MONASH University

Localization of RGB-D Sensors for Robotic and AR Applications

Nalika Damayanthi Nahinnage Dona

A thesis submitted for the degree of Master at

Monash University in 2019

Electrical and Computer Systems Engineering - ECSE

Copyright notice

Abstract

Localization, the problem of estimating the position and orientation of robots given the sensor readings and map of the environment is a fundamental challenge in robotics. A precise understanding of the pose in the environment is a crucial requirement for robots to navigate and operate accordingly. In the past research, localization problem has been addressed using variety of methods such as ultrasonic, laser range sensors, beacons, landmark based methods and map-based approaches. Vision has become more promising sensor technology for localization in the last three decades due to its ability to provide rich, detailed information. Monocular camera-based localization has been popular however depth cannot be observed, and it suffers from scale drift problem. Stereo vision can overcome these issues, but it again introduces computation complexity when estimating depth and synchronizing images.

This research describes a localization approach for indoor robots using low cost and information rich RGB-D cameras which can provide synchronous RGB image and depth map of the observed scene. These sensors have been popular in the recent research despite their limited depth range and filed of view. The proposed localization approach investigates the usage of multiple RGB-D sensors to be used as a strategy to localize indoor robots. Swarm of robots getting the support of a leader or a helper robot in localization and navigation is a common approach in indoor robotic surveillance applications. This work focuses on such an approach by localizing moving RGB-D sensors relative to another static RGB-D sensor in the environment.

The key input for localization is the colour and depth data extracted from the scenes observed by RGB-D sensors. Feature pointes extracted from the colour images are used to generate three-dimensional feature correspondences which are fed into RANSAC pose estimation algorithm. A RANSAC hypothesis transformation between the cameras is obtained iteratively by minimizing the SSD error between the two point clouds. Non-linear optimized pose is then calculated by optimizing the RANSAC estimated pose over all feature correspondences. The resulted pose of moving RGB-D sensor with respect to the static RGB-D sensor is exclusively depends on the significance of overlapped views and the accurate depth information extracted from the sensors.

The performed experiments and analysis of this work demonstrate similar or improved localization accuracy and enhanced range of operation compared to another RGB-D to RGB localization method. Effort taken to test the accuracy of this proposed work by means of popular RGB-D datasets available in the field also support as a study of its applicability in 6-DoF robot localization. Analysis shows that the proposed localization approach can be used to reasonably localize fast moving robots even in reduced feature environments. The same analysis can be applied to evaluate the suitability of this work in augmented reality applications. Hence, this research contributes towards robust localization approaches for robots to collaboratively operate in indoor environments and as a localization approach for RGB-D sensors to use in AR applications.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Signature: .....

Print Name: Nalika Damayanthi Nahinnage Dona

Date: 04-12-2019

Acknowledgements

I would like to acknowledge my warmest gratitude to my supervisors, Prof. Tom Drummond and Dr. Wai Ho Li for their enthusiasm and support throughout the research. They are inspiring researchers in robotics and computer vision. Their friendly guidance and expert advice have been invaluable throughout all stages of the work. I am truly grateful for the extended discussions and valuable suggestions I received from them to carry out this research work and appreciate their patience towards the mistakes and delays I made during the process.

I am grateful to Winston Yi, Vincent Lui and Dinesh Gamage for their valuable discussions and friendship during the development of this work. I am grateful to all of those with whom I have had the pleasure to work during this research.

Special thanks are due to my husband Rameesha De Silva Weerasingha, for his continuous support and guidance. I would like to thank my parents, whose love and guidance are with me in whatever I pursue.

Table of Contents

List of Figures

List of Tables

1 Introduction

Enhancement in computational resources and the availability of advanced sensors enable development of versatile robots that can operate in dynamic environments cooperatively with humans nowadays. Knowledge about the environment is a critical issue for autonomous mobile robots. Advances in research and development related to robotics and sensing technologies over the past few decades have given a great effort to address most of the promising challenges in developing robot systems those who perceive like human.

For mobile robots, the capability of localizing themselves in the environment is highly demanded. Robot localization techniques consider a wide variety of perception models. In recent years odometry sensors such as Global Positioning Systems (GPS) and Inertial Measurement Units (IMU) have been widely used for estimating the motion of mobile robots. Optical wheel encoders are basic odometry sensors and widely used due to their low cost and simplicity. The use of computer vision for localization has been investigated for several decades. Even though most researches pay more attention to other sensors such as laser range-finders and sonar, vision is still an attractive choice of sensors because cameras are information rich, compact and cheaper. With the availability of information rich cameras like RGB-D sensors robots are able to perceive further data about the operating environment which enhances their localization capability. Hence, this thesis introduces an RGB-D vision based localization method for indoor mobile robots.

1.1 Indoor robot Localization

Localization is identified as the problem of estimating the pose, i.e. the position and orientation of mobile robots given the sensor readings and the map of the environment. For mobile robots operate in indoor environments, localization is the fundamental challenge. The tasks such as object recognition, object tracking, navigation and motion planning are all based on the robot knowing its position in the environment. Once the robot identified objects in the environment it is an essential task for robots to know the position of targets so that it can either reach them or keep track of them. Hence, a precise understanding of its pose in the operating environment is more important when the robot has to execute such commands. Moreover, in SLAM applications robots need to keep track of their position and orientation in addition to building accurate maps and localizing themselves in the map simultaneously. Robustness in localization and navigation task depends on the reliability of acquisition of sensor data.

Two main strategies of localization are relative localization and absolute localization. Dead-reckoning [1] and inertial navigation techniques [2] are examples for relative localization strategies that estimates the robot's current position based on a previous or fixed position by integrating speed estimations from the sensors such as accelerometers, gyroscopes, wheel encoders etc. However dead-reckoning is error-prone with time and distance due to integrated noise and drift in wheels causing errors when operating in uneven terrain that can significantly affect the accuracy of the position estimation by cumulating odometry error.

In contrast to dead-reckoning, robots can benefit from absolute localization by knowing its direct position in the environment. Absolute localization strategies estimate the robot pose with respect to a global reference frame independent of time and initial position and hence reduce the accumulated error. Beacons [3] , landmark based technologies and popular Global Positioning System (GPS) technique [4] which based on satellite signals are examples for absolute localization strategies. The main drawback of landmark based techniques is robot localization and hence the operating domain solely depends on the landmarks. Furthermore, GPS based systems give low accuracy if not integrated with other sensors and not suitable to operate in indoor environments.

Mobile robot localization in indoor environments where GPS technology is not supported has been addressed by using various other sensors. Enormous research effort has expended in using range sensors such as ultrasonic [3, 5, 6], laser [7-9] and RFID [10, 11] to localize and navigate mobile robots. During past few decades laser range sensors have become one of the most attractive sensors for localization and map building due to their high accuracy.

Map based localization or model-matching [8, 12, 13] is another absolute localization strategy that uses prior information or map of the environment to position the robot using online sensor inputs. Drawbacks of this approach would be the need of enough sensor information to compare with the map to determine the position and require large amount of processing power.

Above described absolute and relative position estimation systems could be used with multi-sensor fusion approaches to come up with more accurate pose

estimations. In [14] Tsai uses extended Kalman filtering to localize a mobile robot by fusing information from a multisensorial dead-reckoning subsystem and ultrasonic localization subsystem. The multisensorial dead-reckoning system provides absolute and relative robot heading measurements which then are combined with ultrasonic time-of-flight (TOF) measurements to update the vehicle's position. Another robust localization method is suggested by Goel et al in [15] by fusing calibrated odometry with gyroscope and GPS data to mitigate the localization error caused when using absolute and relative localization systems alone. Fusion of several input measurements such as sonar, laser, odometry were also used with a priori map to obtain a refined robot position and orientation [14, 16, 17].

1.2 Vision Based Localization

Vision has become a more popular and promising sensing technology by increasing the scope of applications in autonomous robotics domain such as visual odometry [18], localization [19], autonomous navigation [20, 21], map building [22], path following and surveillance applications. Compared to other on-board sensing techniques, vision-based approaches demand attention due to their ability to provide rich, detailed information about the environment which may not be possible with combination of other types of sensors. Furthermore, vision sensors are low cost, light and compact, easily available and have low power consumption making them very attractive to be used in robot localization.

The suitability of various vision systems including single camera [22, 23], stereo camera pairs [24, 25], multiple cameras [26, 27] and RGB-D cameras has been experimented over the past few decades for robot localization and navigation. The problem with single cameras is it doesn't provide any information about the depth, hence multiple images from different viewpoints are required to get the 3D location of features. Stereo pairs on the other hand can provide 3D location of the observed features however, consecutive image acquisition from stereo pair and matching the feature is slightly more complicated than single camera. Use of multiple cameras increase the overall field of view hence robots can enhance their operating domain. However, one disadvantage of using multiple cameras is their high computational cost.

Research in vision based localization became more active with the availability of RGB-D cameras [28, 29] which provide video (RGB) along with per-pixel depth

information. These inexpensive depth cameras have made available dense 3D point clouds, which were previously only available with more expensive sensors like time-of-flight cameras or 3D laser range finders. In this research we use depth cameras to localize mobile robots in indoor environments.

1.3 Contributions

Accurate and robust localization is a key factor for robots operating in complex environments. The research work in this thesis focuses on a different localization method to use in augmented reality applications and indoor swarm robot applications such as formation control and surveillance. The proposed localization approach is based on a pair of information-rich RGB-D sensors where one RGB-D sensor is localized relative to another RGB-D sensor.

The proposed work aims to localize the two or more sensors while one sensor is kept static in the environment and the other sensor is moving relative to the static sensor. This approach is applicable to a swarm of robots operating in indoor environment while there is an RGB-D camera available in the environment or on a static helper robot so that the robots can be localized with respect to the fixed sensor. This method is also applicable in augmented reality applications where a human user moves an RGB-D device which is then localized using another RGB-D sensor. It is also useful in controlling formation of a swarm of robots who are assembled with RGB-D sensors and navigate in indoor environment.

The proposed localization approach is evaluated through a series of experiments carried out in a laboratory environment using Microsoft Kinect version-1 sensors. The experimental setup ensures that the Kinect sensors observe sufficient visual

features in their scenes. The proposed localization approach is analyzed qualitatively and quantitatively contributing to six datasets extracted based on three scenarios each obtained separately with visually dense and sparse featured scenes. The image sequences in each dataset consist of

1. Raw data with color and depth images and accelerometer data
2. Ground truth data obtained with manually estimated 6 Degree of Freedom (6- DoF) of RGB-D sensors.

In the localization scenario described above the sensors are moved so that they maintain a scene overlap in their field of views (FOV). Also, the approach assumes that there are sufficient visual features available in the camera scenes to maintain the localization.

We contribute by evaluating our proposed localization method comparing with previously suggested RGB to RGB-D localization approach. The collected datasets are tested on this previous localization method and a quantitative analysis is provided afterward. We also contribute by qualitatively investigating the appropriateness of this approach to be used in 6-DoF localization and AR applications by evaluating against a publicly available dataset.

1.4 Thesis Overview

In this chapter, some of the concepts for indoor robot localization including vision-based localization were discussed and the contribution from this work explained. In the next chapter, a review of the previous work done in the related research is discussed more technically. This review is not a complete survey of

the fields involved but does discuss the key concepts and issues involved in RGB-D based research.

With the survey as the base, Chapter 3 discusses and elaborates the concepts used in this research. The methodology of proposed RGB-D based localization approach is described in five detailed sections including a study of the related technology used.

Having described the methodology, the implementation procedure of the proposed work and the experiments conducted to evaluate the concept are presented in Chapter 4. The methods used to prove the completed work with regards to different aspects of its usage are explained in three experimental scenarios.

With the experiments conducted in Chapter 4, the collected results are presented with a comprehensive analysis in Chapter 5. Categorizing into three experimental scenarios, obtained results are compared qualitatively and quantitatively with a previously done localization approach and a popular RGB-D dataset.

With the compared results, the success and failures of proposed localization approach and its relevance in robot localization and augmented reality applications are explained in Chapter 6. In the final chapter, the future work for this research is outlined, and the conclusions of the work presented.

2 Literature Survey

Vision based localization has been remarkably improved over the past three decades. Navigating a robot in an indoor cluttered hallway was hardly possible about 25 years ago but it is not much of a challenge in the recent research. Monocular camera-based localization [22, 23] and SLAM [30, 31] approaches have been proposed by number of researchers, however the depth is not observable from just one camera and the scale of the map and estimated trajectory is unknown. Due to the scale drift problem, monocular approaches can fail when pure rotations are performed in exploration.

Conversely, using stereo or RGB-D camera makes it possible to solve the aforementioned issues with monocular localization. There have been many studies on localization and visual odometry in indoor static environments using stereo vision-based approaches. Stereo vision has been used to acquire three-dimensional vision by simulating human binocular vision on a pair of monocular cameras. Three dimensional images are captured through disparity images, from which the depth information can be obtained. In [32] stereo vision is used for localization and SLAM using a stereo–camera that acquires the position of known landmarks, in indoor environment. A large scale SLAM system with stereo cameras is presented in [33], where the scale-drift problem is avoided using a fixed baseline stereo. In [34], a researcher investigates a visual odometry method for autonomous ground vehicles based on dense disparity images from stereo cameras. Even though stereo vision avoids the scale estimation issue it introduces a computational overhead of depth estimation and image synchronization.

2.1 RGB-D Sensor Based Research

In contrast, using RGB-D sensors for extracting depth information has become popular among robotics and computer vision community. Although the laser scanners can provide accurate depth data, they have become less popular due to being high expensive and heavy. The RGB-D cameras on the other hand can provide both RGB and depth information having benefits of laser and vision sensing together. Due to their relatively low cost, these sensors have been extensively popular among robotics research community in the last few years. Microsoft Kinect [35] is one of the most popular RGB-D sensor developed for video game purposes. Asus Xtion sensor [36] is a more compact alternative with lower weight and powered only via USB connection itself.

In the past few years, visual localization and mapping by using RGB-D cameras has become one of the most active research fields despite their limited depth precision and field of view provided by RGB-D cameras. Even though the Simultaneous Localization and Mapping (SLAM) problem has been addressed broadly using other sensors such as laser, sonar and monocular and stereo cameras, recently appeared low cost, light weight RGB-D cameras providing dense, high frequency depth information are taking a great attention towards solving SLAM problem. Henry *et al* [28] introduces RGB-D mapping, a framework for using RGB-D cameras to generate dense 3D models of indoor environments. This approach aligns two consecutive frames using RGB-D ICP, enhanced ICP algorithm that takes advantage of the combination of RGB and depth information.

KinectFusion proposed by Newcombe *et al* [37] is an outstanding recent approach for real-time dense volumetric reconstruction of complex room-sized scenes using a single handheld Kinect sensor. This real time parallel tracking and mapping system running on GPUs provides accurate and robust tracking of the camera pose by aligning all depth points with the complete scene model and up-to-date surface representation by fusing all registered data. However, the proposed system works well only for mapping medium sized rooms and not suitable for reconstructing large scale models that needs too much memory and would lead to reconstructions with inevitable drift which would cause misalignments upon trajectory loop closures. Efficiently performing automatic re-localization when the tracking has failed in environments with a low number of 3D geometric features is another challenge for KinectFusion.

Kintinuous presented in [38] overcomes KinectFusion's challenge of limiting the mapping to medium size room by making the region of space being mapped can vary dynamically. In KinectFusion, tracking and surface reconstruction is restricted to the region around the point of initialization of the volumetric representation of the scene, known as the truncated signed distance function (TSDF). In contrast, Kintinuous permits the area mapped by the TSDF to move over time by virtually moving the TSDF with camera pose allowing continuously augment the reconstructed surface in an incremental fashion as the camera moves. Kintinuous also present a solution to overcome KinectFusion's inability to function in featureless or reduced featured environment by incorporating a feature based visual odometry system described in [39].

Whelan *et al* describes an extension to Kintinuous in [40], an improved GPU implemented camera pose tracking method and an analysis of combination of various RGB-D visual odometry estimation techniques for robust camera tracking. Additionally, they introduce RGB color integration method into the KinectFusion reconstruction process. However, neither of Kintinuous [38] nor the extension to Kintinuous [40] well addresses the issue of dealing with very high camera velocity or a lack of both visual and depth features.

RGB-D SLAM system presented by Endres *et al* [41] is one of the recent popular approaches that can robustly deal with challenging scenarios such as fast camera motions and feature-poor environments while being fast enough for online operation. In addition to the system they present a thorough experimental evaluation on a publicly available benchmark dataset and also provide an open source implementation of their system for comparison.

Hu *et al* [42] propose a robust algorithm for SLAM using RGB-D sensors which builds local maps either using vision only (RGB-BA) or vision and range depending (RGB-D-BA) on the different scenarios, then a map joining algorithm is applied to combine all the local maps. By applying the heuristic switching, the algorithm is able to handle various failure modes associated with RGBD-BA. Due to the significant deduction in computational cost in map joining strategy, the proposed algorithm is more applicable to large scale RGB-D SLAM. However, this approach has some short-comes such as quick loss of feature tracking, inability to handle the scenes with feature poor planar surfaces etc.

RGB-D sensors are significantly applied in odometry and ego-motion estimation in the past few years. Visual odometry uses camera images to estimate the distance travelled similar to odometry estimation which uses wheel encoders on mobile robots to estimate the change in robot position. Visual odometry enhances a robot's navigational accuracy whereas the rotary encoder based odometry suffers from precision problems due to accumulating errors when the robot slips or slides while operating in non-smooth surfaces.

Sturm *et al* [43, 44] provide a large dataset containing RGB-D image sequences and ground-truth camera trajectories obtained from a high accuracy motion capture system. The dataset has been recorded with a hand-held Kinect camera and also with a Kinect mounted on a Pioneer 3 robot. This has been a popular dataset among RGB-D SLAM community as a benchmark for evaluating the SLAM systems.

Steinbrücker *et al* [29] introduce an energy-based visual odometry method to estimate the rigid body motion of a handheld RGB-D camera for a static scene. In their approach the rigid body motion is represented in terms of its Lie algebra of the twist which maximizes the photo-consistency of the warped images is found so that the warped consecutive images exactly match each other. Their method of visual odometry is validated using the RGB-D dataset by Sturm *et al* [44] and proven to be faster and performing better results than Generalized-ICP (GICP) [45]. Kintinuous system [40] described above also uses a high-performance GPU implemented version of this energy based visual odometry approach.

Handa *et al* [46] present a collection of hand-held RGB-D camera sequences within synthetically generated environments as a new benchmark aimed at RGB-D visual odometry, 3D reconstruction and SLAM systems. This synthetic dataset not only provides ground truth camera pose information for every frame but also provides a means of quantitatively evaluating the quality of the final map or surface reconstruction produced. The realistic trajectories for use in synthesized sequences are obtained by running the Kintinuous system [40] in a standard real environment and taking the estimated camera path as ground truth trajectories.

Dryanovski *et al* [47] introduce a system for visual odometry that does not rely on frame-to-frame or sliding window techniques. In their approach, the 3D sparse feature points in the incoming RGB-D images are aligned using ICP [48] against a global model dataset of 3D features updated through a probabilistic Kalman Filter framework. This approach takes less computational effort and does not include any intensive GPU based computations, hence increases the performance of the overall system. The system is capable of loop closure in room environment with a sufficient accuracy, however further effort is needed on correcting the systematic error in the depth image to avoid performance dropping on visual odometry.

2.2 Usage of Multiple RGB-D Sensors

As far as the research concerned on using multiple RGB-D sensors there is not much effort on using them for localizing each other. Most of the multiple RGB-D based research is focused on people tracking, object detection and recognition, 3D object and scene modelling.

The system for scanning 3D full human bodies proposed by Tong *et al* [49] use multiple Kinects to scan different parts of the human body so that they can be used to observe the body closely then to obtain high quality data. In order to save the original data quality without degrading due to interference issue they maintain non-overlapping regions of the sensor views while scanning human body. They also use a two stage non-rigid registration of the captured data to address the challenge of human body being stirred during the scanning process. This method can deal with non-rigid alignment with loop closure constraints and complex occlusions. However, there are unnatural bending on the body parts due to misalignment and complex occlusions. The quality of the reconstructed models can also be improved further by using super resolution approaches.

The system presented by Alexiadis *et al* [50, 51] on the other hand reconstructs full-geometry 3D textured mesh of moving humans in real time using multiple Kinect sensors. In this approach, separate textured meshes from multiple RGB-D streams are generated using ICP based alignment and fast zippering algorithm. Later in [51] they have improved the system by implementing in CUDA to fuse the information from all the Kinects to produce watertight models in real time.

