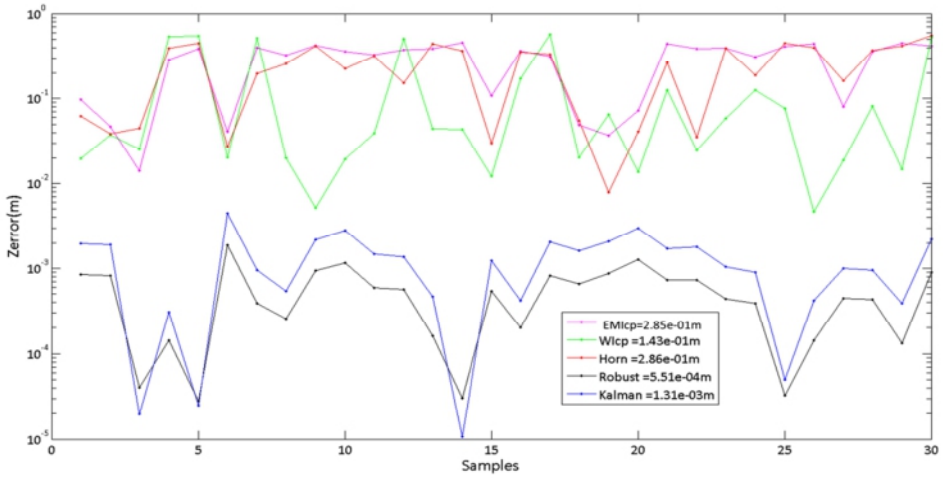
[1] http://www.microsoft.com/en-us/kinectforwindows/. 2015
[2] http://www.naturalpoint.com/optitrack/. 2015


Xerror(m)
Samples
EMIcp=2.46e-01m
Wlcp =1.28e-01m
Horn =2.48e-01m
Robust =6.11e-04m
Kalman =1.49e-03m

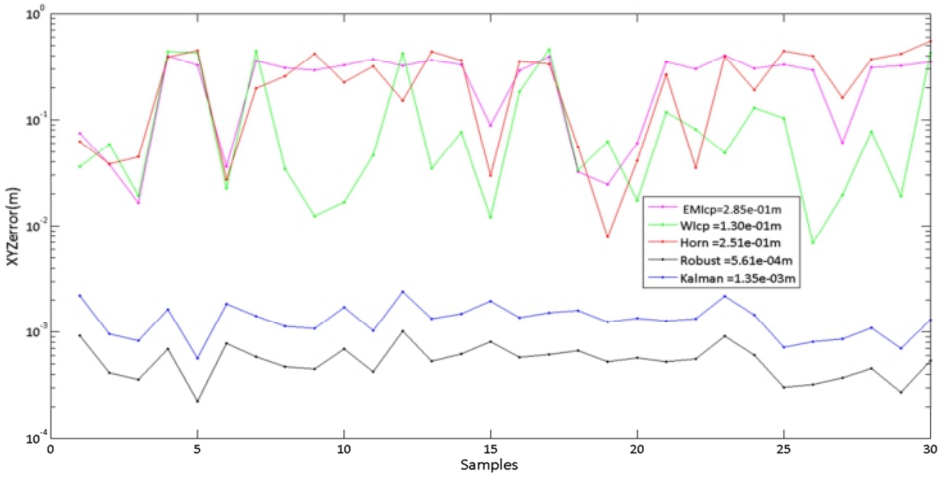
(a)


Yerror(m)
Samples
EMIcp=2.17e-01m
Wlcp =1.19e-01m
Horn =2.19e-01m
Robust =5.19e-04m
Kalman =1.25e-03m

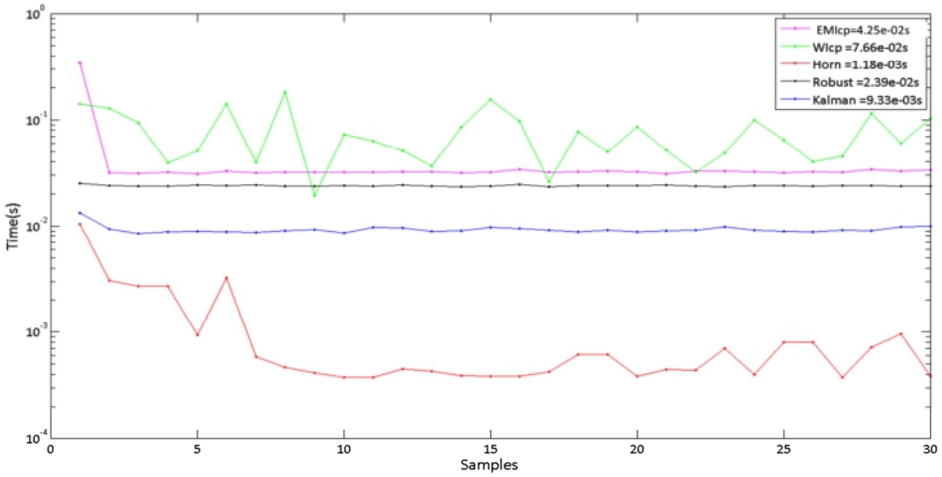
(b)


Zerror(m)
Samples
EMIcp=2.85e-01m
Wlcp =1.43e-01m
Horn =2.86e-01m
Robust =5.51e-04m
Kalman =1.31e-03m

(c)



(d)



(e)

Figure 6.9 New Kinect data. **(a)** x RMSE. **(b)** y RMSE. **(c)** z RMSE. **(d)** xyz RMSE. **(e)** Processing time

RMSE (green in Figure 6.9 (a, b, c, d)) was $(128, 119, 143)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ respectively. The overall error reached 130mm. Horn RMSE (red in Figure 6.9 (a, b, c, d)) was $(248, 219, 286)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. Whereas, the overall error was 251mm. Robust RMSE (black in Figure 6.9 (a, b, c, d)) was $(0.611, 0.51, 0.55)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ respectively. The global error was 0.56mm. Kalman RMSE (blue in Figure 6.9 (a, b, c, d)) was $(1.49, 1.25, 1.31)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ respectively. The total RMSE reached 1.35mm.

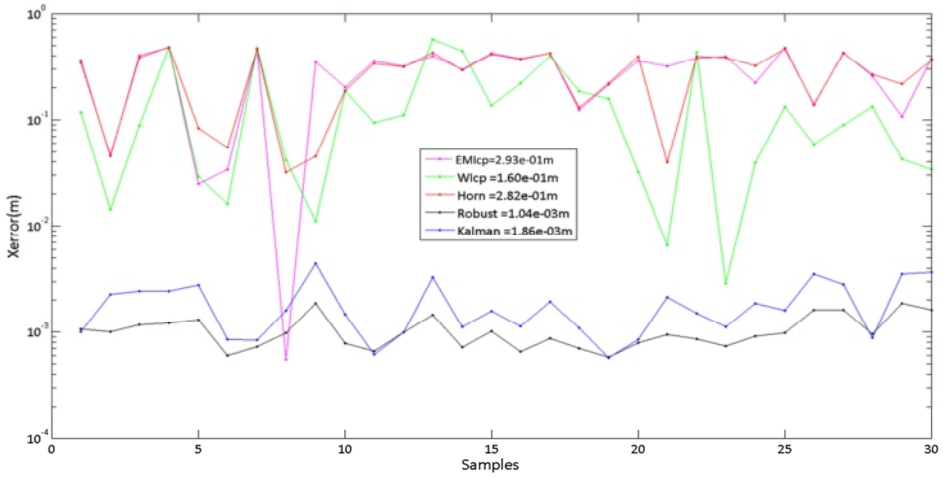
Processing time

As shown in Figure 6.9 (e), the average processing time taken by EMICP (pink line) is 42.5ms (23 FPS). This time is steady for most samples. With WICP (green line) the processing takes an average period of 76.6ms (13 FPS). The latter is still continually fluctuating just like it does with synthetic data. For Horn (red line) the processing takes an average of 1.18ms (847 FPS). Both solutions have a very stable processing time of (23.9, 9.33)ms or (41, 107) FPS for Robust H_∞ and Kalman respectively.

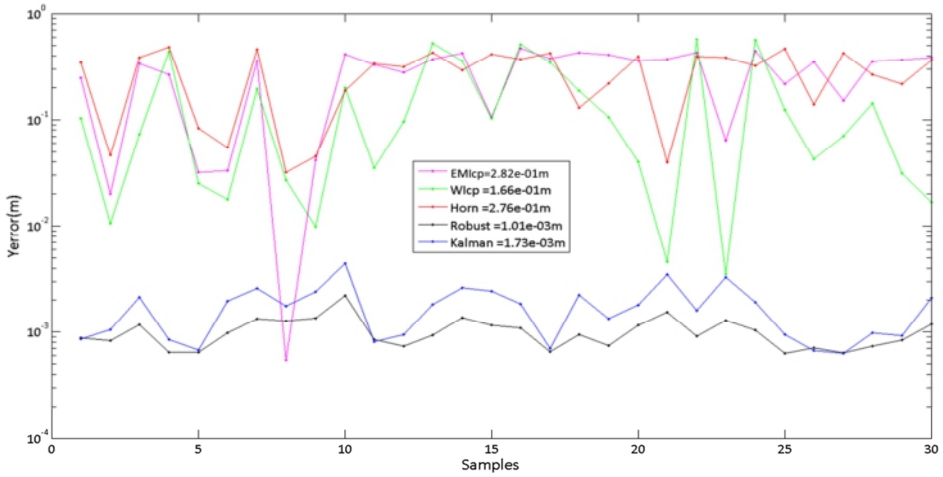
6.10.2.2 Scenario 5: Old Kinect

RMSE

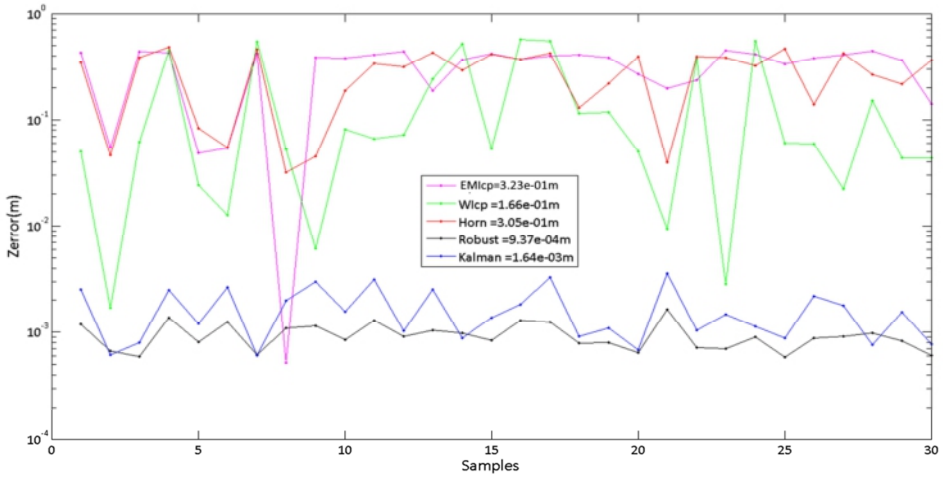
The average RMSE for EMICP (pink in Figure 6.10 ((a, b, c, d)) was $(293, 282, 323)$mm for the triplet $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. The overall error increased from 285mm to 299mm. WICP RMSE (green in Figure 6.10 (a, b, c, d)) was $(160, 166, 166)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ respectively. The total error also increased with about 30mm to reach 164mm. Horn RMSE (red in Figure 6.10 (a, b, c, d)) was $(282, 276, 305)$mm for the triplet $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. The overall error increased by 30mm to reach 288mm. Robust RMSE (black in Figure 6.10 (a, b, c, d)) was $(1.04, 1.01, 0.93)$mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ respectively. The RMSE increased from 0.56mm to 0.99mm. Kalman RMSE (blue in Figure 6.10 (a, b, c, d)) was (1.86, 1.73, 1.64)mm for $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. The overall error increased from 1.35mm to 1.74mm.


EMIcp=2.93e-01m
Wlcp =1.60e-01m
Horn =2.82e-01m
Robust =1.04e-03m
Kalman =1.86e-03m
Xerror(m)
Samples

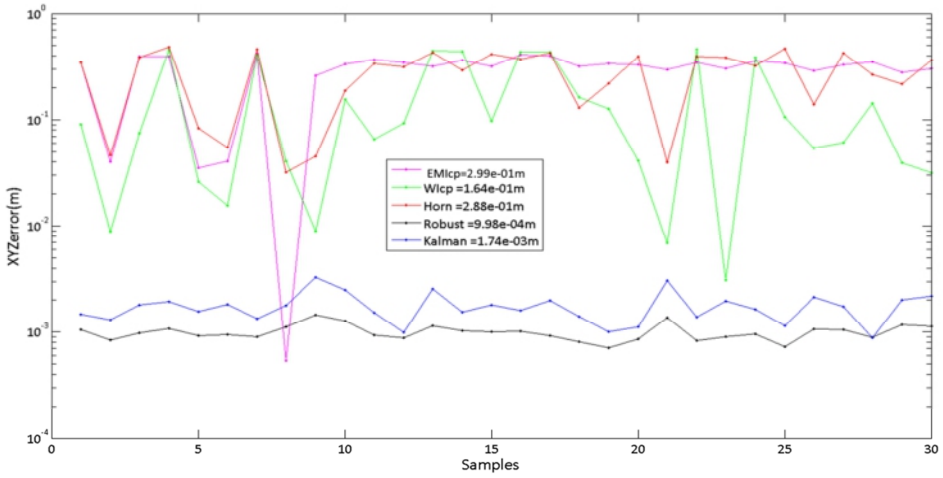
(a)


EMIcp=2.82e-01m
Wlcp =1.66e-01m
Horn =2.76e-01m
Robust =1.01e-03m
Kalman =1.73e-03m
Yerror(m)
Samples

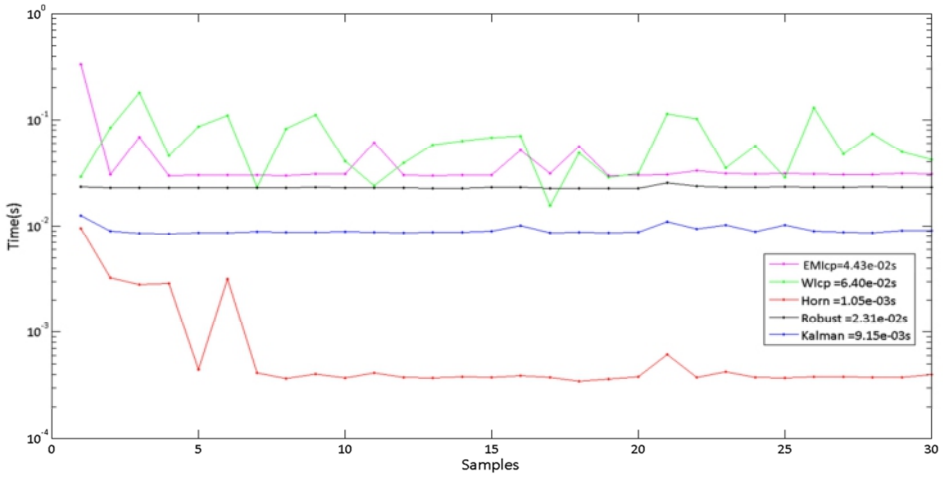
(b)