The surveillance system proposed by Almazan and Jones [52] uses multiple Kinects in non-overlapping configuration to track people in complex environment. This approach uses mean-shift algorithm for tracking people where the position of the search window is determined using Kalman filter.

Most of the RGB-D mapping, SLAM and visual odometry approaches described above use single moving sensor. There are not many multiple RGB-D SLAM

approaches in the research except the work suggested in [53]. The framework for corporative localization and mapping for autonomous flights proposed by Loianno *et al* [53] uses multiple Asus Xtion RGB-D sensors, however the localization task is achieved by a monocular visual odometry algorithm whereas the depth information is used to estimate the scale factor associated with the visual information. The reason for not using multiple RGB-D sensors in aforementioned research could be their limited field of view and the interference. However, multiple RGB-D sensors can be employed to achieve certain goals despite having the interference and limited field of operation. The related literature doesn't witness significant effort in collaborative use of multiple RGB-D sensors localized relative to each other. Most of the multi sensor research focuses on object recognition, people tracking, 3D object and scene modelling. In this research we focus on localizing RGB-D sensors relative to another RGB-D sensor to be used in multi RGB-D applications such as swarm robot surveillance, collaborative augmented reality applications etc.

3 Methodology

According to the research review on using RGB-D sensors described in chapter 2, most of the applications use single moving sensor or multiple static sensors in tasks such as mapping and SLAM, odometry and egomotion, scanning and 3D reconstruction of objects, detecting and tracking moving objects etc. Most of above research focuses on fusing RGB-D datasets using ICP and other approaches. Using multiple RGB-D sensors in aforementioned research has drawn lack of attention due to the interference of depth data while using multiple sensors and the limited range and field of view of the commonly available RGB-D sensors.

The research work presented in this thesis investigates localization of an RGB-D sensor relative to another RGB-D sensor. Localization of the two sensors is performed offline based on two scenarios.

1. Systematically moving an RGB-D sensor relative to a static RGB-D sensor
2. Freehand moving an RGB-D sensor relative to a static RGB-D sensor.

The localization approach proposed here is applicable for mobile robotic applications where there is an RGB-D camera available in the environment or on a static helper robot, especially in swarm robotic applications such as formation control and surveillance. The proposed method is also applicable in collaborative multi use augmented reality applications where a human user moves a hand-held RGB-D device which is then localized using another RGB-D sensor.

In both scenarios, the cameras are positioned so that they maintain some overlapping in their FOVs.

3.1 Pose Estimation

The problem of localizing an RGB-D sensor with respect to another RGB-D sensor is addressed by estimating the transformation in 6DoF between the two sparse 3D point clouds obtained from two sensors by filtering and combining the feature matches and depth information. Microsoft Kinect sensors provide color and depth images at 640 x 480 resolution. The pose estimation algorithm assumes that the images from both Kinects have some overlapping area so that the extracted features from the color images can be matched against each other. The feature correspondences extracted from pairs of keyframes are used to generate sparse 3D point clouds by combining with corresponding color and depth data. Then the two point-clouds are aligned, and the best transformation is iteratively estimated using Random Sample Consensus (RANSAC) algorithm. This transformation is then optimized over all RANSAC inliers hence non-linear optimized pose is obtained. This pose is taken as the transformation or the pose of two Kinect sensors relative to each other.

The Figure 3.1 shows the procedure of estimating the relative pose. The following sub sections describe each step in the diagram in more detail.

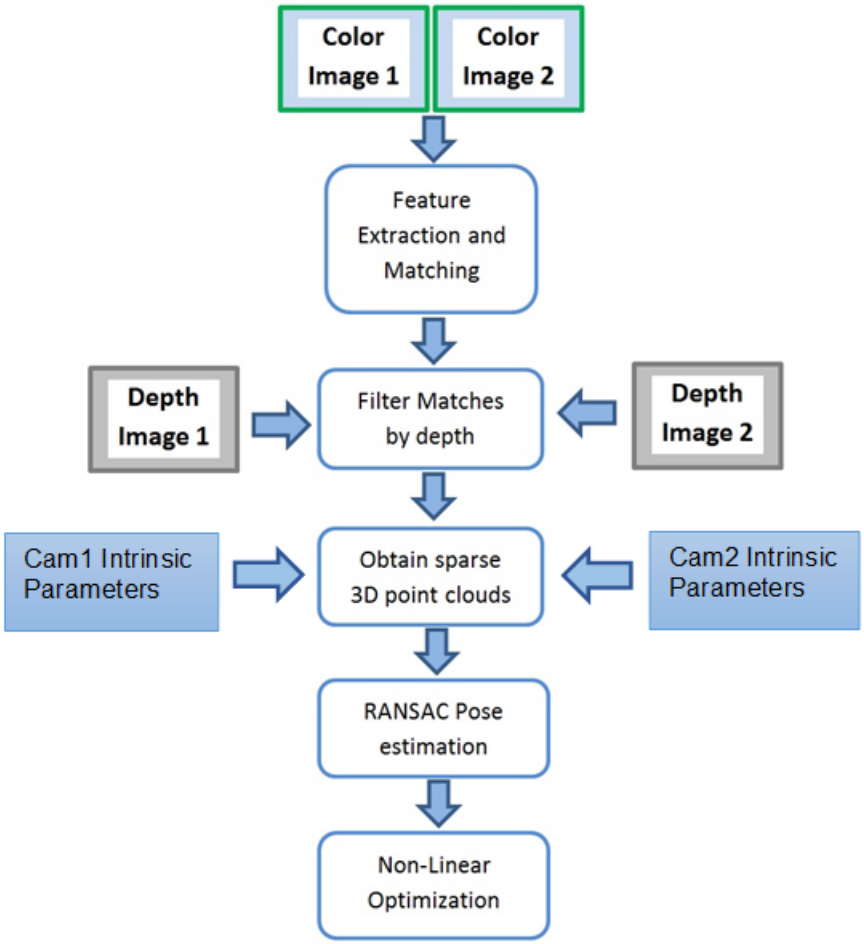


Figure 3.1: The process of estimating relative pose between two RGB-D sensors

3.1.1 Feature Extraction and Matching

In order to extract visual features from the captured frames, we surveyed on several feature detection and matching techniques. In our approach, a significant variation in scale and orientation is expected between the matched keyframes because the captured frames are not the successive frames of a single camera, but

pairs of synchronized frames captured from two different cameras positioned apart in the environment. Recently researchers have taken effort to introduce keypoint detectors and descriptors which are faster and robust to scale and orientation. FAST [54] , SIFT (Scale Invariant Feature Transform) [55], SURF (Speeded-Up Robust Features) [56], BRIEF (Binary Robust Independent Elementary Features) [57] and ORB (Oriented FAST and Rotated BRIEF) [58] are among the popular general purpose keypoint detectors and descriptors.

SIFT features are invariant against scaling, image rotation and robust across changes in lighting conditions and camera viewpoint, addition of noise and a range of affine distortion. However, SIFT features are computationally much more demanding than other feature descriptors and not a good choice for robots that require real-time operation having limited computational resources. SURF on the other hand inspired by SIFT but takes a lower computational cost and more robust against image transformations. BRIEF is a recently developed feature descriptor which has similar performance as SIFT being robust to lighting, blur and perspective distortion. The major drawback of BRIEF is the lack of rotational invariance and being very sensitive to in-plane rotation.

ORB is a combination of FAST keypoint detector and BRIEF feature descriptor introduced recently by Rublee *et al* and significantly faster and lower in terms of computational cost compared to SIFT and SURF. ORB consists of oriented FAST which is an efficiently-computed orientation component added over the widely used FAST corner detector. Since FAST does not provide multi-scale features, ORB applies a scale pyramid of the image and generates FAST features at every level of the pyramid.

We use ORB feature detector and descriptor in our approach because of being relatively faster, computationally efficient, resistant to noise and robust to translation and rotation of the features. We compute two sets of keypoints $k_{rgb1}, k_{rgb2} \in K_{rgb}$ on synchronized pairs of Kinect RGB color frames ($i_{k1}, i_{k2} \in I$ for $k = 1..N$ where N is the total number of RGB frames) using ORB keypoint detector implemented in the Open Source Computer Vision (OpenCV) library [59] . Then we compute a set of ORB feature descriptors F_{k1}, F_{k2} for the color frames and apply a brute-force descriptor matcher with the Hamming Norm and cross checking the correspondences, which results in a set of matches $m_i \in M$ where $i = 1..N$ and N is the total number of frame to frame matches.

Figure 3.2 shows two sample RGB color images from our experiments with matched ORB feature correspondences.

Figure 3.2: Matched ORB feature correspondences are filtered according to relevant depth information from two different RGB-D sensors

3.1.2 Generating 3D Feature Correspondences

The depth measured by Kinect is often degraded by occlusion, limited field of view, and sensor noise and especially in our application due to multi-sensor interference. When multiple Kinect cameras are pointing at the same scene, the projected IR dot patterns are interfered with one another resulting in invalid or zero depth values at certain pixels on the depth image. This could also happen due to reflecting surfaces such as mirrors or from light absorbing black surfaces. Therefore, our approach of localization mostly gives low density point-clouds due to the effect of multi-Kinect interference. Figure 3.3 shows an integrated color and depth image from a Kinect sensor with some patches in black where the corresponding depth value is zero or invalid.

Figure 3.3: Integrated color and depth images from Kinect

Given the intrinsic parameters of the color cameras of two Kinects and the depth values at each pair of 2D feature correspondence, two sets of sparse 3D point-correspondences are obtained. The ORB feature matches (m_i) in the color images are filtered by using respective depth data ($d_{k1}, d_{k2} \in D$ for $k = 1..N$ where N is the total number of depth frames) to obtain the 3D coordinates of each matched feature point.

According to the geometry of pin-hole camera model, the relationship between Kinect pixel coordinates (u, v) and camera coordinates (x, y) are given by,

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1)$$

Where $fx, fy, u0, v0$ are intrinsic camera parameters. Kinects' color cameras are calibrated separately to obtain the intrinsic camera parameters and hence the camera matrix. The calibration method used for Kinect sensors is explained in Section 4.2.

In the camera matrix, $(\boldsymbol{u0, v0})$ is the principal point, the center of the image plane. According to Figure 3.4, 3D world coordinates $\boldsymbol{P_i}$ ($\boldsymbol{i = 1..N}$ where $\boldsymbol{N}$ is the total number of matched 3D points in a frame pair) of a feature point can be taken as $(\boldsymbol{X, Y, Z})$ where $\boldsymbol{Z}$ is the corresponding depth value given by RGB-D sensor. If the pixel coordinates of the feature point is $(\boldsymbol{u, v})$ then according to Equation (1),

$x = (u - u_0)$ and $y = (v - v_0)$.

Then the 3D world coordinates $(\boldsymbol{X, Y, Z})$ of the feature point can be estimated as

$$X = d * (u - u_0) \,/\, f_x \quad (2)$$

$$Y = d * (v - v_0) \,/\, f_y \quad (3)$$

$$Z = d \quad (4)$$

Where d is the measured depth at pixel (u, v).

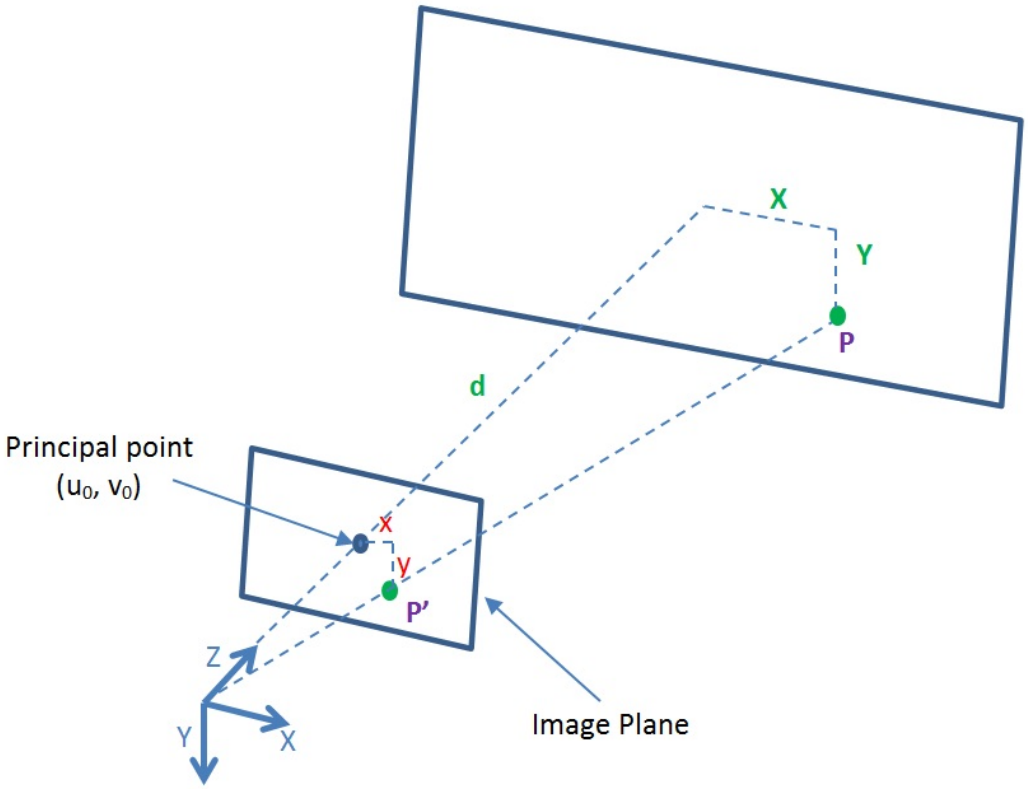


Figure 3.4: Given the depth (d) at a pixel point P'(u, v), 3D world coordinates (X, Y, Z) of the point of interest (P) is calculated using camera intrinsic parameters.

Likewise, a sparse set of point-wise 3D correspondences $P_{k1}, P_{k2} \in P$ between two Kinect frames are determined using image coordinates and the measured depth of the filtered feature matches.

3.1.3 Iterative Pose Estimation using RANSAC

Given two sets of 3D point correspondences, Random Sample Consensus (RANSAC) [60] is a well-known approach to estimate the best transformation and the set of inliers which has been extensively used by the researchers in registering 3D point clouds [39, 41]. RANSAC is one of the best ways to fit a model to experimental data and this algorithm works well even when the data is noisy. When estimating the best transformation out of given 3D point pairs, it iteratively finds the transformation by considering random three point pairs, which is the minimal number from which a rigid transformation in $SE(3)$ can be obtained.

The data extracted from Kinect frames tend to be noisy and hence finding the precise transformation that aligns all the point-correspondences is a challenging task. Therefore, the aim is to find the best estimation that aligns a maximum number of point correspondences within a given Euclidean distance threshold. RANSAC is used to find this best estimation iteratively for a given number of iterations. In our approach, the rigid body transformation of two sets of 3D point correspondences is found using the method described in Section 3.2.4. For each pair of Kinect frames $((i_{k1} + d_{k1}), (i_{k2} + d_{k2}))$ the matched 3D feature correspondences (P_{k1}, P_{k2}) are fed into RANSAC algorithm in which the transformation $T_i = [R_i, t_i]$ (where $i = 1..N$) is found using randomly selected s point pairs for a given number of iterations N. In each iteration, the resultant transformation is used to re-project the remaining 3D feature correspondences and calculate the re-projection error e_{ssd} , i.e. Sum of Squared Differences (SSD)

error based on the Euclidian distances between each re-projected point $P_{k1}^{t} = T_i P_{k1}$ and the original point P_{k2}. The SSD error for the n^{th} point is found as,

$$e_{ssd,n} = \sum_{a=x,y,z} \left[(P_{k1}^{t})_{n,a} - (P_{k2})_{n,a} \right]^2 \tag{5}$$

For each transformation hypothesis T_i , a 3D point pair is considered to be an inlier match if e_{ssd} for that point is below the given threshold t. For each inlier we estimate a consensus score C_i so that,

$$C_i = \sum_{n\_inliers} \left(1 - \frac{e_{ssd,n}}{t} \right) \tag{6}$$

After computing N transformation hypothesis, the transformation T_i that maximize the re-projection consensus score C_i is chosen as the best rigid body transformation hypothesis. The subset with the maximum number of inlier feature points are also recorded.

In our algorithm, RANSAC is used to find the best hypothesis and its consensus set of inlier matches in 500 iterations so that the re-projection error lies within 3cm. The estimated rigid body transformation is optimized by re-estimating over all the RANSAC inlier matches which is then repeatedly refined using all 3D point correspondences.

3.1.4 Estimating Rigid Body Transformation

The 6DOF transformation between the coordinate frames of two Kinects can be thought of as a rigid-body motion and can be expressed as a rotation and a translation (R_k, t_k) where $R_k \in SO(3)$ and $t_k \in \mathbb{R}^3$.

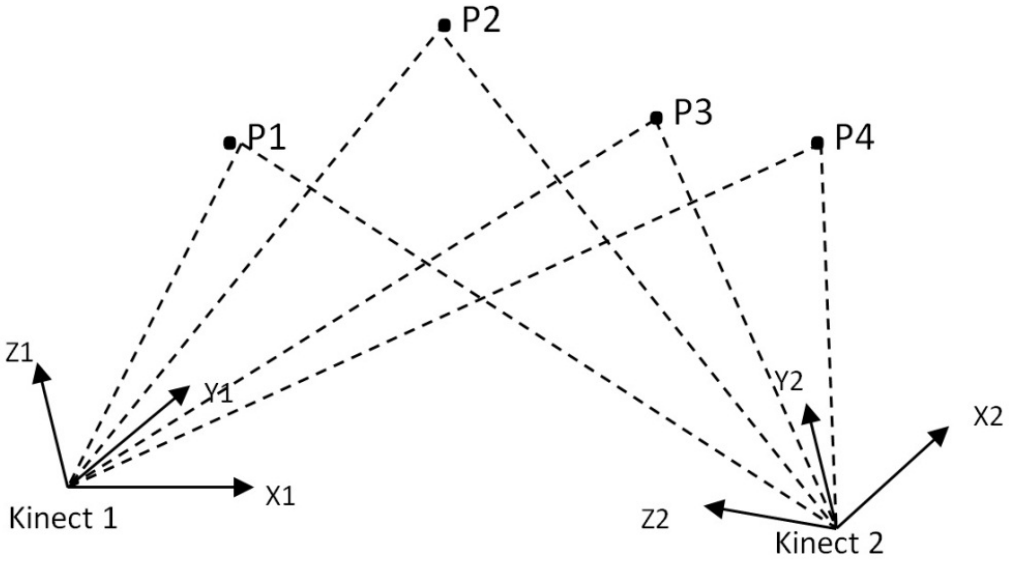


Figure 3.5: Three-D point correspondences measured from two coordinate systems. The transformation between two systems is to be found.

The transformation of Kinect2's coordinate frame relative to Kinect1's coordinate frame (M_{K2K1}) can be written as,

$$M_{K2K1} = \begin{bmatrix} R & t \\ 000 & 1 \end{bmatrix} \tag{7}$$

where $R \in SO(3)$ and $t \in \mathbb{R}^3$

We use the method explained by Horn [61] to find the translation and rotation of the keypoint sets from Kinect-1 frame to Kinect-2 frame.

The six degrees of freedom (6DOF) or in other words the freedom of movement of a Kinect sensor in three dimensional space comes with the degrees of freedom of its translation and rotation. The translation has three degrees of freedom i.e. the translation in three perpendicular axes X, Y, Z. The rotation provides another three DOFs as yaw, pitch, and roll as the rotation about these three axes. Given a large set of three-D point-correspondences from two Kinects, a subset of three point-correspondences which provides nine constraints would be sufficient to recover the six parameters in translation and rotation of the rigid body transformation.

Consider three pairs of point-correspondences $P1, P2, P3$ from Kinect-1 and $Q1, Q2, Q3$ from Kinect-2 as shown in Figure 3.6. For these three non-collinear point pairs, the rotation R can be solved by constructing two triads. Let the two triads in two coordinate systems to be $V1, V2, V3$ and $V1', V2', V3'$ respectively.