EMIcp=3.23e-01m
Wlcp =1.66e-01m
Horn =3.05e-01m
Robust =9.37e-04m
Kalman =1.64e-03m
Zerror(m)
Samples

(c)



(d)



(e)

Figure 6.10 Old Kinect data; large noise magnitude. **(a)** $\boldsymbol{x}$ RMSE. **(b)** $\boldsymbol{y}$ RMSE. **(c)** $\boldsymbol{z}$ RMSE. **(d)** $\boldsymbol{xyz}$ RMSE. **(e)** Processing time

Processing time

As shown in Figure 6.10 (e), the average processing time taken by EMICP (pink line) was 44.3ms (22 FPS). This frame rate is steady for most samples. With WICP (green line), the processing takes an average time of 64ms (15 FPS). The latter is still continually fluctuating. For Horn (red line) the processing takes an average of 1.05ms (952 FPS). Both solutions conserve a stable processing time of $(23.1, 9.15)$ms or $(43, 109)$FPS for Robust H_∞ and Kalman respectively.

Table 6.1 shows the overall results for 1000 simulation samples (100 distinct shapes, with ten different poses each) and 120 different point clouds captured by each of the two Kinects in an indoor space.

Noise	**EMICP**	**WICP**	**Horn**	**Robust**	**Kalman**
Small	298	193	285	0.55	1.52
Average	323	235	315	0.91	1.78
Large	332	260	343	0.96	2.10
New Kinect	274	152	246	0.72	1.62
Old Kinect	310	162	302	1.03	2.03

Table 6.1 RMSE (mm) for the whole set of samples: 1000 for each simulation scenario and 120 for every version of the Kinect

Figure 6.11 illustrates some visual results that were encountered during the experiments. As can be seen in Table 6.1, EMICP and Horn have the largest error of alignment, because they do not consider weighting the points involved in the alignment. This behaviour often occurs with these two algorithms when the shapes present some symmetry. WICP has the ability to cope with such drawbacks since it uses knowledge about the quality of features. The latter helps the algorithm to discard noisy elements. On the other hand, the recursive solutions are capable of precisely aligning the shapes using just a small set of features. The percentage of the last mentioned is about 1% of the size of the entire point cloud.

6.11 Conclusion

In this chapter, a novel approach for robust 3D point cloud registration has been presented. This contribution is based on a recursive optimal state estimation framework of Kalman and Robust H_∞. First, the link between the weighted recursive least-squares formulation and its original counterpart (it has not got neither recursion nor weighting) was demonstrated. 3D point cloud registration problem was then fitted into Kalman filter equations.

Since the parameters of the filter (state and projection matrices) were built from noisy data, a non-negligible instability of estimation was noticed. The last mentioned was therefore interpreted as an uncertainty metric that the solution aims to bound and reduce with the Robust H_∞ filter.

The performance of the proposed solution was tested on many synthetic samples as well as real 3D data delivered by each version of the Microsoft Kinect. The results are promising since it was possible to reach a higher performance and robustness with moderate computational power. In real world situations, sensory outputs are always affected by some noise. Hence, Kalman registration is sufficient for the applications that require higher frame rates (>109 FPS) as long as the data is not significantly noisy. However, when the necessary frame rate is below 40 FPS (which is sufficient for most applications), then it would be preferable to benefit totally from the Robust H_∞ estimator, which yields an even better registration accuracy.

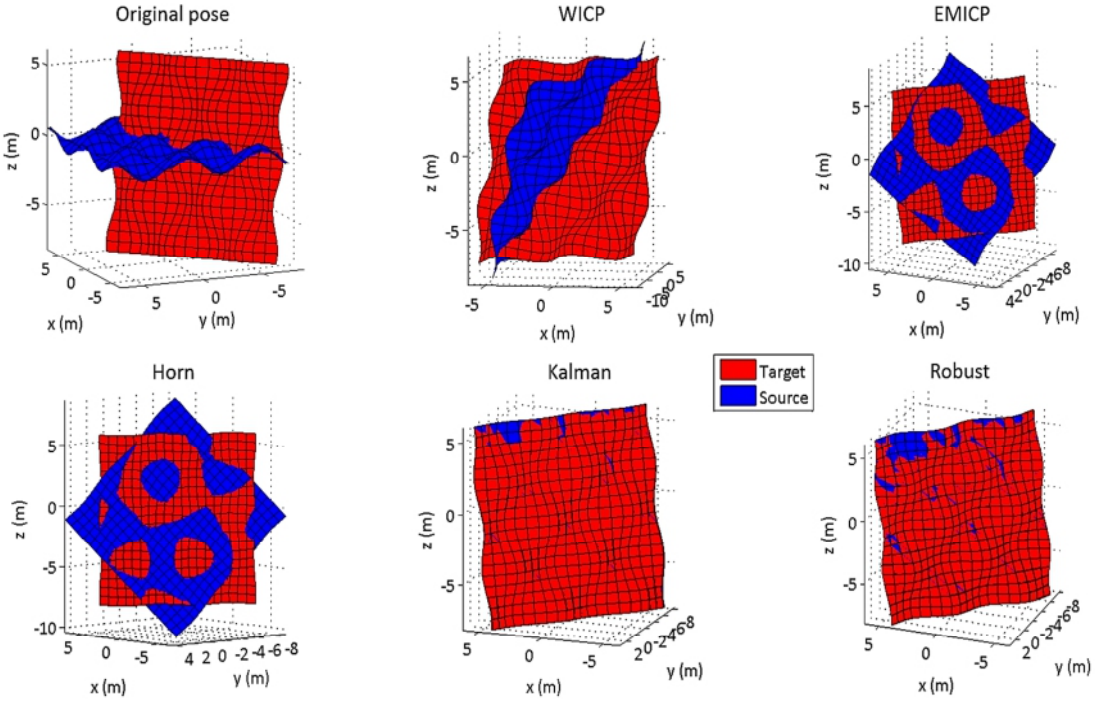
On the other hand, the proposed solution requires some feature points to be extracted from source and target point clouds before the alignment is carried out. The number of the latter is relatively small compared to the entire cloud.