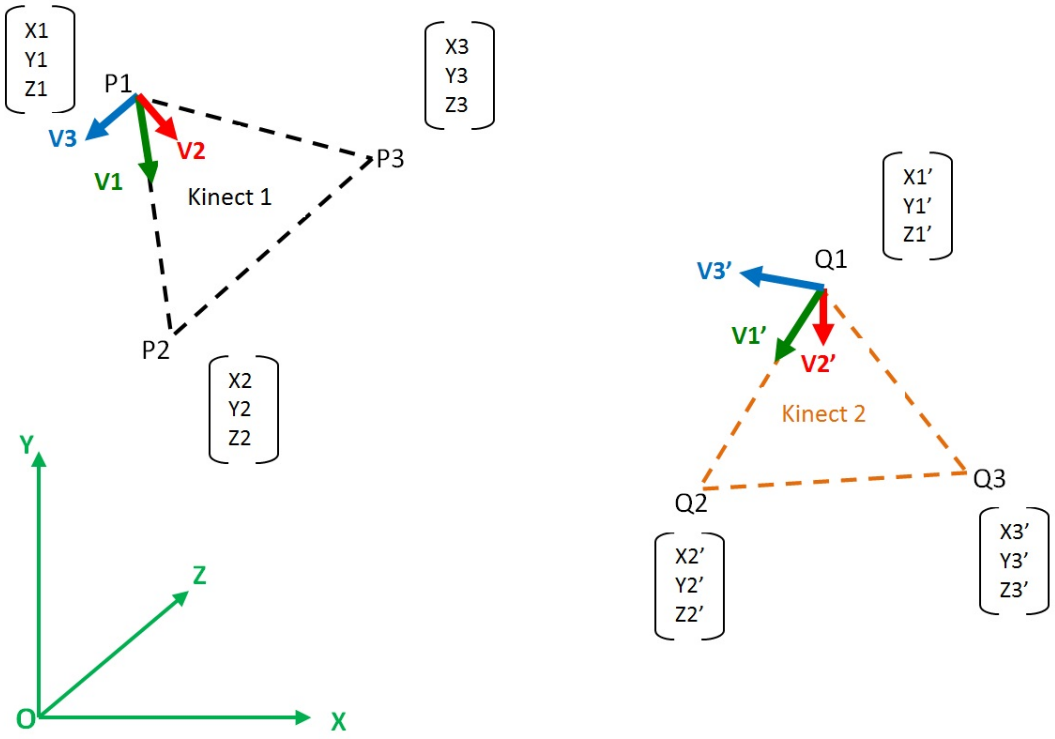


Figure 3.6: The three-D point-correspondences are used to construct two triads.

When constructing the triad using three points $P1, P2, P3$ in Kinect-1 frame, let the origin of the triad to be at the first point $P1$. The vector $V1$ is taken along the line $P1P2$ so that,

$$V1 = P2 - P1 \tag{8}$$

Then the unit vector along $V1$ will be,

$$\widehat{v1} = \frac{V1}{\|V1\|} \tag{9}$$

$V2$ is taken perpendicular to $V1$ and the vector $P1P3$ so that,

$$V2 = \widehat{v1} \ \hat{} \ (P3 - P1) \tag{10}$$

The notation ^ denotes the cross product between two vectors to get the perpendicular vector.

Then the unit vector along $V2$ is,

$$\widehat{v2} = \frac{V2}{\|V2\|} \tag{11}$$

To complete the triad, $V3$ is taken as orthogonal to both $V1$ and $V2$ axes such that the orientation satisfies the right-hand rule. So,

$$\widehat{v3} = \widehat{v1} \ \hat{} \ \widehat{v2} \tag{12}$$

This procedure is repeated for corresponding three points in Kinect-2 coordinate frame to obtain the second triad. If the unit vectors along the axes of two triads are $\widehat{v1}, \widehat{v2}, \widehat{v3}$ and $\widehat{v1'}, \widehat{v2'}, \widehat{v3'}$ then the rotation that aligns these two triads is also the rotation that corresponds to Kinect-1 and Kinect-2 coordinate frames. In other words, the rotation of two Kinect's coordinate frames takes $\widehat{v1}$ into $\widehat{v1'}$, $\widehat{v2}$ into $\widehat{v2'}$ and $\widehat{v3}$ into $\widehat{v3'}$. Now the below M_1 and M_2 are formed by above unit column vectors.

$$M_1 = \begin{bmatrix} \vdots & \vdots & \vdots \\ \widehat{v1} & \widehat{v2} & \widehat{v3} \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{13}$$

$$M_2 = \begin{bmatrix} \vdots & \vdots & \vdots \\ \widehat{v1'} & \widehat{v2'} & v3' \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{14}$$

The product $M_1{}^T V$ gives the components of a given vector V in Kinect-1 frame along the axes of Triad-1. Then the mapping of this onto Kinect-2 coordinate frame is given as,

$$V' = M_2 \, M_1{}^T \, V \tag{15}$$

Therefore, the rotation R is found so that,

$$R = M_2 \, M_1{}^T \tag{16}$$

Matrices M_1 and M_2 belong to orthonormal (orthogonal) matrices since each of their columns is a unit vector and the columns are orthogonal. The matrix product of two orthonormal matrices is another orthonormal matrix. In addition, the inverse (or the transpose) of an orthonormal matrix is an orthonormal matrix. Therefore, the solved rotation R is also an orthonormal matrix that belongs to the orthogonal group.

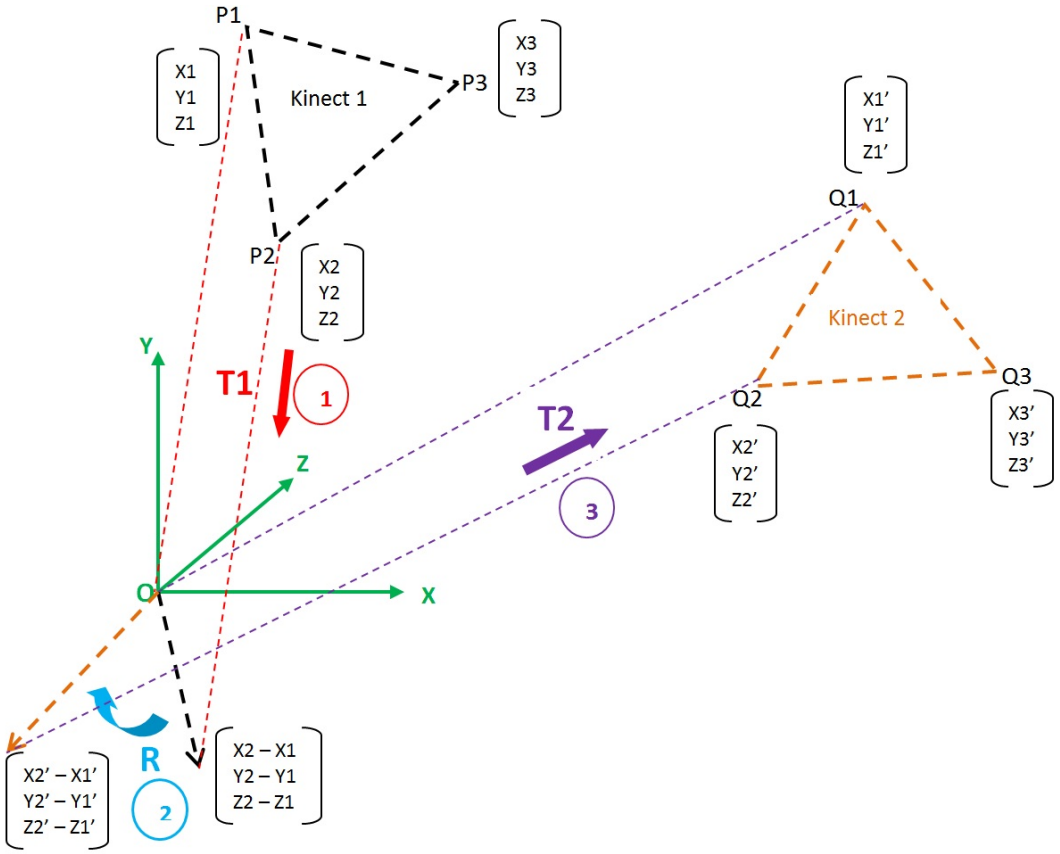


Figure 3.7: The transformation consists of a pure translations and pure rotation.

Consider three-D coordinates of point-correspondences $P1, P2, P3$ from Kinect-1 and $\mathrm{Q1, Q2, Q3}$ from Kinect-2 in homogeneous form as $[X1, Y1, Z1,1]^T$ $[X2, Y2, Z2,1]^T$ $[X3, Y3, Z3,1]^T$ and $[X1', Y1', Z1', 1]^T$ $[X2', Y2', Z2', 1]^T$ $[X3', Y3', Z3', 1]^T$ as shown in Figure 3.7.

According to Figure 3.7, the combined transformation can be expressed as a combination of two pure translations $(T1\ and\ T2)$ and a pure rotation R about the origin of Kinect-1 coordinate frame. The combined transformation E_{K1K2} is the transformation of 3D points-set 1 (from Kinect-1) relative to 3D points-set 2

(from Kinect-2). According to Figure 3.7, the combined transformation E_{K1K2} with two translations and the rotation can be expressed as,

$$E_{K1K2} = \begin{bmatrix} I & T_2 \\ 000 & 1 \end{bmatrix} \begin{bmatrix} R & 0 \\ 000 & 1 \end{bmatrix} \begin{bmatrix} I & T_1 \\ 000 & 1 \end{bmatrix} \tag{17}$$

If pure translation of point P1 at the origin of Kinect-1 frame is T1, then

$$T_1 = -\begin{bmatrix} X1 \\ Y1 \\ Z1 \end{bmatrix} \tag{18}$$

and the pure translation of point Q1 is T2, then

$$T_2 = \begin{bmatrix} X1' \\ Y1' \\ Z1' \end{bmatrix} \tag{19}$$

Because of the noise in 3D point data, the two triangles represented by two sets of 3D points may not be isomorphic. As such, considering a single point in determining translation would be more error prone, hence when finding the translation, the attention was paid for using the centroid of three points such that;

$$T_1 = -\begin{bmatrix} (X1 + X2 + X3)/3 \\ (Y1 + Y2 + Y3)/3 \\ (Z1 + Z2 + Z3)/3 \end{bmatrix} \tag{20}$$

$$T_2 = \begin{bmatrix} (X1' + X2' + X3')/3 \\ (Y1' + Y2' + Y3')/3 \\ (Z1' + Z2' + Z3')/3 \end{bmatrix} \quad (21)$$

Therefore, the translation is estimated so that,

$$T = Centroid\_2 - R * Centroid\_1 \quad (22)$$

Therefore, by using the estimated values T_1 , T_2 and R the combined transformation E_{K1K2} is determined according to Equation (17).

Considering homogeneous coordinates, if $P1 = [X1, Y1, Z1, 1]^T$ is a three-D point in Kinect1's coordinate frame and $Q1 = [X1', Y1', Z1', 1]^T$ is the corresponding three-D point in Kinect2's coordinate frame,

$$E_{K1K2}\, P_1 = Q_1 \quad (23)$$

E_{K1K2} is the inverse of the transformation of Kinect 2's coordinate frame relative to Kinect 1's coordinate frame, *i.e.*

$$E_{K1K2} = M_{K2K1}^{-1} \quad (24)$$

Therefore, combining with Equation (23), above Equation (24) can be re-written as,

$$M_{K2K1}\, Q_1 = P_1 \tag{25}$$

Therefore once E_{K1K2} is estimated, finding the camera transformation matrix M_{K2K1} is straight forward.

As described in Section 3.1.3, an approximate estimation of the pose and a set of inlier three-D point-correspondences are obtained iteratively by using RANSAC algorithm. The obtained estimation is fed into non-linear optimization algorithm to obtain a refined pose over all the inliers.

3.1.5 Non-Linear Pose Optimization

We optimize the RANSAC estimated 6DOF transformation over all the feature correspondences using Newton iteration method.

If the RANSAC estimated pose is E, then for a single point correspondence (P_1 and Q_1 as above) the error e is,

$$e = EP_1 - Q_1 \tag{26}$$

Where Q_1 is a measurement vector and E is a parameter vector. Therefore, in this optimizing technique we obtain the Jacobian J using the derivative of the error e with respect to six parameters of E, i.e. $\propto_i$ ($\propto_{1-3}$ = Translation parameters and $\propto_{4-6}$ = Rotation parameters) so that,

$$E_{t+1} = e^{\sum_{i=1}^{6} \propto_i G_i} E_t \tag{27}$$

Therefore,

$$\frac{\partial e}{\partial \propto} = \frac{\partial e}{\partial (EP_1)} \frac{\partial (EP_1)}{\partial \propto} \tag{28}$$

$\frac{\partial e}{\partial (EP_1)}$ can be simplified as 3x3 identity matrix.

$\frac{\partial (EP_1)}{\partial \propto}$ becomes $G_i(EP_1)$ where G_i is $SE(3)$ Generators.[62]

Therefore, $\frac{\partial e}{\partial \propto}$ or the Jacobian J is found as,

$$\frac{\partial e}{\partial \propto} = I\ G_i\ (EP_1) \tag{29}$$

According to six $SE(3)$ Generators as given below,

$$G1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad G2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad G3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$G4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad G5 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad G6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The Jacobian is derived as,

$$J = \frac{\partial e}{\partial \propto} = \begin{bmatrix} 1 & 0 & 0 & 0 & EP_1[2] & -EP_1[1] \\ 0 & 1 & 0 & -EP_1[2] & 0 & EP_1[0] \\ 0 & 0 & 1 & EP_1[1] & -EP_1[0] & 0 \end{bmatrix} \tag{30}$$

4 Implementation and Experiments

This chapter discusses how we implemented the proposed localization approach and provides a detailed description about the experiments conducted to evaluate the accuracy of our method.

4.1 Implementation

We implemented the pose estimation and pose optimization algorithms using C++ programming language using Microsoft Visual Studio 2012 Version 11.0 IDE. Our localization algorithm can be depicted in pseudo code as below.

Algorithm 1 Iterative Pose Estimation using RANSAC and Non-Linear Optimization

Input: RGB-D Frame of Kinect-1 ($RGBD_1$), RGB-D Frame of Kinect-2 ($RGBD_2$), t (error threshold), $Max\_Iterations$

Output: 6DOF Pose of Kinect-2 with respect to Kinect-1

k_{rgb1} ← Compute ORB Keypoints for $RGBD_1$

k_{rgb2} ← Compute ORB Keypoints for $RGBD_2$

F_{k1} ← Compute ORB Feature Descriptor for $RGBD_1$

F_{k2} ← Compute ORB Feature Descriptor for $RGBD_2$

m_i ← apply a brute-force descriptor matcher with the Hamming Norm to F_{k1} and F_{k2}

Filter ORB feature matches (m_i) according to depth data from $RGBD_1$and $RGBD_2$

P_{k1}, P_{k2} ← Using intrinsic camera parameters, obtain N sparse 3D feature correspondences

RANSAC:

Input: P_{k1}, P_{k2}

Output: T_{RANSAC} ← 6DOF Pose of Kinect-2 with respect to Kinect-1

Repeat

- Randomly select 3 feature correspondences
- $T_i = [R_i, t_i]$ ← Obtain 6DOF pose using 3 feature correspondences
- **For** $i \leftarrow 1$ **to** N **do**
  - $P_{k1}^t = T_i P_{k1}$ ← Reproject feature correspondences
  - e_{ssd} ← Calculate sum of Squared Differences (SSD) error
  - **If** $e_{ssd} < t$ **then**
    - Increase $Consensus_{Score}$ by $(1 - \frac{e_{ssd}}{t})$
    - $Inliers++$ ← Add $P_{k1}$and P_{k2} to inlier feature correspondences
- **If** $Consensus_{Score} > Best\ Inlier_{Score}$ **then**
  - $Best\ Inlier_{Score} = Consensus_{Score}$
  - $T_{RANSAC} = T_i$
  - $RANSAC\ Inliers = Inliers$
- $RANSAC\ Matches$ ← Filter out the best matches
- **Until** $(Iterations > Max\_Iterations)$

NON-LINEAR OPTIMIZATION:

Input: $T_{RANSAC}, P_{k1}, P_{k2}$, $RANSAC\ Inliers$, t

Output: $T_{optimized}$ ← 6DOF Pose of Kinect-2 with respect to Kinect-1, $Inliers_{optimized}$

$E_t = T_{RANSAC}$

For $\boldsymbol{k \leftarrow 1\ to\ 10\ do}$

- For $\boldsymbol{j \leftarrow 1\ to\ 10\ do}$
  - For $\boldsymbol{i \leftarrow 1\ to\ num\_inliers\ do}$
    - Find the Jacobian, $\boldsymbol{J}$
    - Add a single measurement
  - Compute weighted least squares
  - Compute mu ← Compute the weighted least squares set of parameter values by processing all the measurements
  - $E_{t+1} = \exp^{-mu} E_t$ ← Get non-linear optimized pose ($E_{optimized}$)
- For $\boldsymbol{i \leftarrow 1\ to\ N\ do}$
  - $P_{k1}^t = E_{t+1} P_{k1}$ ← Re-project feature correspondences
  - e_{ssd} ← Calculate SSD error using P_{k1}^t and P_{k2}
  - If $\boldsymbol{e_{ssd} < t}$ then
    - Increase $\boldsymbol{C_{score}}$ by $\left(\mathbf{1} - \frac{\boldsymbol{e_{ssd}}}{\boldsymbol{t}}\right)$
    - $Consensus_{set}++$ (Add P_{k1} and P_{k2} into consensus set)
- $Inliers_{optimized} = Consensus_{set}$ ← Obtain new homogeneous inliers set and feed into (k+1)th iteration

Return $E_{optimized}$, $Inliers_{optimized}$

The initial part of the algorithm *i.e.* computing key-points, ORB feature descriptors, applying brute-force descriptor matcher with the Hamming Norm and cross checking the correspondences were performed using ORB implementation available in the OpenCV library [59].

In order to obtain 3D world coordinates of the matched feature points, the intrinsic camera parameters were used. We calibrated Kinect cameras using Camera Calibration Toolbox for Matlab [63] as described below in Section 4.2. The obtained 3D point correspondences were fed into RANSAC algorithm to iterate over 500 times to estimate approximate transformation matrix considering an error threshold of 3cm between the re-projected and measured points.

When implementing our localization approach, we used Tom's Object-oriented Numeric Library (TooN) [64], a C++ numeric library designed to operate efficiently on matrices. We used TooN integrated with libCVD [65] which is a high-performance C++ library for computer vision and image processing.

4.2 Camera Calibration

Camera calibration is the process of estimating intrinsic and/or extrinsic parameters of a camera. Intrinsic camera parameters describe camera's internal characteristics such as focal length, image center, skew, distortion etc. Extrinsic parameters describe camera's position and orientation in the world. Camera parameters are used in estimating 3D world structure of a scene, applying corrections for lens distortion, determining location of the camera in the scene etc. These tasks are commonly applied in 3D computer vision [66], robotics, 3-D scene reconstruction [50] etc.

The input for camera calibration is 3D world points and their corresponding 2D image points. The most popular method to obtain these point correspondences is to use multiple images of a calibration pattern such as a chessboard or a checkerboard. We used Camera Calibration Toolbox for Matlab [63], a MATLAB implemented tool to obtain intrinsic and extrinsic parameters of cameras when given a set of images of a calibration pattern. We captured the images of a rectangular checkerboard pattern and fed into the Toolbox. After the four extreme corners of each checkerboard pattern are defined, the tool extracts the grid corners giving an option to re-extract if the user is unsatisfied with the distortion. After corner extraction, we used the tool to run main camera calibration procedure which is done in two steps. The first step computes a closed form solution for the calibration parameters ignoring any lens distortion. The second step runs non-linear optimization which minimizes the total re-projection error over all the calibration parameters. Then the grids were re-projected on the original images based on calculated intrinsic and extrinsic parameters. This process of computing grid corners, computing camera parameters and re-projection on original images were done several times until the re-projection error becomes minimum and converges to a certain value. The following Figure 4.1 shows a few sample images of the checkerboard captured using a Kinect sensor for calibration process.