Although the selected subset of key points is theoretically representative and sufficient to obtain a decent registration, a brute force refinement may be still required for the best possible result. For instance, this can be done by taking as correspondent for every single point in the source its closest neighbour in the

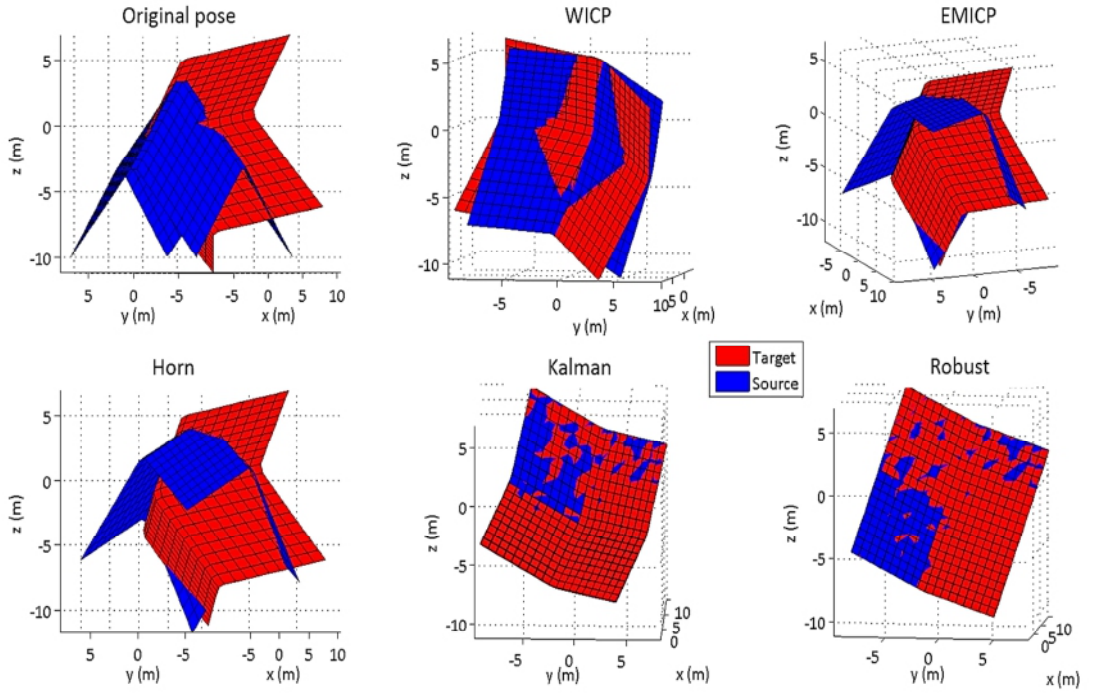
target. The list of correspondences is then delivered to the same algorithms (Kalman, Robust) to run the alignment. This process can be repeatedly performed until the desired result is achieved.

The contributed solution is extensible to any dimension for point clouds, meshes as well as surfaces given that some distinctive features are available. In the case where features are missing, a brute force strategy is still possible.

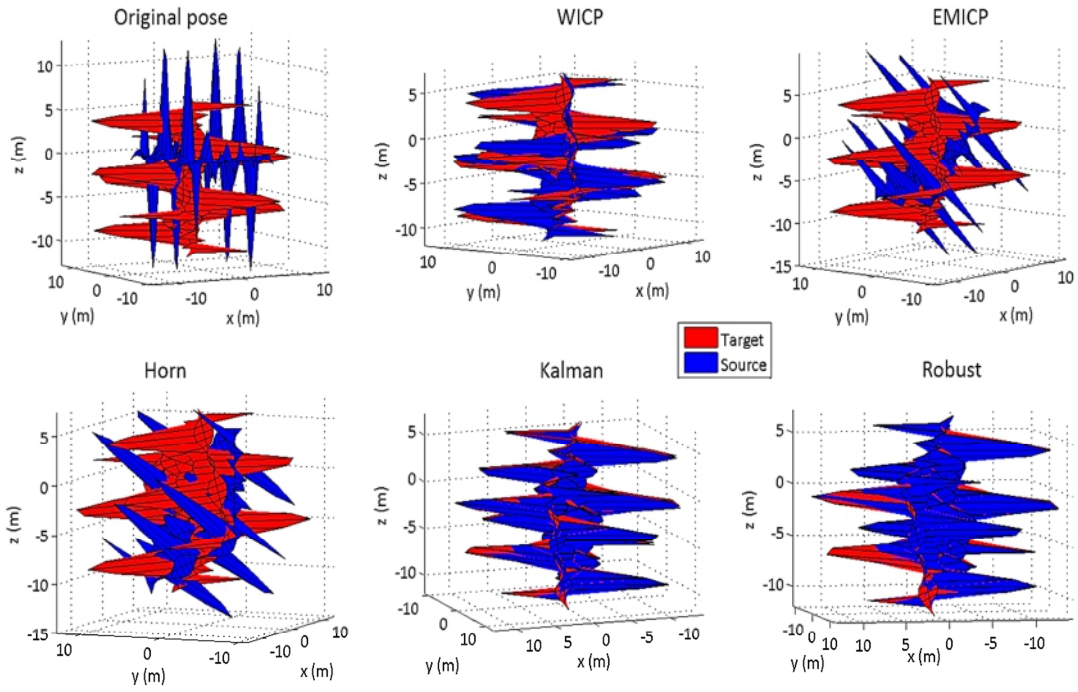
The investigation of other potential applications should also be carried out for the recursive filtering algorithms in the field of computer vision. It would also be interesting to implement the robust filter in the graphic processor to reach higher frame rates. In addition, in a multiview scenario (many sensors streaming images in parallel), fusion algorithms open a new perspective for the users to reconstruct 3D scenes and to track moving objects cooperatively. This new horizon would be useful for the emerging technologies of virtual and augmented reality.


Original pose
WICP
EMICP
Horn
Kalman
Robust
Target
Source
x (m)
y (m)
z (m)

(a)


Original pose
WICP
EMICP
Horn
Kalman
Robust
Target
Source
x (m)
y (m)
z (m)

(b)



(c)

Figure 6.11 Some visual results of the tested algorithms

7 Conclusion & Future Works

7.1 Conclusion

This thesis has investigated the enhancement of the capabilities of cheap depth sensors to develop robust and accurate real-time tracking and registration solutions for Mixed Reality applications. The most significant contributions were therefore devoted to overcoming the different challenges related to the capture of the geometrical and photometrical cues from the real world. Several innovative strategies have been contributed in the domains of RGBD sensor correction/calibration, RGBD image segmentation, multiview real-time 6 DOF tracking and 3D registration. In addition, the achieved progress has led to several other works that can constitute a starting point for future research.

Chapter 3: GPU-Based Real-Time RGBD Data Filtering

A novel mathematical model for the discrete depth map was initially constructed and adapted to a Kalman filtering scheme.

This solution was tested on object tracking applications to assess its accuracy of localising moving targets in real-time. The results have, therefore, proven its rapid smoothing capability. The operational range of the camera has also been extended as a consequence of the filtering. Additionally, the solution was tested on 3D scanning applications in order to assess its ability of surface reconstruction. The outcome of the filtered data gives smoothness and accuracy, whereas, the one based on raw data still suffers from discontinuities and bumpiness.

An architecture that aims at integrating the filter into the existing driver of the depth sensors was also proposed. However, a GPU implementation was required to maintain the real-time operability of the filter. As a result, consumer cameras have become sufficiently endowed to achieve what could not be otherwise done without expensive, sophisticated laser scanners.

Chapter 4: RGBD Data Correction and Background Removal

A novel phenomenon of decay in accuracy was discovered in ageing RGBD sensors. The classical calibration techniques have shown several weaknesses in handling all aspects of sensing accuracy. The proposed correction, on the other hand, uses knowledge about the structure of the depth map, supplied by the technique presented in the previous chapter, and the GPR correction to adjust the precision of the depth readings.

Another innovative real-time background removal approach based on cooperative RGBD data fusion was also presented. The latter has been validated with several tests in real scenarios with different conditions. The results were promising, albeit only the background subtraction was applied without any additional filtering.

Chapter 5: Real-Time Multiview Data Fusion for Object Tracking with RGBD Sensors

Another novel approach to moving vehicle tracking in indoor environments with a setup of multiple RGBD cameras was contributed. All the details about the methodology and the different constraints of implementation were provided. A solution based on the Robust H_∞ filtering scheme was first designed to deliver accurate sensor-wise estimates. Then, another fusion framework built upon a Covariance Intersection algorithm was used to combine optimally the single estimates into a unique, consistent result.

The GPU implementations of computationally greedy processing stages helped significantly to accomplish real-time performance. The experiments have shown a high accuracy of tracking at a frame rate of 25 FPS with a configuration of five Kinects.

Chapter 6: A Recursive Robust Filtering Approach for 3D Registration

The last contribution of this thesis was based on a novel adaptation of recursive optimal state estimation filters to solve the registration problem. To this end, the original alignment error function was reformulated into a recursive least squares framework. Then, a Kalman filter was used to minimise it in the L_2 norm sense. However, since the parameters of the filter were built upon noisy sensory data, they were liable to error and perturbations. A Robust H_∞ framework, that minimises the L_∞ norm regarding error function, was additionally adapted to handle the parametric uncertainties.

The proposed solution was tested on many synthetic as well as real 3D point clouds. The alignment based on a Kalman filter takes less time and is sufficient when the data is not significantly noisy. However, it would be preferable to benefit totally from the Robust H_∞ for better accuracy and robustness.

7.2 Future Works

Although the objectives set in the introductory chapter were mostly achieved, there are still additional improvements that could be the subject of future research.

For Chapter 3, it would be preferable to embed the filtering algorithm in dedicated computing boards in order to free the system entirely from the burden of filtering 3D data. In addition, other imaging problems could be tackled by fitting them in the framework of optimal state estimators. The latter can guarantee the optimality when correctly applied to the problem.

For Chapter 4, a possible future work could be the investigation of the same calibration approach to correct mutually multiple depth cameras. This allows the sensors with low accuracy to compensate with the outputs of the most reliable cameras.

Furthermore, the result of fusing depth and colour images for background segmentation may also inspire other solutions to leverage several image modalities in order to overcome the weaknesses of a single one.

For Chapter 5, it would be interesting to investigate the capabilities of the same framework for multiview object recognition purposes. This would be beneficial since it would allow a complete acquisition of the geometry, texture as well as the behaviour of the targets. In addition, the proposed marker detection strategy could be replaced with a GPU-accelerated markerless approach as long as a sufficient number of good feature points could be obtained. Such an alternative would be more ergonomic because it avoids the utilisation of artificial markers.

For Chapter 6, it is worth considering investigation of the robust filter in other problems where uncertainty cannot be neglected. Moreover, in a multiview setup, it is recommended to combine the elementary sensor-wise pose estimates

with fusion algorithms in order to overcome occlusions and the contrast in accuracy across different views.

Finally, the contributions delivered by this thesis would serve as a solid basis for the design and development of a complete MR solution. The hardware needed is just some low-cost RGBD cameras and GPUs. The proposed tracking as well as registration solutions, software means, would afterwards exploit optimally the potential of rudimentary hardware for a more realistic immersion/interaction.

Bibliography

[1] F. P. Brooks, "What's real about virtual reality?," *IEEE Computer Graphics and Applications*, vol. 19, no. 6, pp. 16–27, 1999.

[2] W. Boehler, M. B. Vicent, and A. Marbs, "Investigating laser scanner accuracy," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, no. Part 5, pp. 696–701, May 2003.

[3] D. Lichti, S. Gordon, and M. Stewart, "Ground-based laser scanners: operation, systems and applications," *Geomatica*, vol. 56, no. 1, pp. 21–33, 2002.

[4] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun, "Performance capture from sparse multi-view video," *ACM Transactions on Graphics*, vol. 27, no. 3, p. 1, Aug. 2008.

[5] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06)*, vol. 1, pp. 519–528.

[6] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan, "Scanning 3D full human bodies using Kinects.," *IEEE transactions on visualization and computer graphics*, vol. 18, no. 4, pp. 643–50, Apr. 2012.

[7] A. Amamra and N. Aouf, "Robust and Sparse RGBD Data Registration of Scene Views," in *2013 17th International Conference on Information Visualisation*, 2013, pp. 488–493.

[8] K. Litomisky, "Consumer RGB-D Cameras and their Applications," 2012.

[9] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim, "Shake'n'sense: reducing interference for overlapping structured light depth cameras," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1933–1936, 2012.

[10] A. Amamra and N. Aouf, "GPU-based real-time RGBD data filtering," *Journal of Real-Time Image Processing*, Sep. 2014.

[11] J. Quarles, S. Lampotang, I. Fischler, P. Fishwick, and B. Lok, "A Mixed Reality Approach for Merging Abstract and Concrete Knowledge," *2008 IEEE Virtual Reality Conference*, no. 1, pp. 27–34, 2008.

[12] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," *IEICE TRANSACTIONS on Information and Systems 77.12 (1994): 1321-1329.*, 1994.

[13] M. Mallem, "Augmented Reality: Issues, trends and challenges," in *2010 2nd International Conference on Image Processing Theory, Tools and Applications*, 2010, pp. 8–8.

[14] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE Computer Graphics and Applications*, vol. 21, no. 6, pp. 34–47, 2001.

[15] *Digital Human Modeling: Trends in Human Algorithms*, vol. 24. Springer Science & Business Media, 2008.

[16] Wen Qi and W. Qi, "A Vision-Based Augmented Reality System for Visualization Interaction," in *Ninth International Conference on Information Visualisation (IV'05)*, 2005, pp. 404–409.

[17] V. C. Brenner, "The Geoscope-A Mixed-reality system for planning and public participation," *25th Urban data management symposium. 2006.*, 2006.

[18] M. Bajura, H. Fuchs, and R. Ohbuchi, "Merging virtual objects with the real world," *ACM SIGGRAPH Computer Graphics*, vol. 26, no. 2, pp. 203–210, Jul. 1992.

[19] M. Livingston, L. Rosenblum, and S. Julier, "An augmented reality system for military operations in urban terrain," 2002.

[20] N. Suzuki, A. Hattori, and M. Hashizume, "Benefits of augmented reality function for laparoscopic and endoscopic surgical robot systems," *navigation*, 2008.

[21] S. Tachi, "Experimental Study on Remote Manipulation Using Virtual Reality," *Proceedings of the Eighth international symposium on measurement and control in robotics, Czech Technical University in Prague, Czech Republic. 1998.*, 1998.

[22] A. Webster, S. Feiner, and B. MacIntyre, "Augmented reality in architectural construction, inspection and renovation," *Proc. ASCE Third Congress on Computing in Civil Engineering. 1996.*, 1996.

[23] G. Reinhart and C. Patron, "Integrating augmented reality in the assembly domain-fundamentals, benefits and applications," *CIRP Annals-Manufacturing Technology*, 2003.

[24] F. Fritz, A. Susperregui, and M. Linaza, "Enhancing cultural tourism experiences with augmented reality technologies," 2005.

[25] D. Bhatnagar, "Position trackers for Head Mounted Display systems: A survey," *University of North Carolina, Chapel Hill TR93*, 1993.

[26] D. Gonzales, D. R. Criswell, E. Heer, U. S. A. Force, and U. S. N. A. and S. Administration, *Automation and robotics for the Space Exploration Initiative: results from Project Outreach*. Rand, 1991.

[27] M. L. Heilig, "El cine del futuro: the cinema of the future," *Presence: Teleoperators and Virtual Environments*, vol. 1, no. 3, pp. 279–294, Jul. 1992.