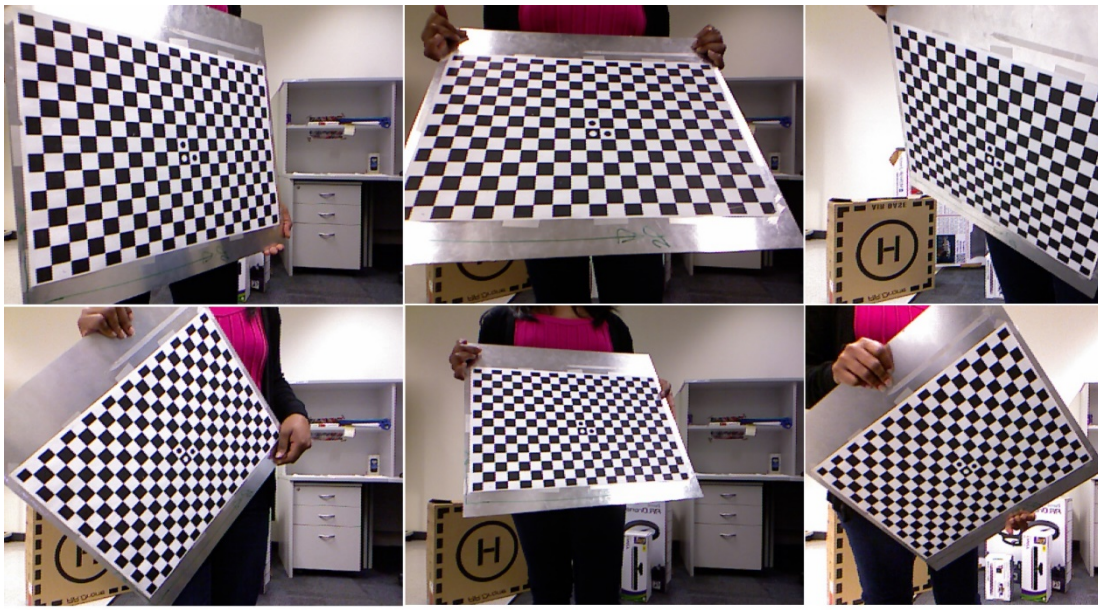


Figure 4.1: Sample Images of the Checkerboard used for Kinect camera calibration

4.3 Experiments

The proposed localization approach for multiple RGB-D sensors was analyzed and evaluated using different laboratory experiments. During all our experiments, Microsoft Kinect version1 [35] sensors were used to capture color images and depth data. The proposed localization approach was evaluated offline based on several sets of data collected from Kinect RGB-D sensors. One of the purposes of collecting these datasets was to investigate how the translation and rotational difference between the Kinects affect the estimated pose and how robustly our algorithm responds to these variations. These facts were investigated by capturing data using two Kinects while one sensor is moving with a translation and rotation relative to the other.

The experiments also help to examine how suitable our localization approach when used in swarm robot systems where the robots move relative to a stationary RGB-D sensor or to a static helper robot with and RGB-D sensor. The pose of one Kinect relative to the other was estimated using our proposed localization approach and evaluated against the ground truth measurements in each of the experiment scenarios.

To test the robustness of our localization algorithm against different scene conditions such as feature rich and feature less, two types of scenes as dense and sparse were considered while collecting data. This section explains these scenarios in more detail and how the experiments for data collection were carried out.

We compare this proposed localization approach with our previous RGB to RGB-D localization approach [67] that we presented in Australasian Conference in Robotics and Automation (ACRA) held in 2011. The purpose of this system is to estimate the pose of a mobile robot in an external stationary Kinect's coordinate frame. In this previous approach, the 6DoF pose of the smart phone robot is estimated by matching the appearance-based feature correspondences between the mobile phone camera and the Kinect. An overview of this robot localization process is illustrated in Figure 4.2.

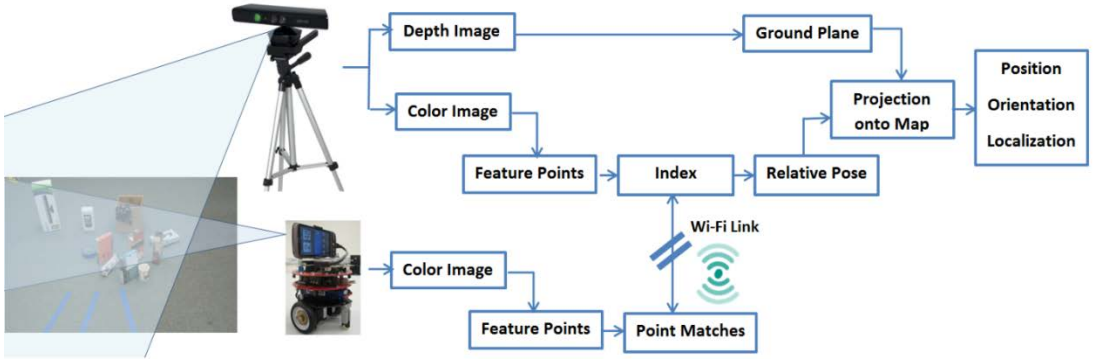


Figure 4.2: Overview of the ACRA System: Visual Localization between a Mobile Phone Camera and an External RGB-D Sensor

The process begins by detecting salient features and building feature descriptors in greyscale images from both the Kinect and smartphone cameras. FAST [54] features are computed from the greyscale images in both the Kinect-CPU server and the robot while the features are detected across a pyramid, 5-layers in the server and 2-layers on the robot. For each FAST corner, a HIPS [68] feature descriptor is calculated. The feature locations and descriptors built on the mobile phone are sent to a Kinect CPU server via a WiFi link. Sparse features are sent over the wireless ink to reduce the amount of transmitted data. For each feature, a 32-byte descriptor and 8 bytes of coordinates are sent. The server performs robust feature matching to provide a set of correspondences between 2D points from the robot camera and 3D points from the Kinect. The Lie algebra of rigid body motions is used to linearize the displacement of these salient features and to calculate a pose estimation from correspondences. This way, the robot is localized in 3D space from the Kinects perspective and the location is projected onto the ground plane to obtain the robots 2D position.

In the analysis of our proposed approach, we use this previous work to compare the accuracy of our new method. One of the reasons to use this previous work as a baseline is the methodology being very similar hence provides a good comparison of the results. Both these methods are trying to localize a moving robot/sensor in external RGB-D sensor's coordinate frame however, our approach uses RGB-D data from the moving element which introduces some interference to the scenario but enhancing the information with depth data while ACRA method provides interference-free 2D feature correspondences and locations from the moving element. Since both methods are trying to investigate a suitable localization approach for mobile robots to operate in stationary sensor's framework using different feature detection and matching (FAST, HIPS versus Oriented-BRIEF) and pose estimation algorithms the analysis of the results from two methods will give a good estimation of which method is more eligible.

Our previous approach (ACRA) uses a CPU server to process data (RGB-D) from the attached stationary Kinect and the data (feature descriptors) send by the smart phone over a wireless link. Therefore, when running our data sets in ACRA system, we applied the data set captured from two Kinects in each of the scenarios in place of the frames from Kinect and smart phone. When integrating our data set in the ACRA localization approach we used the following method.

1. The ACRA algorithm captures color and depth images directly from the attached Kinect sensor. Instead we forced the algorithm to obtain the saved depth and color frames from the stationary Kinect (Kinect-1) in our data set.

2. Instead of transferring feature locations and descriptors from the smart phone over the wireless link, we integrated the algorithm running on smart phone into the Kinect server application so that the whole system runs in one place eliminating the requirement of wireless link. Then the application was modified to retrieve the saved color images of the moving Kinect representing the color images from the smart phone.

Then the ACRA system was run for the data sets obtained for some of the scenarios and the analysis of results is given in Chapter 5.

4.3.1 Data Collection

We evaluate our proposed localization approach for RGB-D sensors based on offline data. Therefore, effort was taken to collect six different datasets for different scenarios as described below. For every positioning of Kinects described in below scenarios, two data sets were captured considering a dense and a sparse scene. The images captured for sparse scene has relatively lower number of feature matches however, to make the scene sparser when running the algorithm, we occluded a part of the view seen by one camera. This technique is explained using the images in Section 5.1.1.2. Our approach is based on key-point matches of both camera views; hence we assume that the RGB-D sensors on the robots or in the environment are pointing in a direction so that they share a part of their field of views. Therefore, the color and depth images are captured so that they observe an overlapped scene in both camera views. Figure 4.3 shows a sample scene in front of the Kinects in laboratory environment.

Figure 4.3: A sample scene in front of the Kinects

We used Microsoft Kinect for Windows SDK [69] version 1.8 to interface multiple Kinects on a single Windows 7 PC. The APIs available through Natural User Interface (NUI) were used to initialize the sensor array and obtain synchronous color and depth data at 640 x 480 resolution from two Kinects connected to the same PC.

In each of the three experiments described below, the captured six data sets consist of the following.

I. Raw data: RGB image, depth image, accelerometer readings
II. Ground Truth: Manually measured 6 Degree of Freedom (x, y, z, 3 axis rotations) pose measurements of RGBD sensors

To improve accuracy of the measurements, we removed the outer casing of the Kinects and used only the inner structure in our experiments as shown in Figure 4.4. Also, two wheels made of Perspex was attached to the Kinects so that they become easy to slide along the metal bar during the experiments.

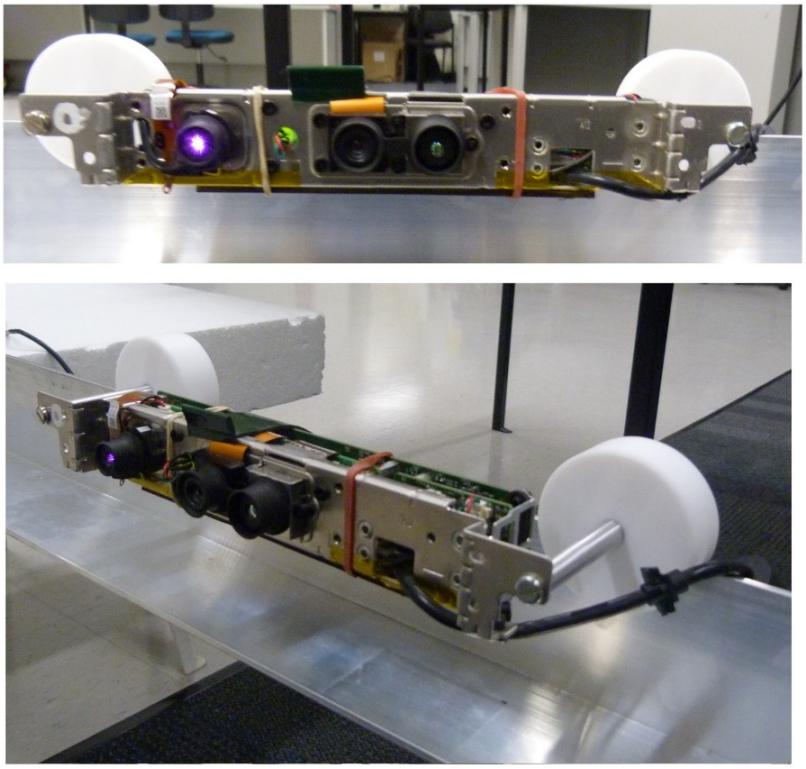


Figure 4.4: Kinects with the outer casing removed. Two wheels made of Perspex are attached to hold the Kinects on a metal bar

4.3.1.1 Experiment-1: Systematic Translation only of one sensor relative to a stationary sensor

The first experiment investigates impact of proposed localization approach on RGB-D sensors those are moving only with a translation relative to a static sensor. Data was collected using two Kinect sensors while one Kinect was rigidly fixed, and the other Kinect was systematically placed at different locations so that they only maintain an offset in translation. In the experiment setup, a rigid long metal bar was used to guide the moving sensor so that there is no rotational difference between the two sensors. The bar was marked with distance measurements relative to the stationary sensor to easily obtain the ground truth measurements while capturing Kinect data at different locations. The scene at which the Kinects were facing was assumed to be static during the time of capturing data. Figure 4.5 shows the experimental setup used to collect data in the research lab.

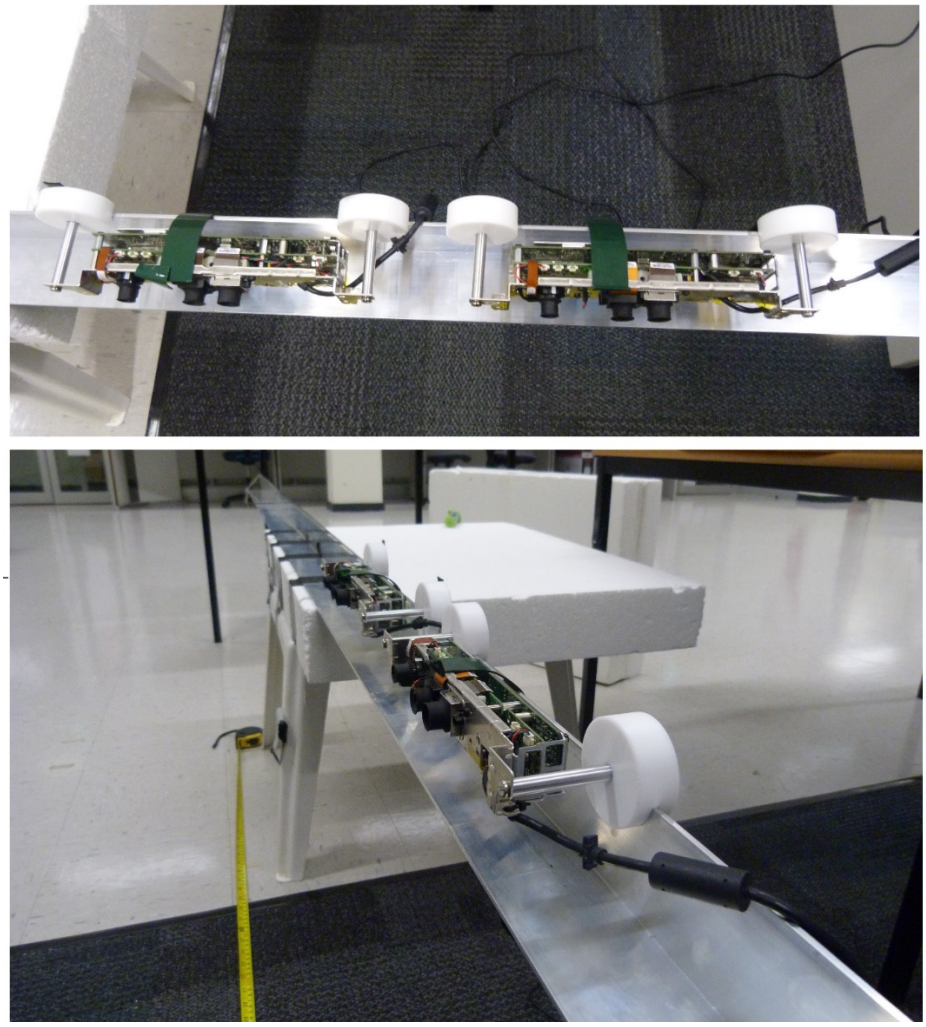
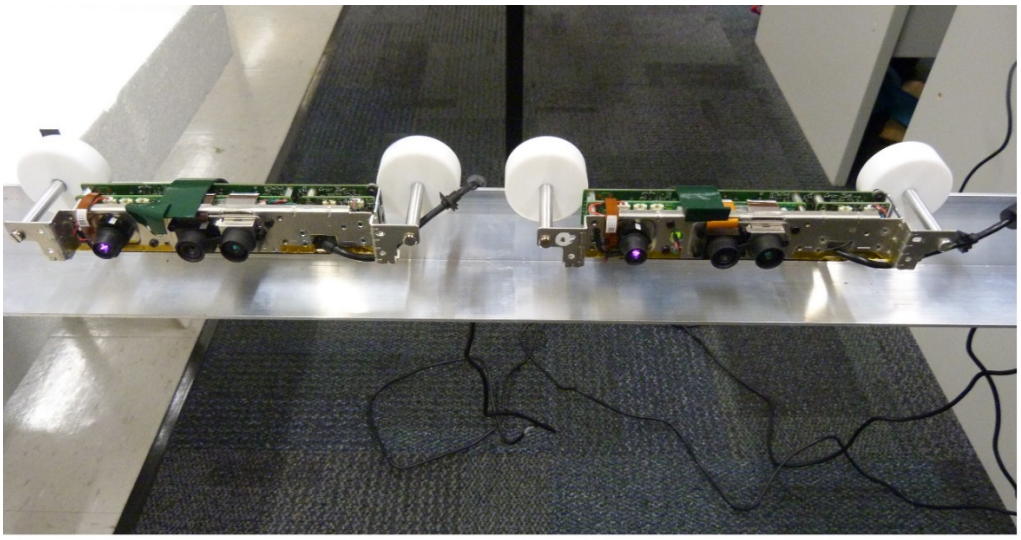


Figure 4.5: Two Kinects attached to the metal bar using the cut-through slots on the wheels so that the Kinects can slide along the bar

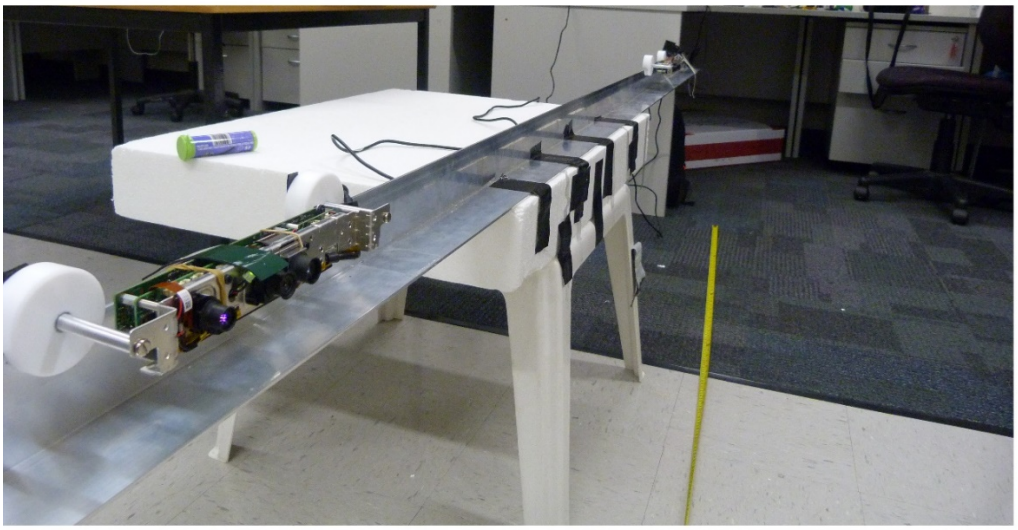
The Figure 4.6 shows sample scenes observed by two Kinects while they are placed at a certain displacement in translation.

Figure 4.6: The color images obtained from two Kinects while they are 0.5m apart.

For this experiment, capturing Kinect data was started when the Kinects were 30cm away from each other and repeatedly captured at locations with 10cm increments while moving the second Kinect away from the static Kinect as shown in Figure 4.7. This way, the ground truth measurements were easily recorded for each Kinect position. Instead of a single data frame, we captured 10 data frames from both Kinects at each position so that the average pose can be estimated over 10 frame pairs. The maximum allowed translation between two Kinects were 1.5m for the dense scene and 1.7m for the sparse scene which are limited by the number and the quality of features available in overlapping area.



(a)



(b)

(c)

Figure 4.7: (a)Starting position of two Kinects (b)One Kinects is moved away from the static Kinect (c)Kinects are aligned to each other without angular difference

4.3.1.2 Experiment-2: Systematic Translation and Rotation of one sensor relative to a stationary sensor

The purpose of this experiment was to examine the effect of the proposed work on RGB-D sensors those are moving with a rotation and translation relative to a static sensor. The experiment is applicable to a scenario of moving an RGB-D sensor along X- and Z-axes while rotating about Y-axis with respect to a static RGB-D sensor. In this experiment, data was collected by systematically moving one Kinect sensor while maintaining a certain translation and rotation relative to a static Kinect sensor. We used a turn table to obtain accurate ground truth translation and rotation while moving the Kinect sensor. The Figure 4.8 shows the experimental setup with the turn table used in data collection.

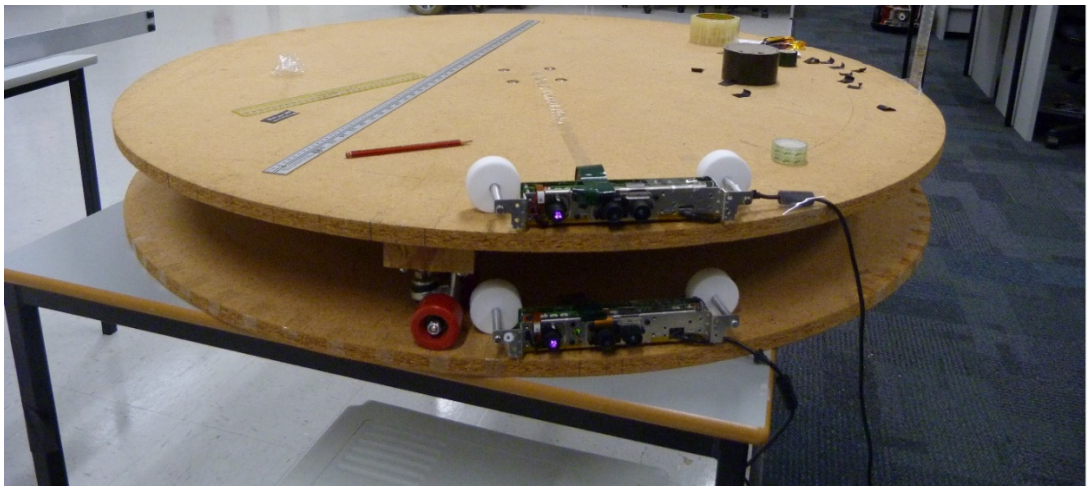


Figure 4.8: Experimental setup with the Kinects placed on the turn table.