[28] R. Klette, *Concise Computer Vision*. London: Springer London, 2014.

[29] T. C. Randy Pausch, "A Literature Survey for Virtual Environments: Military Flight Simulator Visual Systems and Simulator Sickness.," *Presence*, vol. 1, pp. 344 – 363, 1992.

[30] R. B. Welch, *Perceptual modification: adapting to altered sensory environments*. Academic Press, 1978.

[31] G. K. L. Tam, Z. Cheng, Y. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X. Sun, and P. L. Rosin, "Registration of 3D Point Clouds and Meshes : A Survey From Rigid to Non-Rigid," vol. 19, no. 7, pp. 1–20, 2013.

[32] A. Adams, N. Gelfand, J. Dolson, and M. Levoy, "Gaussian KD-trees for fast high-dimensional filtering," *ACM Transactions on Graphics*, vol. 28, no. 3, p. 1, Jul. 2009.

[33] M. A. Treiber, *Optimization for Computer Vision*. London: Springer London, 2013.

[34] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, p. 629, Apr. 1987.

[35] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society of America A*, vol. 5, no. 7, p. 1127, Jul. 1988.

[36] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136.

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

[38] W. Narzt, G. Pomberger, A. Ferscha, D. Kolb, R. Müller, J. Wieghardt, H. Hörtner, and C. Lindinger, "Augmented reality navigation systems," *Universal Access in the Information Society*, vol. 4, no. 3, pp. 177–187, Dec. 2005.

[39] M. Gross, S. Lang, K. Strehlke, A. Vande Moere, O. Staadt, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, and L. Van Gool, "blue-c," *ACM Transactions on Graphics*, vol. 22, no. 3, p. 819, Jul. 2003.

[40] A. F. Bill Triggs, Philip Mclauchlan, Richard Hartley, "Bundle adjustment – a modern synthesis," *Vision algorithms: theory and practice. Springer Berlin Heidelberg, 2000. 298-372.*

[41] R. Hartley and P. Sturm, "Triangulation," *Computer vision and image understanding 68.2 (1997): 146-157.*, 1997.

[42] K. Kolev, M. Klodt, T. Brox, S. Esedoglu, and D. Cremers, "Continuous Global Optimization in Multiview 3D Reconstruction," pp. 1–13.

[43] K. Khoshelham, "ACCURACY ANALYSIS OF KINECT DEPTH DATA," 2010.

[44] J. Geng, "Structured-light 3D surface imaging: a tutorial," *Advances in Optics and Photonics*, 2011.

[45] M. Andersen, T. Jensen, and P. Lisouski, "Kinect depth sensor evaluation for computer vision applications," 2012.

[46] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of Kinect depth data for indoor mapping applications.," *Sensors (Basel, Switzerland)*, vol. 12, no. 2, pp. 1437–54, Jan. 2012.

[47] T. Svoboda, D. Martinec, and T. Pajdla, "A Convenient Multicamera Self-Calibration for Virtual Environments," *Presence: Teleoperators and Virtual Environments*, vol. 14. pp. 407–422, 2005.

[48] G. Kurillo and R. Bajcsy, "Wide-area external multi-camera calibration using vision graphs and virtual calibration object," in *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, 2008, pp. 1–9.

[49] A. Maimone and H. Fuchs, "Reducing interference between multiple structured light depth sensors using motion," *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, no. May, pp. 51 – 54, 2012.

[50] C. Tomasi, "Good Features," in *IEEE Computer Society Conference*, 1994, pp. 593–600.

[51] P. Azad, T. Asfour, and R. Dillmann, "Combining Harris interest points and the SIFT descriptor for fast scale-invariant object recognition," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 4275–4280.

[52] K. Xu, L. Qin, and L. Yang, "RGB-D fusion toward accurate 3D mapping," in *2011 IEEE International Conference on Robotics and Biomimetics*, 2011, pp. 2618–2622.

[53] I. Sipiran and B. Bustos, "Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes," *The Visual Computer*, vol. 27, no. 11, pp. 963–976, Jul. 2011.

[54] X. Jiang, O. R. P. Bellon, D. Goldgof, and T. Oishi, Eds., *Advances in Depth Image Analysis and Applications*, vol. 7854. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[55] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3384–3391.

[56] A. Flint, A. Dick, and A. van den Hengel, "Thrift: Local 3D Structure Recognition," in *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007)*, 2007, pp. 182–188.

[57] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, 2004.

[58] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer vision–ECCV 2006*, 2006.

[59] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1–4.

[60] F. Tombari, S. Salti, and L. Di Stefano, "Performance evaluation of 3D keypoint detectors," *International Journal of Computer Vision*, 2013.

[61] L. D. S. Federico Tombari , Samuele Salti, "A Combined Texture-Shape Descriptor For Enhanced 3d Feature Matching," in *Conference, Ieee International Processing, Image*, 2011, pp. 825–828.

[62] F. Tombari, S. Salti, and L. Di Stefano, "A combined texture-shape descriptor for enhanced 3D feature matching," in *2011 18th IEEE International Conference on Image Processing*, 2011, pp. 809–812.

[63] Y. Allusse, P. Horain, A. Agarwal, and C. Saipriyadarshan, "GpuCV: A GPU-accelerated framework for image processing and computer vision," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 5359 LNCS, pp. 430–439.

[64] S. Akhter and J. Roberts, *Multi-core programming*. 2006.

[65] K. Kraus and N. Pfeifer, "Determination of terrain models in wooded areas with airborne laser scanner data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 53, no. 4, pp. 193–203, 1998.

[66] W. Cady, "Radar scanners and radomes," 2008.

[67] Y. Y. M. Kim, C. Theobalt, J. Diebel, J. Kosecka, B. Miscusik, and S. Thrun, "Multi-view image and ToF sensor fusion for dense 3D reconstruction," in *2009*

IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, 2009, pp. 1542–1549.

[68] C. Meinherz, "Time of flight camera unit and optical surveillance system," *US Patent App. 13/086,686*, Oct. 2011.

[69] J. Smisek, M. Jancosek, and T. Pajdla, "3D with Kinect," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 1154–1160.

[70] O. Hall-Holt and S. Rusinkiewicz, "Stripe boundary codes for real-time structured-light range scanning of moving objects," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2001, vol. 2, pp. 359–366.

[71] F. Menna, F. Remondino, R. Battisti, and E. Nocerino, "Geometric investigation of a gaming active device," *Videometrics, Range Imaging, and Applications XI*, p. 80850G–80850G–15, Jun. 2011.

[72] M. Camplani, T. Mantecon, and L. Salgado, "Depth-color fusion strategy for 3-D scene modeling with Kinect.," *IEEE transactions on cybernetics*, vol. 43, no. 6, pp. 1560–71, Dec. 2013.

[73] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, p. 35, 1960.

[74] L. Ling, E. Cheng, and I. S. Burnett, "An Iterated Extended Kalman Filter for 3D mapping via Kinect camera," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 1773–1777.

[75] T. Hervier, S. Bonnabel, and F. Goulette, "Accurate 3D maps from depth images and motion sensors via nonlinear Kalman filtering," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5291–5297.

[76] S. Park, S. Yu, J. Kim, S. Kim, and S. Lee, "3D hand tracking using Kalman filter in depth space," *EURASIP Journal on Advances in Signal Processing*, vol. 2012, no. 1, p. 36, 2012.

[77] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. Wiley-Interscience, 2006.

[78] D. Liebowitz and A. Zisserman, "Metric rectification for perspective images of planes," in *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, 1998, pp. 482–488.

[79] G. Arce, *Nonlinear signal processing: a statistical approach*. 2005.

[80] J. Fung and S. Mann, "Using graphics devices in reverse: GPU-based Image Processing and Computer Vision," in *2008 IEEE International Conference on Multimedia and Expo*, 2008, pp. 9–12.

[81] F. Jargstorff, "GPU Image Processing," *SIGGRAPH 2004*, 2004.

[82] E. Kilgariff and R. Fernando, "The GeForce 6 series GPU architecture," *ACM SIGGRAPH 2005 Courses*, 2005.

[83] P. Carr, "GPU Accelerated Multimodal Background Subtraction.," *DICTA*, 2008.

[84] I. I. Conference and I. Processing, "A Combined Texture-Shape Descriptor For Enhanced 3d Feature Matching," pp. 809–812, 2011.

[85] S. Izadi, A. Davison, A. Fitzgibbon, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, and D. Freeman, "KinectFusion," in *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*, 2011, p. 559.

[86] J. Davis, "Fusion of time-of-flight depth and stereo for high accuracy depth maps," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[87] W. Chiu, U. Blanke, and M. Fritz, "Improving the Kinect by Cross-Modal Stereo.," *BMVC*, 2011.

[88] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, Feb. 2012.

[89] S. Matyunin, D. Vatolin, Y. Berdnikov, and M. Smirnov, "Temporal filtering for depth maps generated by Kinect depth camera," in *2011 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, 2011, pp. 1–4.

[90] M. Cristani, M. Bicego, and V. Murino, "Multi-level background initialization using Hidden Markov Models," in *First ACM SIGMM international workshop on Video surveillance - IWVS '03*, 2003, p. 11.

[91] M. D. Abràmoff, W. L. M. Alward, E. C. Greenlee, L. Shuba, C. Y. Kim, J. H. Fingert, and Y. H. Kwon, "Automated segmentation of the optic disc from stereo color photographs using physiologically plausible features.," *Investigative ophthalmology & visual science*, vol. 48, no. 4, pp. 1665–73, Apr. 2007.

[92] G. Gordon, T. Darrell, M. Harville, and J. Woodfill, "Background estimation and removal based on range and color," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, 1999, pp. 459–464.

[93] F. E. Alsaqre and Y. Baozong, "Moving object segmentation from video sequences: an edge approach," in *Proceedings EC-VIP-MC 2003. 4th EURASIP Conference focused on Video/Image Processing and Multimedia Communications (IEEE Cat. No.03EX667)*, 1997, vol. 1, pp. 193–199.

[94] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, 1999, pp. 246–252.

[95] S. Lee and C. Jeong, "Real-time Object Segmentation based on GPU," in *2006 International Conference on Computational Intelligence and Security*, 2006, pp. 739–742.

[96] J. Fu, S. Wang, Y. Lu, S. Li, and W. Zeng, "Kinect-like depth denoising," in *2012 IEEE International Symposium on Circuits and Systems*, 2012, pp. 512–515.

[97] D. MacKay, "Introduction to Gaussian processes," *NATO ASI Series F Computer and Systems Sciences*, 1998.

[98] S. Seo, M. Wallat, T. Graepel, and K. Obermayer, "Gaussian process regression: Active data selection and test point rejection," *Mustererkennung 2000*, 2000.

[99] D. Reynolds, "Gaussian mixture models," *Encyclopedia of Biometrics*, 2009.

[100] E. J. Fernandez-Sanchez, J. Diaz, and E. Ros, "Background subtraction based on color and depth using active sensors.," *Sensors (Basel, Switzerland)*, vol. 13, no. 7, pp. 8895–915, Jan. 2013.

[101] A. Yilmaz, O. Javed, and M. Shah, "Object tracking," *ACM Computing Surveys*, vol. 38, no. 4, p. 13–es, Dec. 2006.

[102] M. Taj and A. Cavallaro, "Multi-view multi-object detection and tracking," *Computer Vision*, 2010.

[103] R. Nevatia, "Self-calibration of a camera from video of a walking human," in *Object recognition supported by user interaction for service robots*, 2002, vol. 1, pp. 562–567.

[104] K. Okuma, A. Taleghani, and N. De Freitas, "A boosted particle filter: Multitarget detection and tracking," *Computer Vision-ECCV*, 2004.

[105] S. Z. Li, "Multi-pedestrian detection in crowded scenes: A global view," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3124–3129.

[106] A. F. Bobick, S. S. Intille, J. W. Davis, F. Baird, C. S. Pinhanez, L. W. Campbell, Y. A. Ivanov, A. Schütte, and A. Wilson, "The KidsRoom: A

Perceptually-Based Interactive and Immersive Story Environment," *Presence: Teleoperators and Virtual Environments*, vol. 8, no. 4, pp. 369–393, Aug. 1999.

[107] S. Intille, J. Davis, and A. Bobick, "Real-time closed-world tracking," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 697–703.

[108] C. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, Jul. 1997.

[109] J. Flusser and T. Suk, "Rotation Moment Invariants for Recognition of Symmetric Objects," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3784–3790, Dec. 2006.

[110] J.-Q. Tarn, "A state space formalism for anisotropic elasticity. Part I: Rectilinear anisotropy," *International Journal of Solids and Structures*, vol. 39, no. 20, pp. 5143–5155, Oct. 2002.

[111] S. Y. Chen, "Kalman Filter for Robot Vision: A Survey," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409–4420, Nov. 2012.

[112] L. Liu, B. Sun, N. Wei, C. Hu, and M. Q.-H. Meng, "A Novel Marker Tracking Method Based on Extended Kalman Filter for Multi-Camera Optical Tracking Systems," in *2011 5th International Conference on Bioinformatics and Biomedical Engineering*, 2011, pp. 1–5.

[113] Y. Rui and Y. Chen, "Better proposal distributions: object tracking using unscented particle filter," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, vol. 2, pp. II–786–II–793.

[114] M. Pupilli and A. Calway, "Real-Time Camera Tracking Using a Particle Filter.," *BMVC*, 2005.

[115] Wolfgan Niehsen and W. Niehsen, "Information fusion based on fast covariance intersection filtering," in *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002. (IEEE Cat.No.02EX5997)*, 2002, vol. 2, pp. 901–904.

[116] D. Smith and S. Singh, "Approaches to Multisensor Data Fusion in Target Tracking: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 12, pp. 1696–1710, Dec. 2006.

[117] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, p. 116, Jan. 2013.

[118] D. S. O. Correa, D. F. Sciotti, M. G. Prado, D. O. Sales, D. F. Wolf, and F. S. Osorio, "Mobile Robots Navigation in Indoor Environments Using Kinect

Sensor," in *2012 Second Brazilian Conference on Critical Embedded Systems*, 2012, pp. 36–41.

[119] T. Nakamura, "Real-time 3-D object tracking using Kinect sensor," in *2011 IEEE International Conference on Robotics and Biomimetics*, 2011, pp. 784–788.

[120] "kinect_calibration/technical - ROS Wiki." [Online]. Available: http://wiki.ros.org/kinect_calibration/technical. [Accessed: 27-Jan-2014].

[121] M. Sedláček, "Evaluation of RGB and HSV Models in Human Faces Detection. Central European Seminar on Computer Graphics, Budmerice."

[122] Y. S. Hung and F. Yang, "Robust H∞ filtering with error variance constraints for discrete time-varying systems with uncertainty," 2003.