The static Kinect was rigidly attached on to the bottom plane of the turn table which is not turning. Since the top layer is easily rotatable, we attached the second Kinect on to the top layer. This way we achieved the translational and rotational difference between the two Kinects without moving the Kinects themselves but using the turn table. This technique avoided any misalignments and human errors occur when moving the Kinects manually which affect the accuracy of ground truth information.

Figure 4.9 shows how the Kinect sensors are attached to the top and bottom layers of the turn table.

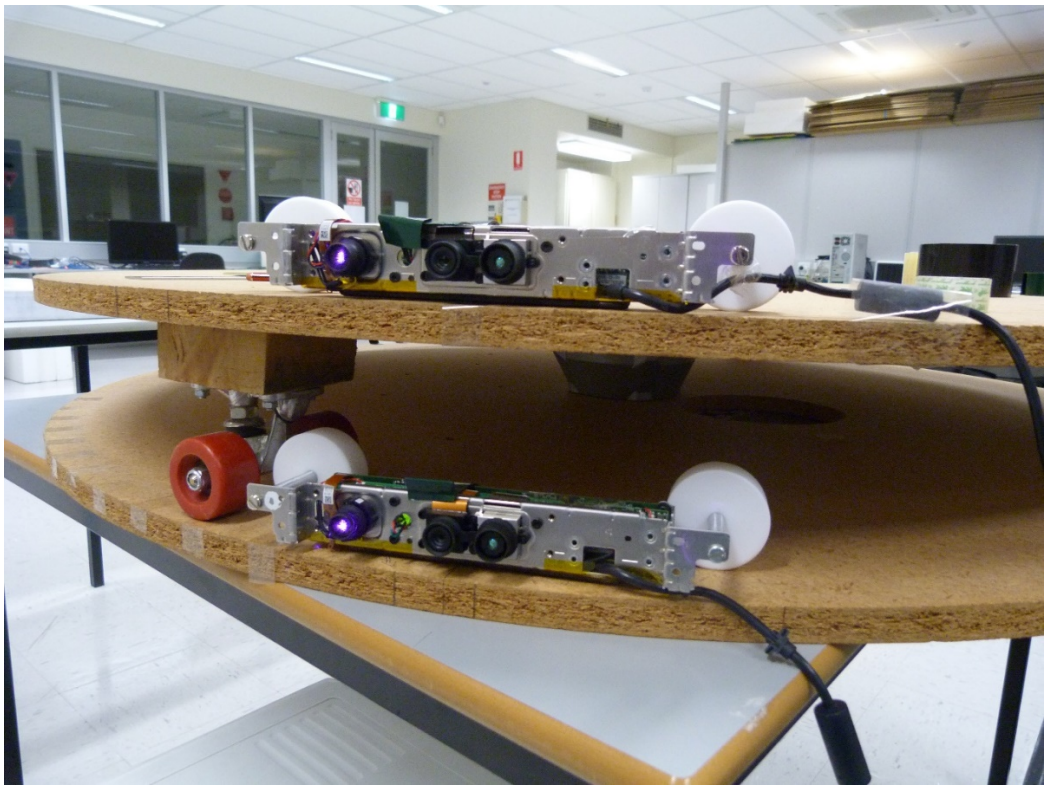


Figure 4.9: Static Kinect is attached to the bottom plane with markers for the angular displacement.

The top plate was rotated 5 degrees at each step while capturing the Kinect data. Markers were placed along the bottom plate circumference at the arc lengths correspond to 5 degrees. The center of the wheels attached to the top plate was taken as the reference when measuring the corresponding distance along the plate circumference. Because the distance between the two plates is always a fixed value, the ground truth translation along Y axis is not varying between the two Kinects. The gap between two plates were 12.5 cm, hence the ground truth translation along Y axis is always limited to this value. The following Figure 4.10 shows how we obtained corresponding ground truth translation along X and Z axis when the turn table is rotated at an angle of θ.

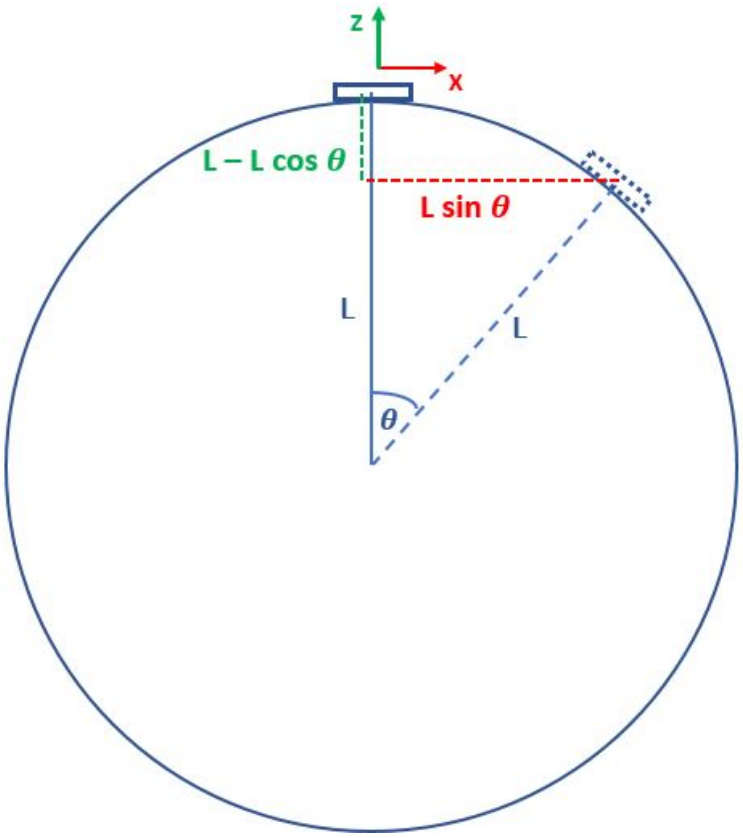


Figure 4.10: Rotation and Translation Estimated using the Turn Table

When the top plate is rotated θ degrees with respect to the fixed Kinect, then the corresponding translation along X and Z axes are L sin θ and (L – L cos θ) respectively where L is the distance from the center of turn table to the camera center. In our experiments, we maintained a distance of 0.6m between the centers of the turn table and Kinect color camera.

The Figure 4.11 shows when the two Kinects are placed with a translation and rotation. Similar to Experiment-1: Systematic Translation only of one sensor relative to a stationary sensor, 10 data frames were captured at each 5 degrees angular offsets.

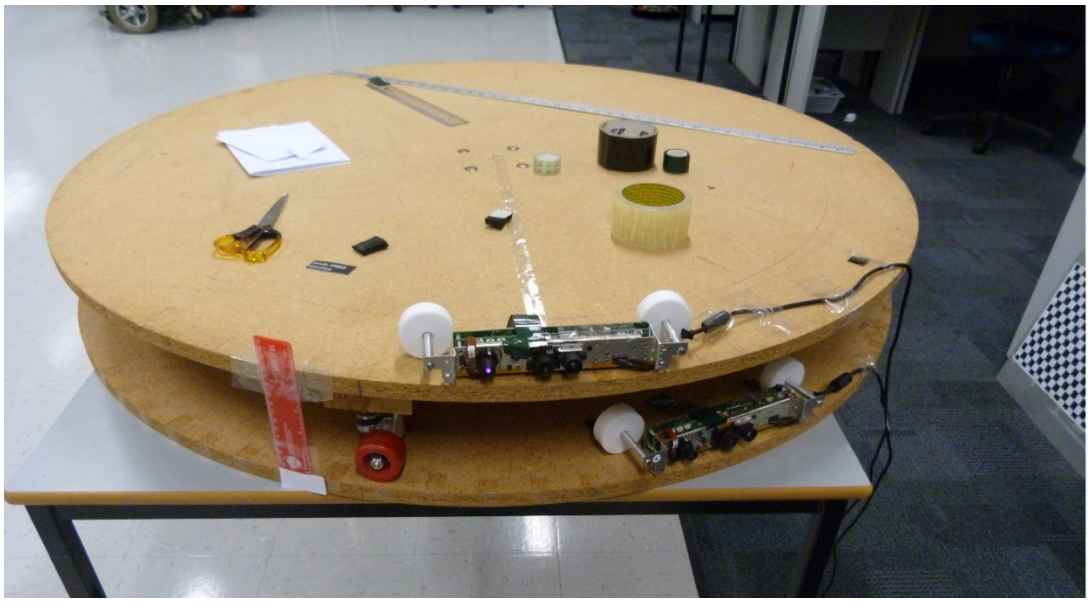


Figure 4.11: The Kinect on the top plate is rotated relative to the static Kinect on the bottom plate

The following Table 4.1 shows the obtained ground truth values corresponding to each measurement step for this experiment.

Angle in Degrees (θ)	Distance in meters from camera center to turn table center (L)	Translation along X Axis in meters (L sin θ)	Translation along Y Axis in meters (Fixed)	Translation along Z Axis in meters (L – L cos θ)
5	0.6	0.052293	0.125	0.002283
10	0.6	0.104189	0.125	0.009115
15	0.6	0.155291	0.125	0.020445
20	0.6	0.205212	0.125	0.036184

25	0.6	0.253571	0.125	0.056215
30	0.6	0.300000	0.125	0.080385
35	0.6	0.344146	0.125	0.108509
40	0.6	0.385673	0.125	0.140373
45	0.6	0.424264	0.125	0.175736
50	0.6	0.459627	0.125	0.214327

Table 4.1: Ground Truth Translations at different Angles

When the top plate is rotated θ degrees with respect to the fixed Kinect, the moved Kinect only rotates about Y axis and the ground truth rotation between two Kinects about X and Z axes are zero. The ground truth rotation matrix when the two Kinects are rotated at θ angle about Y axis at each step is derived using the general rotation matrix corresponding to Euler angles $\emptyset$, θ, φ about X, Y and Z axes respectively.

$$R = \begin{bmatrix} cos\theta\ cos\varphi & sin\emptyset\ sin\theta\ cos\varphi - cos\emptyset\ sin\varphi & sin\emptyset\ sin\varphi + cos\emptyset\ cos\varphi\ s \\ cos\theta\ sin\varphi & cos\emptyset\ cos\varphi + sin\emptyset\ sin\theta\ sin\varphi & cos\emptyset\ sin\theta\ sin\varphi - cos\varphi\ s \\ -sin\theta & sin\emptyset\ cos\theta & cos\emptyset\ cos\theta \end{bmatrix} \quad (31)$$

When the rotation about X and Z axes are zero *i.e.* $\emptyset = 0$, $\varphi = 0$ then the ground truth rotation matrix, R_{gnd} can be derived as below.

$$R_{gnd} = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \quad (32)$$

4.3.1.3 Experiment-3: Freehand movement of one sensor relative to a stationary sensor

With freehand moving experiments of an RGB-D sensor relative to a static similar sensor, we investigate the suitability of proposed approach in 6-DOF localization applications. The experiment described here, and the obtained results given in Section 5.1.3 helps to evaluate the eligibility of suggested work in localizing indoor robots, UAVs, drones or 6-DOF localization of hand-held RGB-D sensors which are useful in Augmented Reality (AR) applications.

In this experiment, data was captured while one Kinect was moving freely with 6DOF transformation relative to a fixed Kinect in the environment. Therefore, both relative translation and rotation of Kinects changed during the data capture. However, obtaining ground truth trajectory of moving Kinect was a challenging task in this experiment. Therefore, the Kinect was moved in known patterns/trajectory and returned to the same starting position. As an example, the front rectangular surface of a box placed in front of the scene was assumed to be a trajectory of the Kinect. The relative rotation between the Kinects were assumed to be fixed during this experiment, i.e. the Kinect was moved so that there is no rotational difference between the Kinects. The following Figure 4.12 shows a diagram of the experimental setup.

However, this experiment is prone to significant human errors because there can be drifts and slight rotations while manually moving the Kinect along the edges of the box. Eventhough the ground truth is assumed to be a perfect

rectangle with sharp edges, the Kinect's true trajectory could be slightly different. We have given the estimated trajectory of this experiment in Section 5.1.3.

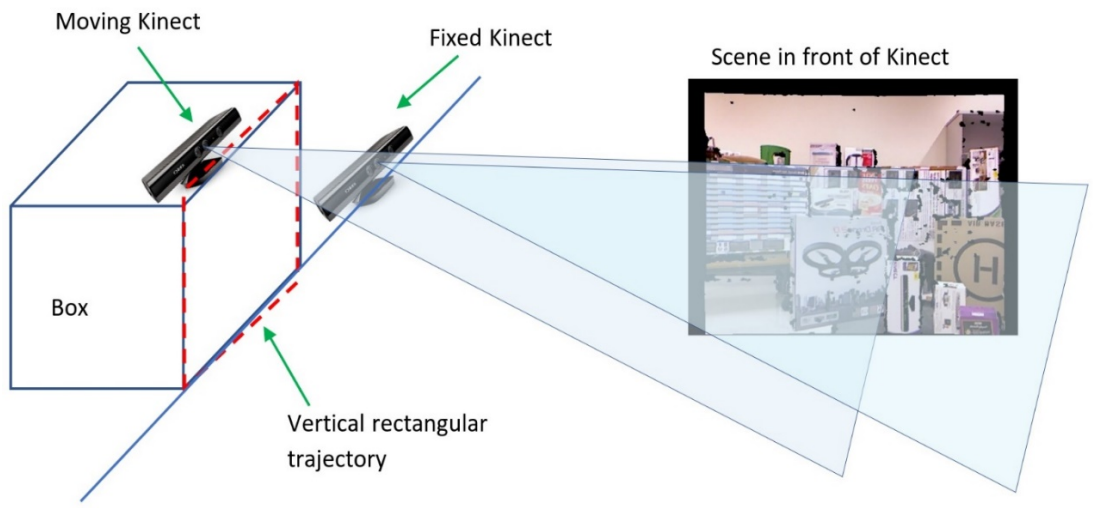


Figure 4.12: Setup for Experiment-3

Precise ground truth information is important for qualitative and quantitative analysis of this experiment data due to being highly inclined to human errors. Therefore, the attention was drawn towards evaluating this experiment scenario with publicly available RGB-D datasets. In the recent research, there are only a few efforts taken to produce or experiment with multiple Kinects. RGB-D People dataset [70] was captured to evaluate people detection and tracking algorithms for robotics and interactive systems. The data was collected synchronously in an indoor environment using three combined static Kinects with a joint field of view. Human fall detection dataset [71] was captured by Bogdan et al using two static Kinects in an indoor room to detect and recognize human falls when integrated with corresponding accelerometer data. Big BIRD dataset [72] collected by the researchers in University of California Berkeley and the dataset collected by

Susanto et al [73] address object detection and recognition using multiple Kinects to accelerate the developments in computer vision and robotic perception. Even though these datasets are captured by multiple Kinect sensors their application on Experiment 4.3.1.3 is unfitting due to the Kinects being mostly static.

On the other hand, there are several publicly available single Kinect RGB-D benchmark data sets to evaluate visual odometry and SLAM systems. As discussed in Chapter 0, ICL-NUIM dataset introduced in [46] is being used to evaluate RGB-D visual odometry, 3D reconstruction and SLAM algorithms. The dataset consists of handheld RGB-D camera sequences within synthetically generated office room and a living room where the latter consists of associated 3D polygonal model in addition to trajectory data which allows evaluation of the accuracy of the final reconstruction.

In this analysis we use the TUM Benchmark dataset, for evaluating RGB-D SLAM systems published by Sturm et al [43]. The dataset consists of image sequences from a single Kinect with highly accurate and time-synchronized ground truth camera poses generated using a motion capture system. This benchmark dataset can also be used to evaluate visual odometry systems on RGB-D data. The dataset consists of 39 sequences recorded using a hand-held Kinect in two different environments. The first sets of trajectories are recorded in a typical office environment, “fr1” with around 6m x 6m area. The second set of trajectories are recorded in a large industrial hall, “fr2” with about 10m x 12m in size. The average, camera speed of considered fr1 datasets are faster than fr2 datasets. This can be clearly shown in the color images of fr1 datasets having significant motion blur.

In our analysis we use several fr2 data sequences from the categories “Testing and Debugging” and “Handheld SLAM”. Because the dataset consists of image sequences captured from a single Kinect, we use the first data point in the dataset as from the fixed Kinect and the remaining data as from the moving Kinect. We used “xyz” data sequences captured by moving the camera approximately along X-, Y- and Z- axis with little rotational components. Analyzing our algorithm on this dataset will prove its accuracy when moving 6-DOF. The analysis in Section 5.1.3 will also show the quality of estimated trajectory when using the Hand-held SLAM fr2/desk dataset. In fr2/desk sequence the images are taken in a static office environment consisting two tables with various accessories including keyboard, monitor, books etc. The ground truth trajectory for this dataset is about 18m long with average translational velocity of 0.193m/s and average angular velocity 6.338deg/s. Since fr2 sequences have been captured relatively slower consists of images with less motion blur, the evaluation of proposed approach against these sequences demonstrate the best-case scenario.

To evaluate against worst case scenario, we use several fr1 data sequences from the same categories. Camera motion in fr1/xyz sequence is similar to fr2/xyz sequence but with a different environment. For this sequence, the Kinect was pointed at a typical desk in an office environment. This sequence contains only translatory motions along the principal axes of the Kinect, while the orientation was kept (mostly) fixed. On the other hand, fr1/desk sequence contains several sweeps over four desks in a typical office environment. In our analysis we use these two sequences from fr1. The average translational speed of the camera in fr1/desk sequence is about 0.413m/s while the angular velocity is around 23.327

deg/s which is very faster compared to 0.193m/s and 6.338 deg/s in fr2/desk sequence. Compared to fr2 sequences we analyzed, fr1 sequences are relatively fast and consists of images with motion blur, hence these sequences are used to demonstrate the usage of proposed work in worst case scenario.

When evaluating against these data sequences, we assume a particular Kinect frame as the reference frame obtained from the static Kinect. The pose of sub sequence frames is estimated relative to this reference frame. Since our algorithm is very much dependent on the number of good feature matches extracted in the overlapping views, it is vital to have a reasonable overlapping area between the two Kinect frames being matched. However, data collected from a moving hand-held Kinect might not always satisfy this requirement. Referring to RGB images of most of these benchmark sequences, there are many image pairs with no common views at all. Therefore, considering only the first Kinect frame as the reference is not suitable in this analysis. Due to this reason, when evaluating our algorithm on TUM data sequences we reinitiate the Kinect-1 reference frame at a certain point along the trajectory. This way we avoid losing feature matches in both Kinect views. This strategy is explained in the Proof of Concept section in more details.

5 Results and Analysis

The proposed localization approach was evaluated in two ways, using the acquired data in laboratory environment and also using publicly available datasets. The data collection was conducted as explained in Section 4.3.1. For all the scenarios described in this section we investigated the behavior of the proposed work against ground truth data obtained at each of the experiments. Evaluation using collected datasets is twofold. We evaluate the proposed approach using collected data and we use the same data set on the previous ACRA work [67] described in Section 4.3 to compare the accuracy.

The implemented proposed work was run offline on collected different data sets for each scenario. The same datasets collected for Scenario-1 and Scenario-2 also used to evaluate against ACRA method. When analyzed on ACRA system, the stationary Kinect data was used as the single Kinect used in ACRA method and the color images from moving Kinect was used in place of the RGB color images provided by the smart phone. The following sections provide a detailed analysis of the three scenarios with the results obtained from all the experiments to evaluate its accuracy and robustness to apply in realistic applications.

5.1.1 Scenario-1: Systematic Translation only of one sensor relative to a stationary sensor

In this scenario, we analyzed the datasets collected for both dense and sparse scenes. The Kinects were initially placed with 30cm displacement and then the translational distance was increased by 10cm at each step until there is no overlap

between their FOVs. The sensors start to observe a scene with a displacement >= 30cm in X direction and zero displacement in Y and Z directions.

5.1.1.1 Scenario-1 Dense Scene

In our experiments, we used dense and sparse scenes where dense scene consisted of relatively higher number of feature matches than the sparse scene. The following Figure 5.1 shows some sample dense scene image pairs captured from both Kinects together with matched ORB feature correspondences. In each image pair, the images from the stationary Kinect is shown on the right-hand side. These four image pairs are randomly picked from each ten frame pairs captured when the Kinects are 30cm, 50cm, 70cm and 90cm apart.

Figure 5.1: Initial matches of dense scene for Scenario-1 when the Kinects are 30cm, 50cm, 70cm and 90cm apart

These matches are then filtered by RANSAC algorithm and non-linear optimization algorithm described above and the following Figure 5.2 shows the remaining final inlier matches for above frames after optimizing.

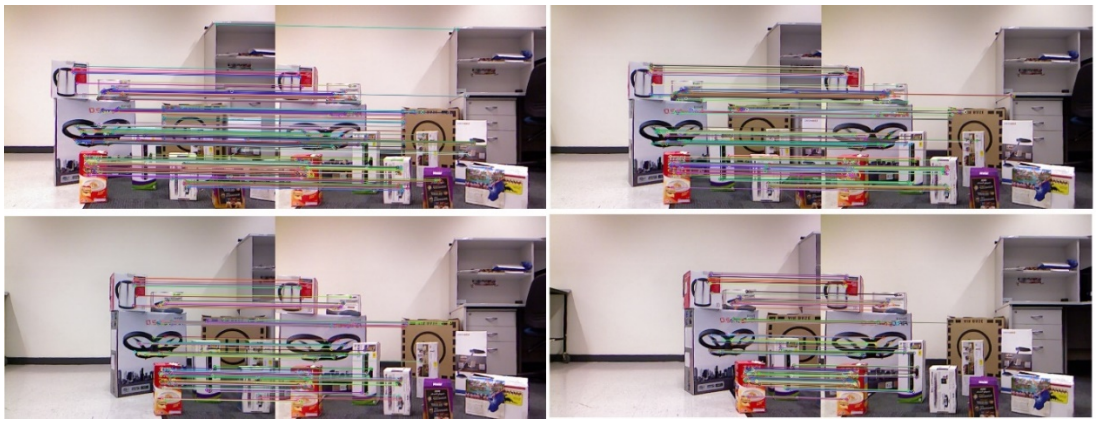


Figure 5.2: Optimized inlier matches of dense scenes for Scenario 1

From top-left, top-right, bottom-left, bottom-right order the frame pairs are correspondent to Kinects being apart 30cm, 50cm, 70cm and 90cm respectively

For each frame pair, we calculate the Euclidian translation error between the ground truth translation vector T_{gnd} and optimized translation vector T_{opt} as below.

$$T_{error} = \sqrt{\sum_{i=1}^{3} \left(T_{gnd}[i] - T_{opt}[i]\right)^2} \tag{33}$$

When calculating rotational error, we consider the ground truth rotation as, $R_{gnd} = I$ because the Kinects are placed only with a difference in translation. The

angular error is calculated by taking the norm of the vector form of difference matrix between optimized rotation, R_{opt} and R_{gnd}. Therefore, the optimized rotation simply becomes the difference rotation between the ground truth rotation and the optimized rotation. The vector form of this difference rotation is the vector of Euler angles along x, y, and z axes. Then rotational error, R_{error} is estimated taking the norm of difference vector.

$$R_{error} = l^2 norm\ (vec(R_{opt}^T * I)) \tag{34}$$

- **Comparison against Ground Truth**

The error in translation and rotation compared to ground truth measurements for the frame pairs captured for dense scene is shown in the below graphs. The horizontal axis represents the ground truth distance between the Kinects ranging from 0.3m – 1.5m. The error value represented by each point is calculated as a mean error by matching 10 frame pairs.

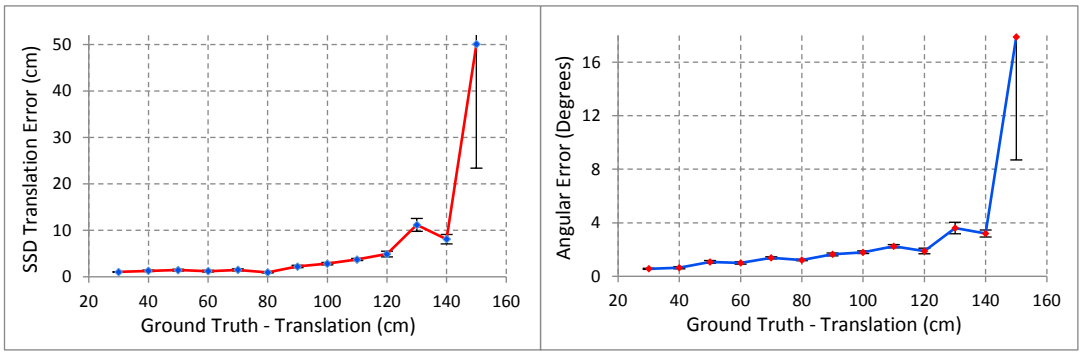


Figure 5.3: Estimated Euclidian Translation Error and Rotation Error for a dense scene when the sensors are systematically positioned with only a translation of range from 30 – 150 cm

The translation error is lies almost within 10cm and angular error lies within 4^0 until the Kinects are 1.4m apart. However, when they are apart more than 1.4m the error increases highly due to the fewer number of available feature matches between the frames. Therefore, above data shows 96.5% average translational accuracy and 98% per centimeter average rotational accuracy over 1.4m span.

The following Figure 5.4 shows the variation of percentage inlier feature matches with the translation and rotation for the dense scene. Inlier percentage was calculated as,

$$Inlier\ \% = \left(\frac{Number\ of\ optimized\ inlier\ matches}{Number\ of\ total\ matches}\right) X\ 100\% \qquad (35)$$

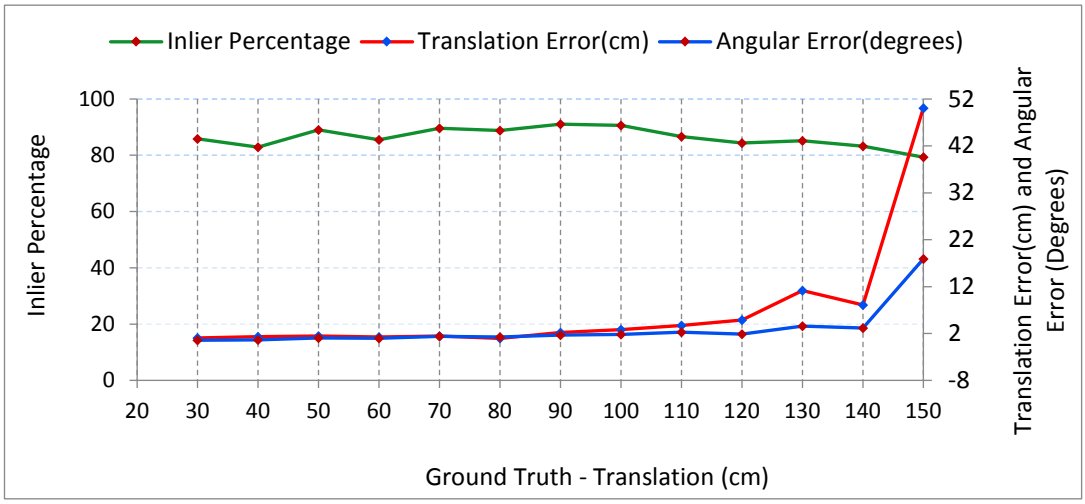


Figure 5.4: Change of inlier percentage with respect to translation and angular error for a dense scene

The inlier percentage remains above 80% except when Kinects are more than 1.4m away, however the accuracy of the estimated pose after this point severely depends on the availability of number of good inliers.

- **Comparison against ACRA Work**

As described above in Section 4.3, we used this data set to run on our previous localization approach submitted to ACRA. The following figures show the inlier matches for the same frame pairs shown above but running on ACRA system. In each frame pair, the left image shows the moving Kinect image while the right image is considered to be the stationary Kinect.

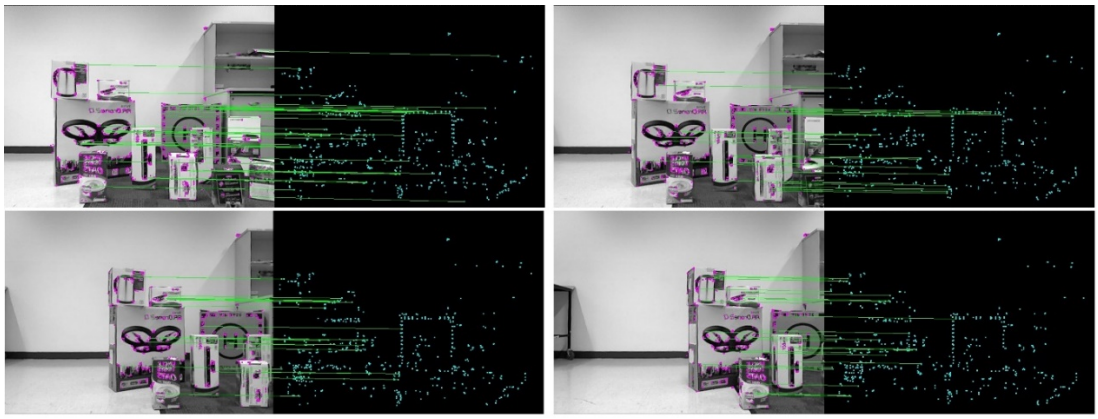


Figure 5.5: Inlier matches obtained from ACRA method for Scenario-1 dense scene

From top-left, top-right, bottom-left, bottom-right order the frame pairs are correspondent to Kinects being apart 30cm, 50cm, 70cm and 90cm respectively.

The following graphs show the comparison of Euclidian translation error and rotation error for both approaches relative to ground truth values.

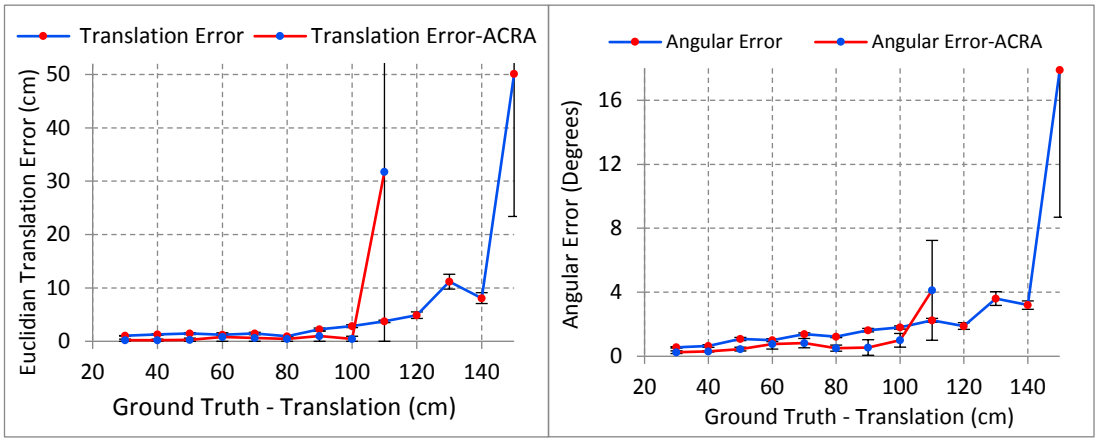


Figure 5.6: Translation and rotation error of estimated pose using proposed localization approach compared to previous (ACRA) approach for the dense scene in scenario-1

According to the results, even though the ACRA localization approach gives slightly accurate results, it is not robust as our new approach for the translations longer than 1m. The ACRA approach fails to work for data obtained by placing Kinects more than 1.1m apart. The proposed approach works well for translations up to 1.4m with an average Euclidian error of 3.5cm. Therefore, above data shows 97.4% average translation accuracy with our method and 96.1% accuracy with ACRA method for 1.1m span. The corresponding per centimeter rotational accuracy is 98.2% from this method and 98.8% from ACRA method. The proposed approach enhances the translational operating range by 27% over ACRA method.

The following graph shows the inlier percentage of both approaches for the translation up to 1.5m.

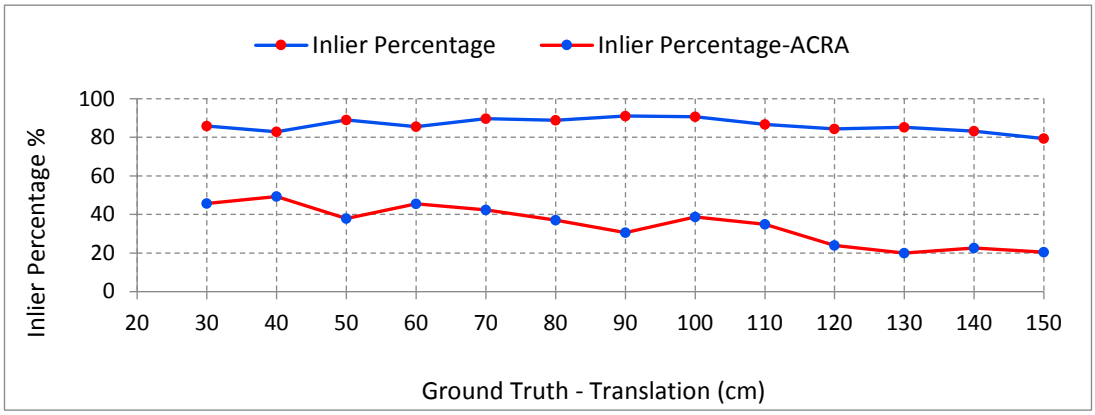


Figure 5.7: The Inlier percentage of both approaches for ground truth translation up to 150cm

The proposed approach filters higher number of feature matches as inliers while the ACRA approach performs by filtering almost half of the inliers than that. However, when the Kinects are more than 1m apart, the ACRA work fails to respond even if the percentage of inliers drops only by 20%.

5.1.1.2 Scenario-1 Sparse Scene

In order to test the robustness of our approach against the number of feature matches between the Kinect images, we considered the same scenario with reduced feature environment. We captured data for the sparse scene which has 10% less number of feature matches than the dense scene. In order to make the scene sparser, we covered a part of the scene by removing the depth data while running the proposed approach. As an example, the following Figure 5.8 shows the initial matches between the frame pairs when the Kinects are at 30cm, 50cm, 70cm and 90cm apart.

As seen in the images, a part in the right-hand side image is occluded by removing depth data which gives no feature correspondences for that part of the image pair. This technique allowed to make the scene sparser by removing the

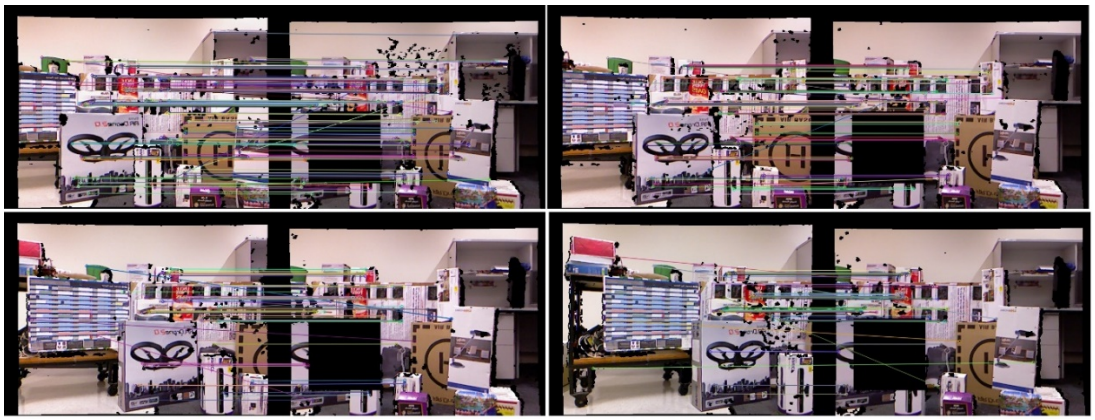


Figure 5.8: Initial Matches of Sparse Scenes for Scenario-1

From top-left, top-right, bottom-left, bottom-right order the frame pairs are correspondent to Kinects being apart 30cm, 50cm, 70cm and 90cm respectively.

feature matches. The following table shows the average number of feature matches in our dense dataset and sparse dataset for the Scenario-1. This average was estimated considering 10 frame pairs at each ground truth location.

Ground Truth Distance Between Kinects (cm)	Average Number of Feature Matches in the Dense Data Set	Average Number of Feature Matches in the Sparse Data Set
30	373.9	209.4
40	298.2	154.4
50	272.5	130.6
60	275.1	113.4

70	220.2	91
80	186.4	85.9
90	176.1	73.2
100	152.9	69.2
110	127.4	58.1
120	82.1	57.3
130	44.6	48.9
140	26.1	42.5
150	13.4	24.2

Table 5.1: Average number of feature matches for dense and sparse scenes for Scenario-1

The following images in Figure 5.9 show the filtered inlier feature matches for the above same sparse featured frame pairs to obtain the optimized pose. When comparing two figures, it can be clearly seen that outlier matches presented in Figure 5.8 has been removed in Figure 5.9.

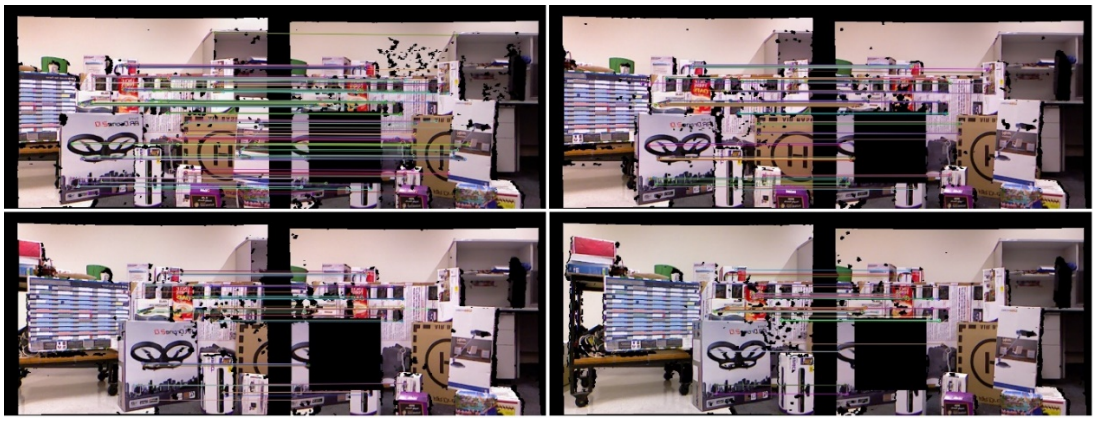


Figure 5.9: Filtered Matches of Sparse Scenes for Scenario-1

From top-left, top-right, bottom-left, bottom-right order the frame pairs are correspondent to Kinects being apart 30cm, 50cm, 70cm and 90cm respectively.

- **Comparison against Ground Truth**

The following graphs show the estimated translation and rotation errors compared to ground truth values and compared to the corresponding dense scene results.