[123] D. Franken and A. Hupper, "Improved fast covariance intersection for distributed data fusion," in *2005 7th International Conference on Information Fusion*, 2005, vol. 1, p. 7 pp.

[124] C. G. M. P. A. P. S. S. Rita Cucchiara, "Improving Shadow Suppression in Moving Object Detection with HSV Color Information."

[125] M. S. Mahmoud, "Resilient linear filtering of uncertain systems," 2004.

[126] L. Xie, L. Lu, D. Zhang, and H. Zhang, "Improved robust H2 and H∞ filtering for uncertain discrete-time systems," 2004.

[127] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey.," *Cognitive processing*, vol. 12, no. 4, pp. 319–40, Nov. 2011.

[128] A. Parr, R. Miesen, F. Kirsch, and M. Vossiek, "A novel method for UHF RFID tag tracking based on acceleration data," in *2012 IEEE International Conference on RFID (RFID)*, 2012, pp. 110–115.

[129] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce, "Non-uniform Deblurring for Shaken Images."

[130] A. Baumberg, "Reliable feature matching across widely separated views," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, 2000, vol. 1, pp. 774–781.

[131] 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*, vol. PAMI-9, no. 5, pp. 698–700, Sep. 1987.

[132] Turgay Celik and Kai-Kuang Ma, "Fast object-based image registration using principal component analysis for super-resolution imaging." pp. 705–710, 2008.

[133] P. J. Besl and H. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[134] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, Mar. 1966.

[135] O. Faugeras and M. Hebert, "The Representation, Recognition, and Locating of 3-D Objects," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 27–52, Sep. 1986.

[136] C. Dorai, J. Weng, and A. K. Jain, "Optimal registration of object views using range data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, pp. 1131–1138, 1997.

[137] Chu-Song Chen, C.-S. Chen, Y.-P. Hung, and J.-B. Cheng, "RANSAC-based DARCES: a new approach to fast automatic registration of partially overlapping range images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 11, pp. 1229–1234, 1999.

[138] M. W. Walker, L. Shao, and R. A. Volz, "Estimating 3-D location parameters using dual number quaternions," *CVGIP: Image Understanding*, vol. 54, no. 3, pp. 358–367, Nov. 1991.

[139] T. Tamaki, M. Abe, B. Raytchev, and K. Kaneda, "Softassign and EM-ICP on GPU," *2010 First International Conference on Networking and Computing*, pp. 179–183, Nov. 2010.

[140] S. Gold, A. Rangarajan, C.-P. Lu, S. Pappu, and E. Mjolsness, "New algorithms for 2D and 3D point matching," *Pattern Recognition*, vol. 31, no. 8, pp. 1019–1031, Aug. 1998.

[141] S. Fantoni, U. Castellani, and A. Fusiello, "Accurate and Automatic Alignment of Range Surfaces," in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, 2012, pp. 73–80.

[142] J. Servos and S. L. Waslander, "Multi channel generalized-ICP," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3644–3649.

[143] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. Hill, M. O. Leach, and D. J. Hawkes, "Nonrigid registration using free-form deformations: application to breast MR images.," *IEEE transactions on medical imaging*, vol. 18, no. 8, pp. 712–21, Aug. 1999.

[144] A. Larusso, D. Eggert, and R. Fisher, "A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations.," in *Procedings of the British Machine Vision Conference 1995*, 1995, pp. 24.1–24.10.

[145] 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, Apr. 1991.

[146] L. Nielsen, "Least-squares estimation using Lagrange multipliers," *Metrologia*, vol. 37, no. 2, pp. 183–183, Apr. 2000.

[147] K. Kanatani, "Analysis of 3-D rotation fitting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 543–549, May 1994.

[148] S. Granger and X. Pennec, *Computer Vision — ECCV 2002*, vol. 2353. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002.

[149] Y. Lu and J.-Q. Fang, *Advanced Medical Statistics*. WORLD SCIENTIFIC, 2003.

[150] X. Pennec and J.-P. Thirion, "A Framework for Uncertainty and Validation of 3-D Registration Methods Based on Points and Frames," *International Journal of Computer Vision*, vol. 25, no. 3, pp. 203–229, Dec. 1997.

[151] B. Ma and R. E. Ellis, *Surface-based registration with a particle filter*, vol. 3216. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[152] E. A. Wan and R. van der Merwe, *Kalman Filtering and Neural Networks*. New York, USA: John Wiley & Sons, Inc., 2001.

[153] N. Ohta and K. Kanatani, *Optimal estimation of three-dimensional rotation and reliability evaluation*, vol. 1406. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998.

[154] R. Balachandran, J. M. Fitzpatrick, and R. F. Labadie, "Fiducial registration for tracking systems that employ coordinate reference frames | (2005) | Balachandran | Publications | Spie," 2005, pp. 134–145.

[155] S. J. Julier and J. K. Uhlmann, "Unscented Filtering and Nonlinear Estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, Mar. 2004.

[156] F. Kahl and R. Hartley, "Multiple-view geometry under the Linfinity-norm.," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 9, pp. 1603–17, Sep. 2008.

[157] B. Micusik and R. Pflugfelder, "Localizing non-overlapping surveillance cameras under the L-Infinity norm," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2895–2901.

[158] H. Lee, Y. Seo, and R. Hartley, "Homography estimation with L∞ norm minimisation method," *14th Korea-Japan Joint Workshop FCV, pp. 87-91. 2008.*, 2008.

[159] B. Bellekens, V. Spruyt, R. Berkvens, and M. Weyn, "A Survey of Rigid 3D Pointcloud Registration Algorithms," no. c, pp. 8–13, 2014.

[160] " Front Matter : Classics in Applied Mathematics: Vol. , No. (Society for Industrial and Applied Mathematics) ."

[161] J. Willems, "Least squares stationary optimal control and the algebraic Riccati equation," *IEEE Transactions on Automatic Control*, vol. 16, no. 6, pp. 621–634, Dec. 1971.

[162] R. J. Carroll and D. Ruppert, *Transformation and Weighting in Regression*. CRC Press, 1988.

[163] T. P. Ryan, *Modern regression methods, Volume 1*. Wiley, 1997.

[164] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, 1981.

[165] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. Series D, p. 35, 1960.

[166] S. Chicotay, O. E. David, and N. S. Netanyahu, "Image Registration of Very Large Images via Genetic Programming," in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 329–334.

[167] J. Fix, "An introduction to Kalman filters," pp. 1–11, 2012.

[168] F. Le Gall, "Powers of tensors and fast matrix multiplication," in *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation - ISSAC '14*, 2014, pp. 296–303.

[169] Y. Kanazawa and K. Kanatani, "Do we really have to consider covariance matrices for image features?," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2001, vol. 2, pp. 301–306.

[170] M. J. Brooks, W. Chojnacki, D. Gawley, and A. van den Hengel, "What value covariance information in estimating vision parameters?," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2001, vol. 1, pp. 302–308.

[171] A. Marinov, N. Zlateva, D. Dimov, and D. Marinov, "Weighted ICP Algorithm for Alignment of Stars from Scanned Astronomical Photographic Plates," *Serdica Journal of Computing*, vol. 6, no. 1. pp. 101–110, 2012.

[172] J. Hermans, D. Smeets, D. Vandermeulen, and P. Suetens, "Robust point set registration using EM-ICP with information-theoretically optimal outlier handling," in *CVPR 2011*, 2011, pp. 2465–2472.

[173] J. Kuipers, *Quaternions and rotation sequences*. 1999.