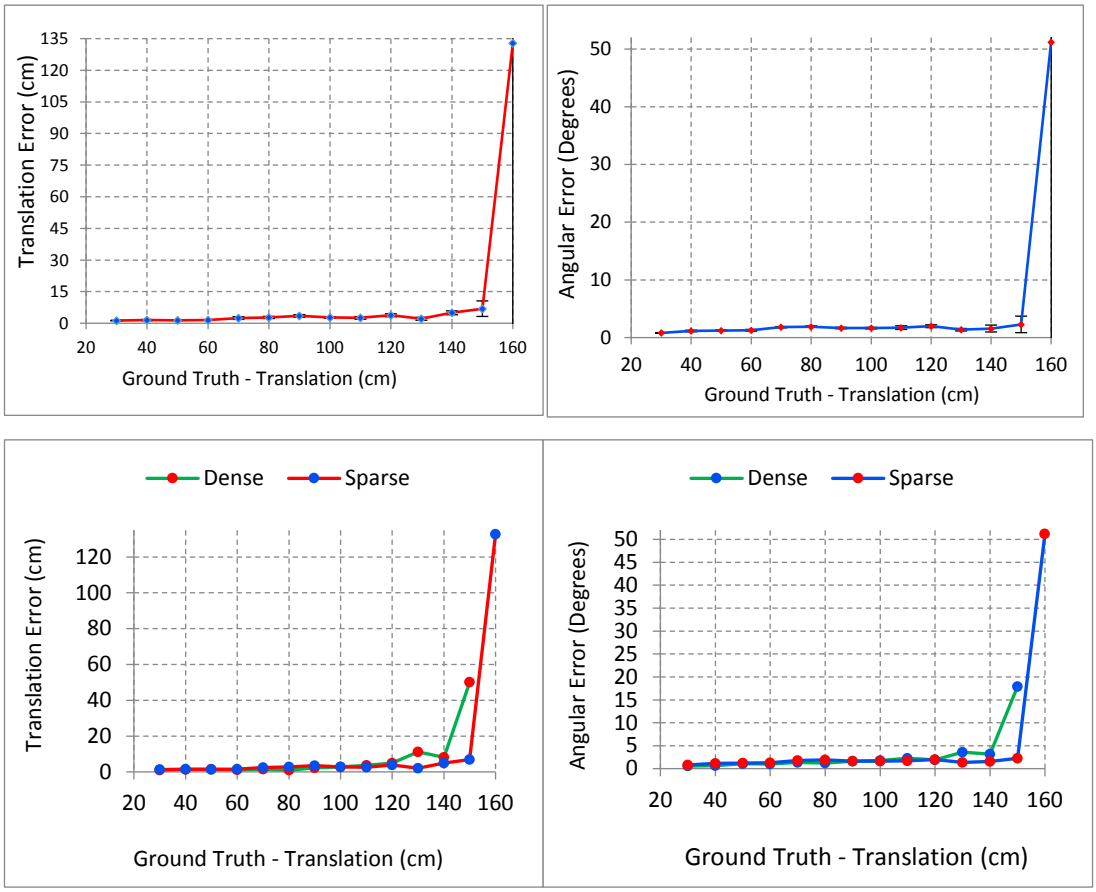


Figure 5.10: Translation and rotation error for dense and sparse scenes for Scenario-1

Above results demonstrate that our approach works well with the sparse scenes giving very similar or even better results compared to dense scenes. The localization accuracy up to 1.5m translation is very small having less than 3cm average error. Above data shows average translational accuracy of 96.7% and per centimeter average rotational accuracy of 98% over 1.5m span. Hence, it proves that our approach is robust against the number of inliers and does not dramatically affect the accuracy of the estimated pose. The inlier percentage of

both dense and sparse scenes for the above experiment is given in Figure 5.11 below.

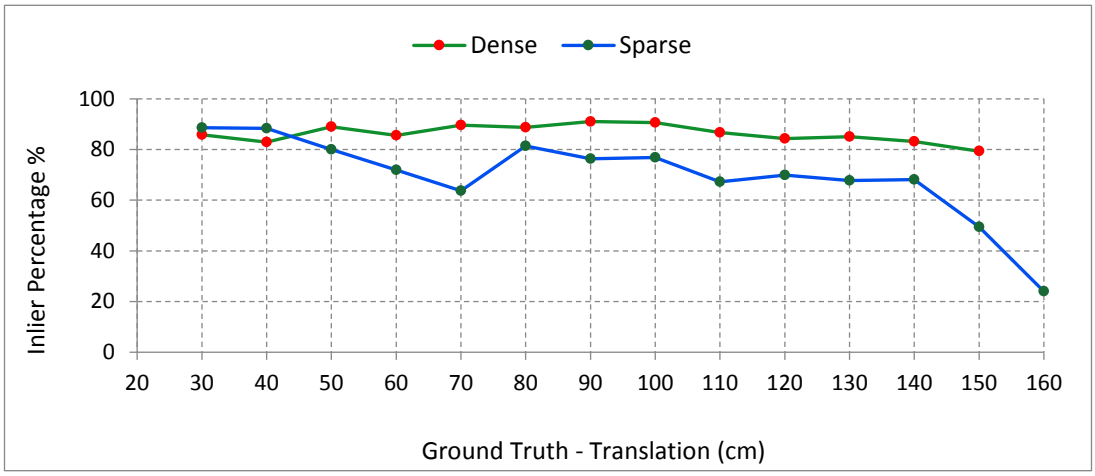


Figure 5.11: Inlier Percentage for both Dense and Sparse Scenes

Acording to the graph, the inlier percentage has dropped from 90% to 25% for the sparse scene while it is not much changed for the dense scene. However our algorithm has performed well for the sparse scene even if the inlier percentage has intensely dropped.

- **Comparison against ACRA Work**

We used the same set of sparse scene data for scenario-1 to run on ACRA localization approach. The following images in Figure 5.12 show the inlier feature matches obtained by ACRA algorithm for the same set of image pairs shown above.

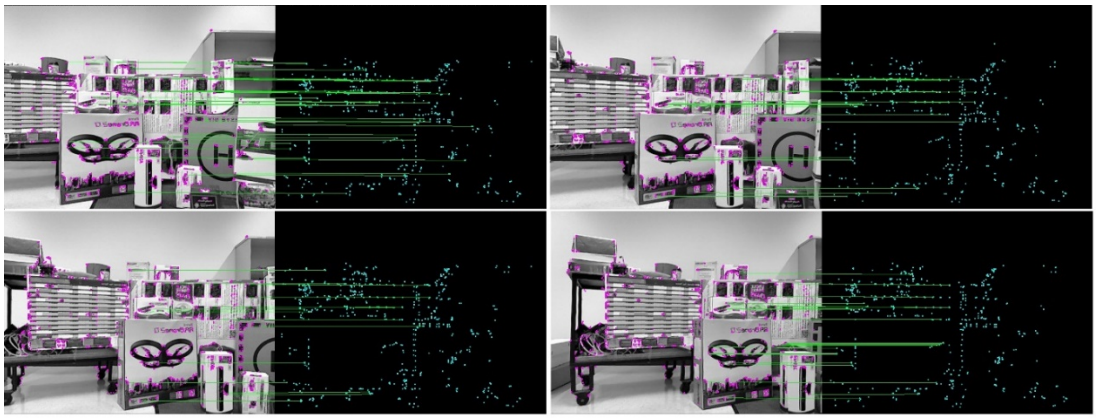


Figure 5.12: Filtered Matches using ACRA method for Sparse Scene in Scenario-1

From top-left, top-right, bottom-left, bottom-right order the frame pairs are correspondent to Kinects being apart 30cm, 50cm, 70cm and 90cm respectively.

The following graphs show how the ACRA method responded to the sparse scene in Scenario-1. The graphs below show the comparison of translation and rotational errors for the same sparse scene using both approaches. Also, the bottom graph shows the variation of inlier percentage for both approaches.

Figure 5.13: Translation and Rotation Error, Inlier percentage for proposed approach and ACRA method for sparse scenes in Scenario-1

According to these results, similar to the dense scene, the ACRA algorithm could perform only up to 1.1m to estimate the pose for the sparse scene. The proposed method produces 95.5% average translational accuracy while the ACRA method provides 98% corresponding accuracy for 1.1m span. Per centimeter average rotational accuracy achieved with our method is 97.8% while it is 98% with ACRA method for 1.1m span. However, in the sparse scenario our method enhances the operating range by 36%. The bottom graph shows that inlier percentage lies below 30% throughout the estimation.

5.1.2 Scenario-2: Systematic Translation and Rotation of one sensor relative to a stationary sensor

In this scenario, the Kinects were placed on a turn table initially with only a ground truth translation of 0.125m along Y- axis. Then, one Kinect was rotated in steps of 5 degrees angle and ten frame pairs were captured at each step for both dense and sparse scenes. The corresponding translation in X and Z axes are given in Table 4.1. The ground truth rotation matrix R_{gnd} at each step was determined as explained in Section 4.3.1.2.

5.1.2.1 Scenario - 2 Dense Scene

The following Figure 5.14 shows some sample dense scene image pairs captured from both Kinects with the matched ORB feature correspondences for this scenario. In each image pair, the images from the stationary Kinect is shown on the right-hand side. These four image pairs are randomly picked from each ten frame pairs captured when the Kinects are rotated at 5-, 15-, 25- and 35-degrees angles.

Figure 5.14: Initial Matches of Dense Scene for Scenario-2

From top-left, top-right, bottom-left, bottom-right order the frame pairs are correspondent to Kinects being rotated with 5-, 15-, 25- and 35-degrees angles respectively.

There is a significant drop of the number of feature matches observed within a rotation of 30 degrees. The following pictures show the filtered matches for above frames to obtain the optimized pose.

Figure 5.15: Optimized Matches of Dense Scene for Scenario-2

From top-left, top-right, bottom-left, bottom-right order the frame pairs are correspondent to Kinects being rotated with 5-, 15-, 25- and 35-degrees angles respectively.

- **Comparison against Ground Truth**

The translation error, T_{error} and rotation error, R_{error} calculated between the estimated pose and the ground truth pose is shown in the following graphs. These errors were calculated as the same way as described in above Section 5.1.1.1 and these are plotted against the corresponding ground truth values.

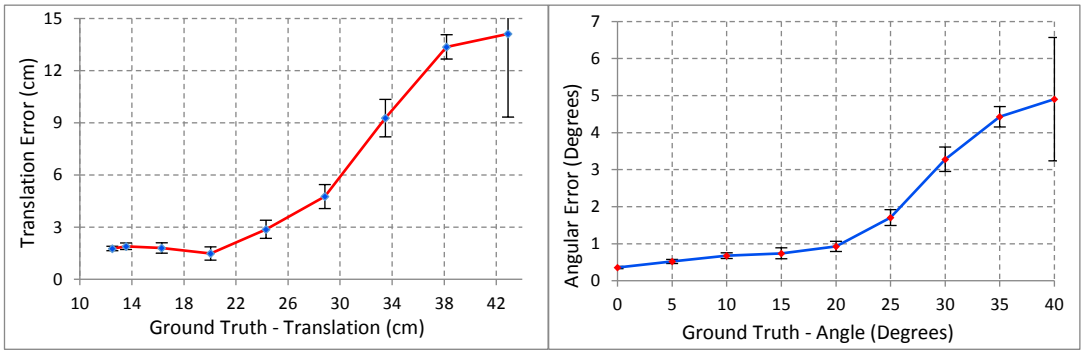


Figure 5.16: Euclidian Translation error and Angular Error for a dense scene when one sensor is systematically positioned with a translation and rotation

Above data shows that the average rotational accuracy for 40^0 rotation is 91.3% while the corresponding translational accuracy is around 81%. Compared to translation error between ground truth translation from 30-40cm for the dense scene in Scenario-1, there is a significant increase of 97% in the expected error for an equivalent translation predicted from these results. However, the rotation error is considerably less in this scenario. The following graph shows the estimated rotation about $X-, Y-$and $Z-$ axes. According to the plot, the rotation about $Y-$ axis is almost linear against the ground truth while the rotation about X and Z is very small and negligible compared to Y- axis rotation.

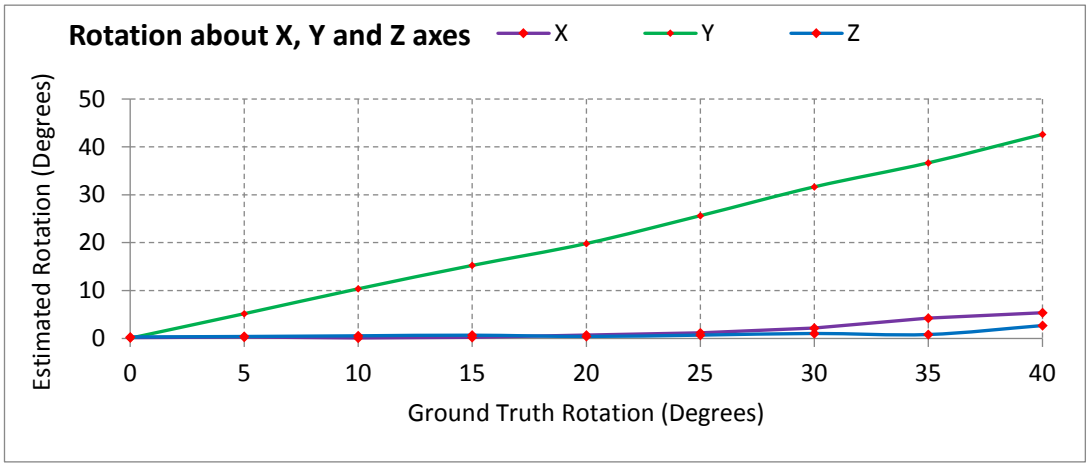


Figure 5.17: Rotation about X, Y and Z axes for a dense scene when one sensor is systematically positioned with a translation and rotation

The following Figure 5.18 shows the variation of percentage inlier feature matches for this experiment. Inlier percentage was calculated as explained in Section 5.1.1.1.

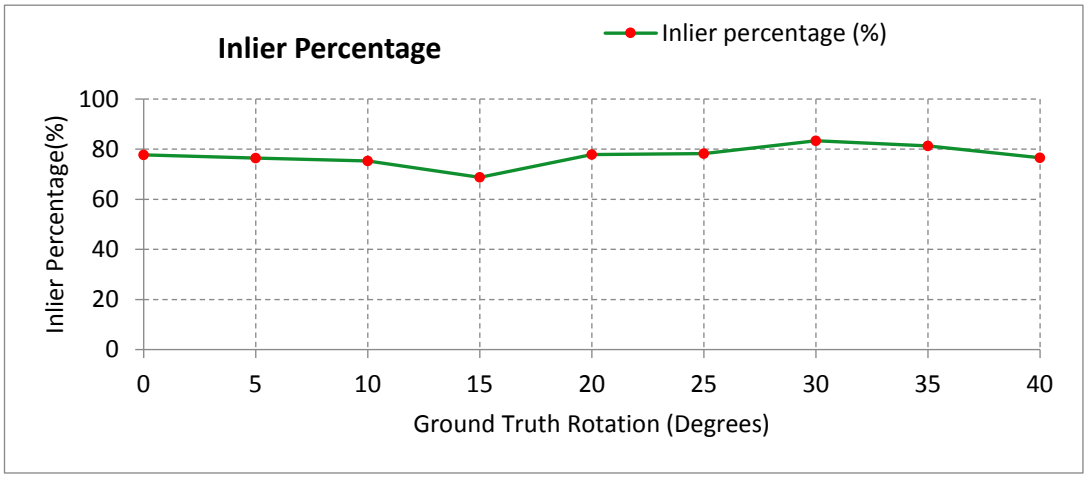


Figure 5.18: Inlier percentage for a dense scene when one sensor is systematically positioned with a translation and rotation

- **Comparison against ACRA Work**

We used the same data set on ACRA system for Scenario-2. The inlier matches filtered by this RGB-D to RGB localization approach is shown in the below Figure 5.19. In each frame pair, the left image is the moving Kinect image while the right image is considered to be the still Kinect.

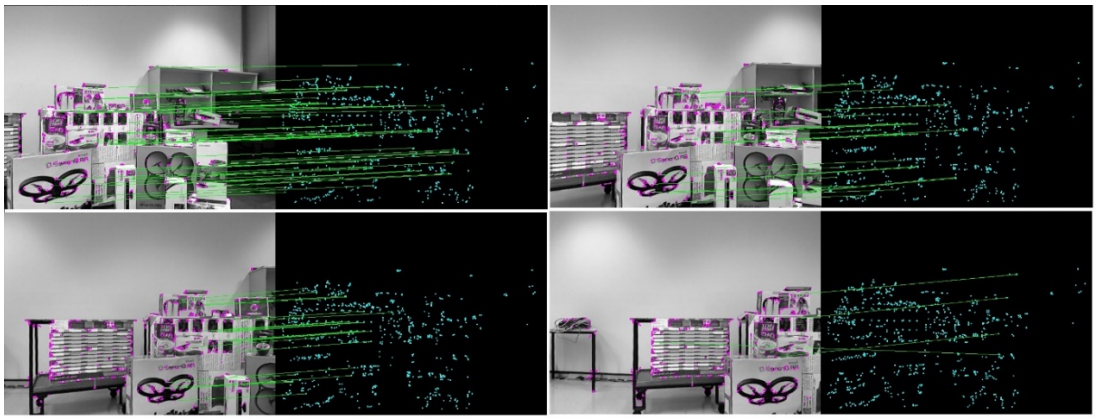


Figure 5.19: Inlier Matches obtained using ACRA method for Scenario-2 dense scene data set

From top-left, top-right, bottom-left, bottom-right order the frame pairs are correspondent to Kinects being rotated with 5-, 15-, 25- and 35-degrees angles respectively.

The translation and rotation error for both approaches compared to corresponding ground truth values are shown in the below graphs.

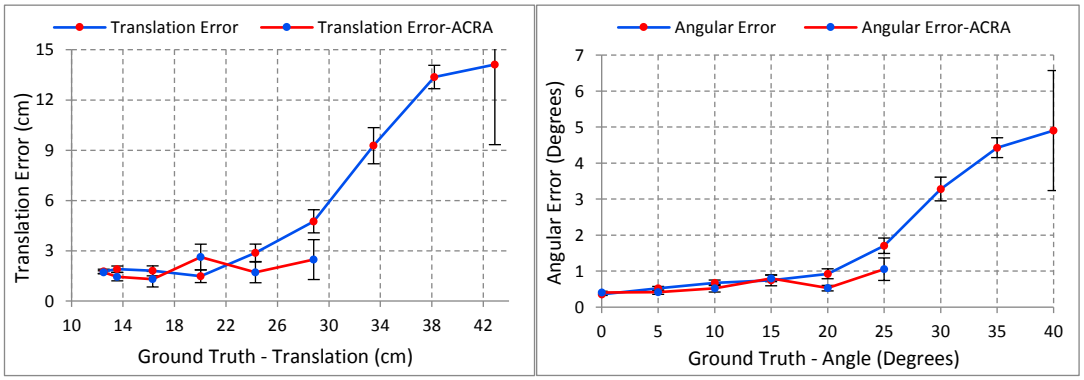


Figure 5.20: Translation and rotation error of estimated pose using proposed localization approach compared to previous (ACRA) approach for the dense scene in Scenario-2

According to above graphs the ACRA approach performs similar to the proposed approach, however it fails to estimate the pose when the Kinects are angled more than 25 degrees. Above data shows that our method ensures 93.3% average rotational accuracy over 25^0 span while ACRA method provides 94.8% accuracy. The corresponding average translational accuracy from our method is 87.5% while it is 89.8% from ACRA method. However, in this dense scene experiment our method increases the operating range by 60%.

The following graphs show the comparison of rotation about $X-, Y-$and $Z-$ axes for both approaches.

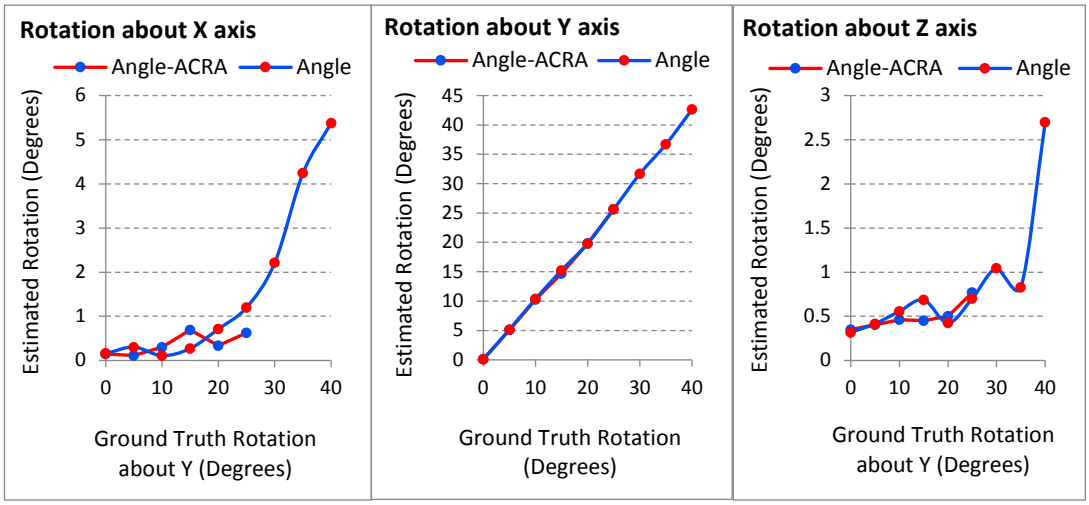


Figure 5.21: Estimated Angle about X. Y, Z axes for proposed approach and ACRA method for dense scene

Estimated rotation about Y axis is almost same for both approaches up to 25 degrees, while estimated rotation about $X-$and $Z-$ axes is different by each approach. However, the proposed approach gives 87%, 93.5% and 93% accuracy of angles about $X, Y$and Z axes respectively when the Kinects are even at 40 degrees angle.

The percentage inlier matches for both approaches is shown in the following graph. Considering the angular difference from 0 to 40 degrees, the inlier percentage of the proposed approach varies only by 14% while in ACRA approach, the inlier percentage varies by 38%.

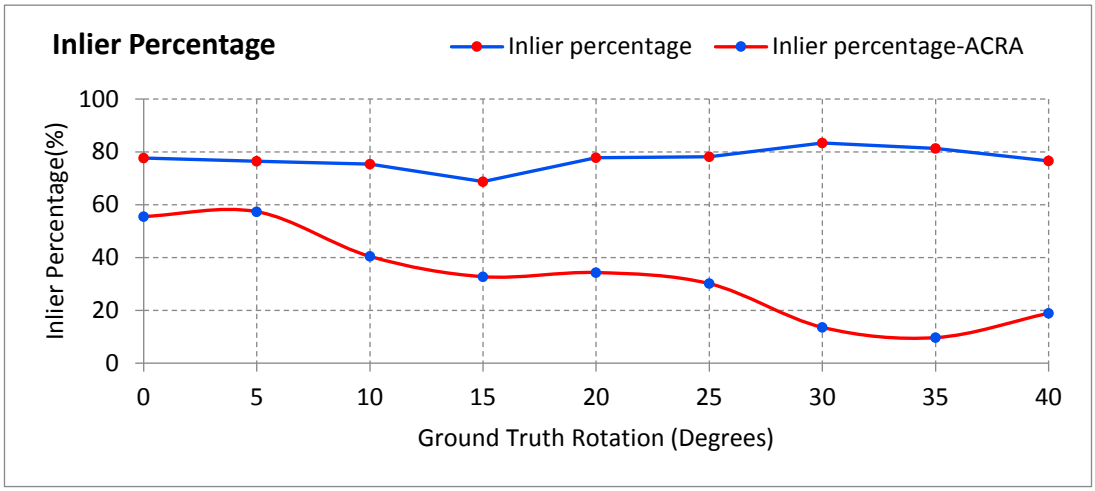


Figure 5.22: The Inlier Percentage of both approaches for angle between Kinects up to 40 degrees

5.1.2.2 Scenario - 2 Sparse Scene

The matched ORB features for the image pairs captured for sparse scene for Scenario-2 is shown below. As in the dense scene, the below image pairs are randomly picked from each ten frame pairs captured when the Kinects are rotated at 5-, 15-, 25- and 35- degrees angles.

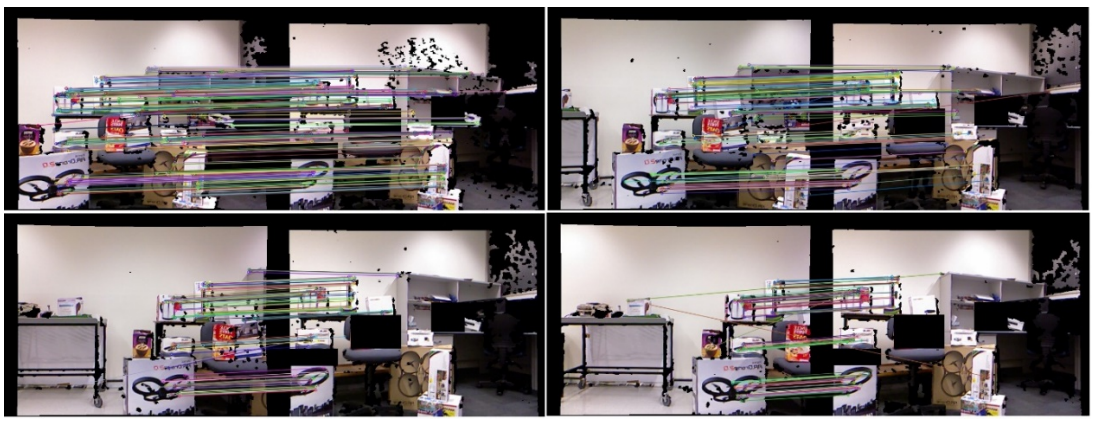


Figure 5.23: Initial matches of sparse scene for Scenario-2

From top-left, top-right, bottom-left, bottom-right order the frame pairs are correspondent to Kinects being rotated with 5-, 15-, 25- and 35-degrees angles respectively.

To make the scene sparser, we have systematically created occlusions as shown in the right-hand side images. This way, we could reduce the number of initial feature matches significantly compared to dense scene for the same scenario. The following table shows the average number of initial feature matches for both dense and sparse scenes for Scenario-2.

Ground Truth Angle about Y axis between the Kinects (degrees)	Average Number of Feature Matches in the Dense Data Set	Average Number of Feature Matches in the Sparse Data Set
0	666.7	425.7
5	531.7	338.5
10	311.2	254.2
15	187.8	182.9
20	150.9	122.7
25	137.9	119.5
30	115	110.3
35	96.7	119.6
40	25.8	46.2
45		51.1

Table 5.2: Average Number of Feature Matches for Dense and Sparse Scenes for Scenario-2

The filtered inliers for above four image pairs are shown in the following figure.

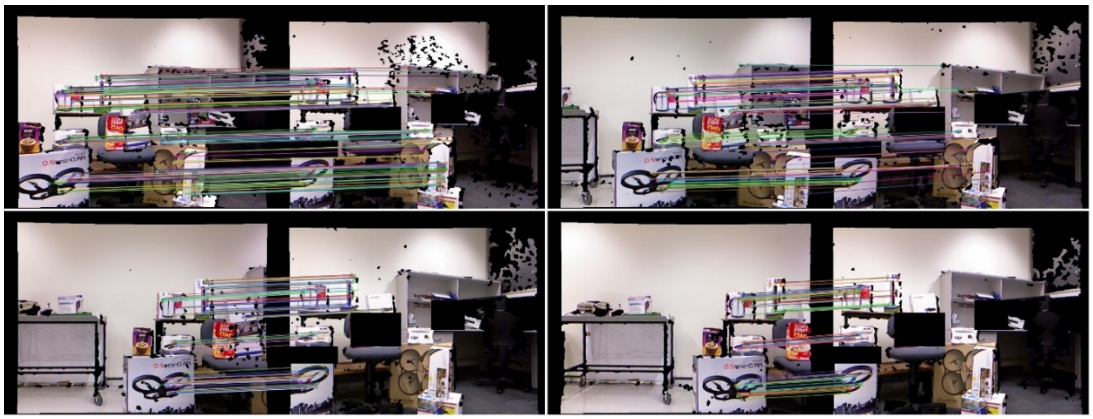


Figure 5.24: Optimized matches for sparse scene for Scenario-2

From top-left, top-right, bottom-left, bottom-right order the frame pairs are correspondent to Kinects being rotated with 5-, 15-, 25- and 35-degrees angles respectively.

- **Comparison against Ground Truth**

The following Figure 5.25 shows the comparison of translation and rotation errors of sparse and dense scenes for Scenario-2.

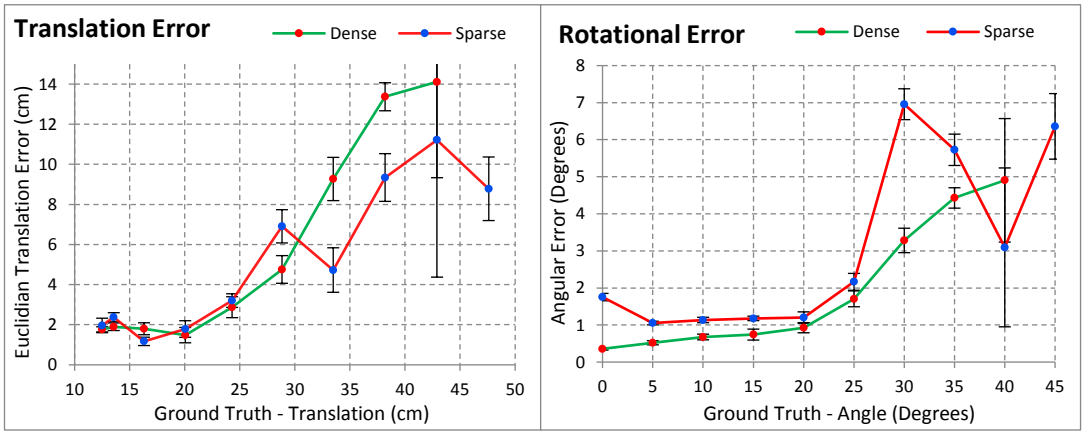


Figure 5.25: Translation and rotation error for dense and sparse scenes for Scenario-2

According to graphs, the translation and rotation errors of sparse scene seem irregular and varying drastically above and below the corresponding dense scene values. Above data shows that the average rotational accuracy of 87% over 45^0 span and the corresponding average translational accuracy of 83%.

The following Figure 5.26 shows the comparison of inlier percentage for both dense and sparse scenes.

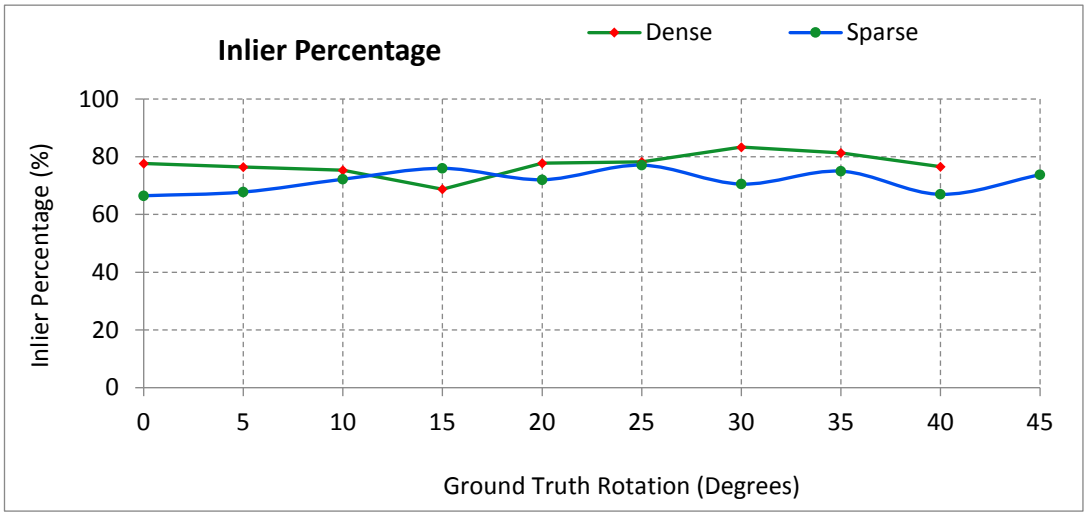


Figure 5.26: Inlier percentage for both dense and sparse scenes for Scenario-2

In average, the sparse scenes have about 5% less inliers chosen from the initial feature matches.

- **Comparison against ACRA Work**

Similar to Scenario-1, we used the same set of sparse scene data from Scenario-2 to run on ACRA localization approach. The following images show the inlier feature matches obtained by ACRA algorithm for the same set of image pairs shown above.

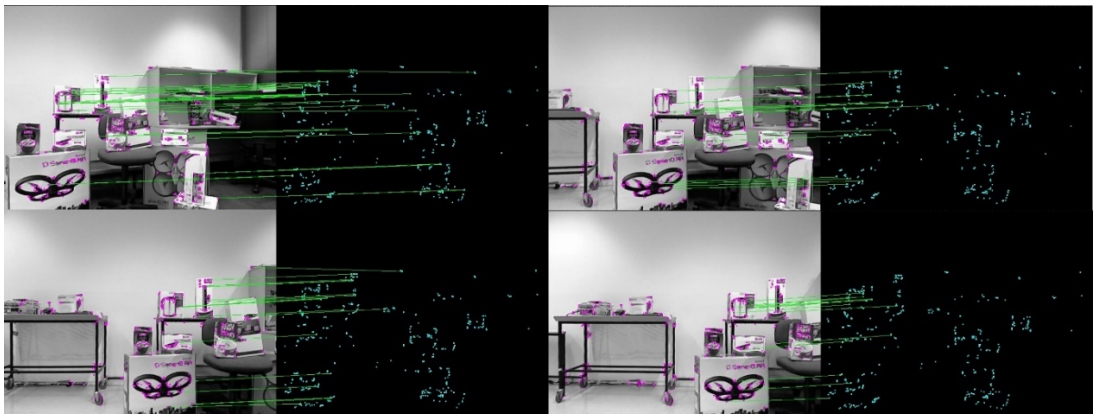


Figure 5.27: Inlier matches obtained using ACRA method for Scenario-2 sparse scene

From top-left, top-right, bottom-left, bottom-right order the frame pairs are correspondent to Kinects being rotated with 5-, 15-, 25- and 35-degrees angles respectively.

The following graphs show how ACRA method responded to the sparse scene in Scenario-2. The graphs show the comparison of translation and rotation errors for the same sparse scene using both approaches.

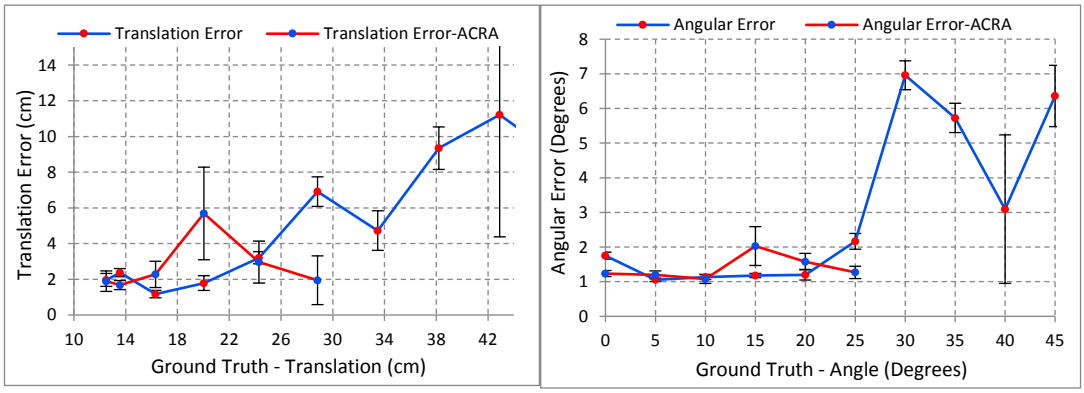


Figure 5.28: Translation and rotation error of estimated pose using proposed localization approach compared to previous (ACRA) approach for the sparse scene in Scenario-2

According to above graphs ACRA approach fails to estimate the pose when the Kinects are rotated more than 25 degrees. Above data shows that average rotational accuracy achieved by our method is 89% while it lies at 87.8% by ACRA method. Corresponding average translational accuracy achieved by our method is 85.6% while it is 85% from ACRA method. Therefore, in this sparse scene experiment, our method has performed better than ACRA method while it enhances the operating range by 80%. The following graphs show the comparison of rotation about $X-, Y-$and $Z-$ axes for both approaches.

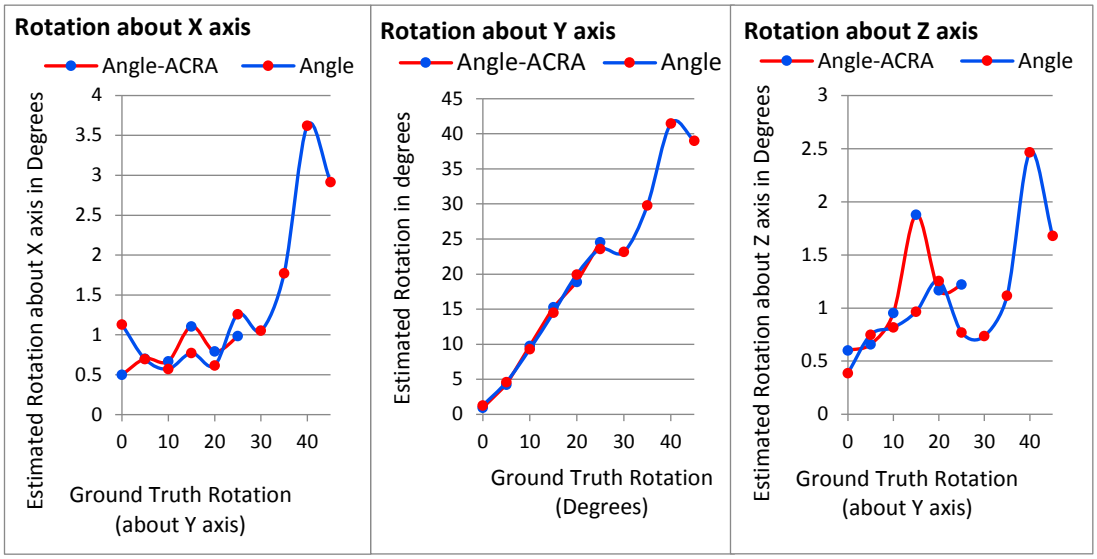


Figure 5.29: Estimated angle about X-, Y-, Z- axes for proposed approach and ACRA method for Scenario-2 sparse scene

Estimated rotation about Y axis is almost same for both approaches up to 25 degrees, while the estimated rotation about $X$and Z axes are different by each approach. However, the proposed approach gives 95%, 85% and 97% accuracy of angles about $X-, Y-$and $Z-$ axes respectively when the Kinects are even at 35 degrees angle.

The percentage inlier matches for both approaches is shown in the following graph. Considering the angular difference from 0 to 40 degrees, the inlier percentage of the proposed approach varies only in 10% while in ACRA approach the inlier percentage varies in 33%.

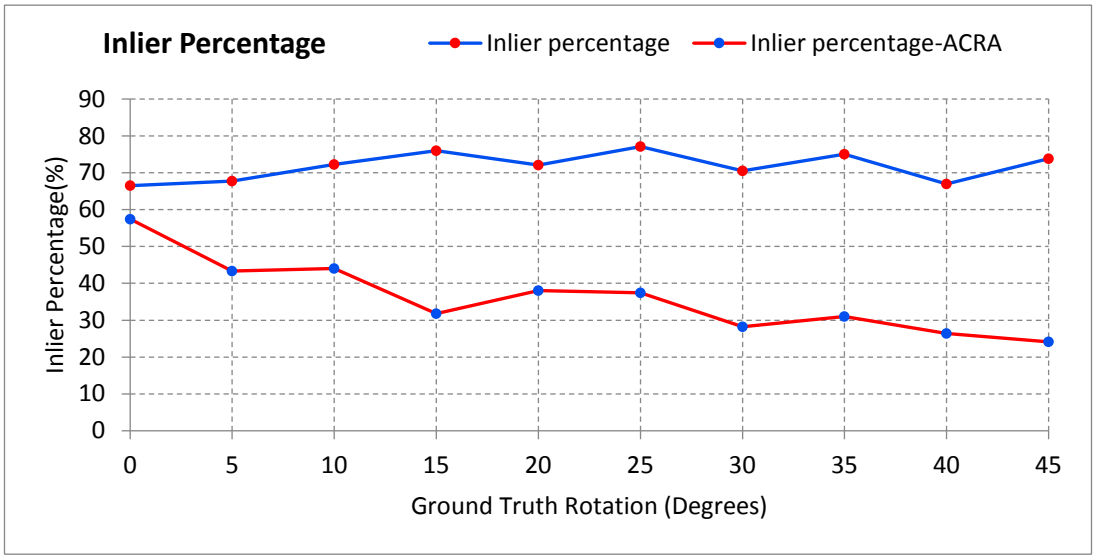


Figure 5.30: Percentage Inlier matches obtained from both approaches for Scenario-2 sparce scene

5.1.3 Scenario-3: Freehand movement of one sensor relative to a stationary sensor

This scenario is more suitable to evaluate the appropriateness of proposed work in AR applications where a human user moves an RGB-D device by hand which is then localized using another RGB-D sensor. Also, it can be used in swarm robotics applications such as formation control and surveillance where UAVs or drones with RGB-D sensors try to localize in 6-DOF relative to another helper robot carrying an RGB-D sensor.

The most challenging task while proving the concept in this scenario was to obtain accurate ground truth poses of the free-hand moving RGB-D sensor. With the experiments conducted in the laboratory environment, it was difficult to

record precise ground truth information with the available limited resources. Therefore, the investigation of proposed work in localizing a moving RGB-D sensor becomes twofold. One method is to examine the proposed approach on the data collected in laboratory environment with less accurate ground truth information. The other method is to evaluate using publicly available dataset with precise ground truth information. In this section we demonstrate the accuracy of the proposed work using these two methods.

As explained in Section 4.3.1.3, in the first method we collected data frames from Kinects considering dense and sparse scenes while keeping Kinect-1 stable and freehand moving Kinect-2 on known trajectories. The following two sub sections present the results obtained from these experiments.

5.1.3.1 Scenario - 3 Dense Scene

Similar to Scenario 1 and 2, the experiment was conducted for both dense and sparse scenes. The experimental results of moving Kinect-2 along the perimeter of a rectangular shape on X-Y plane in front of the dense scene is shown in Figure 5.31.

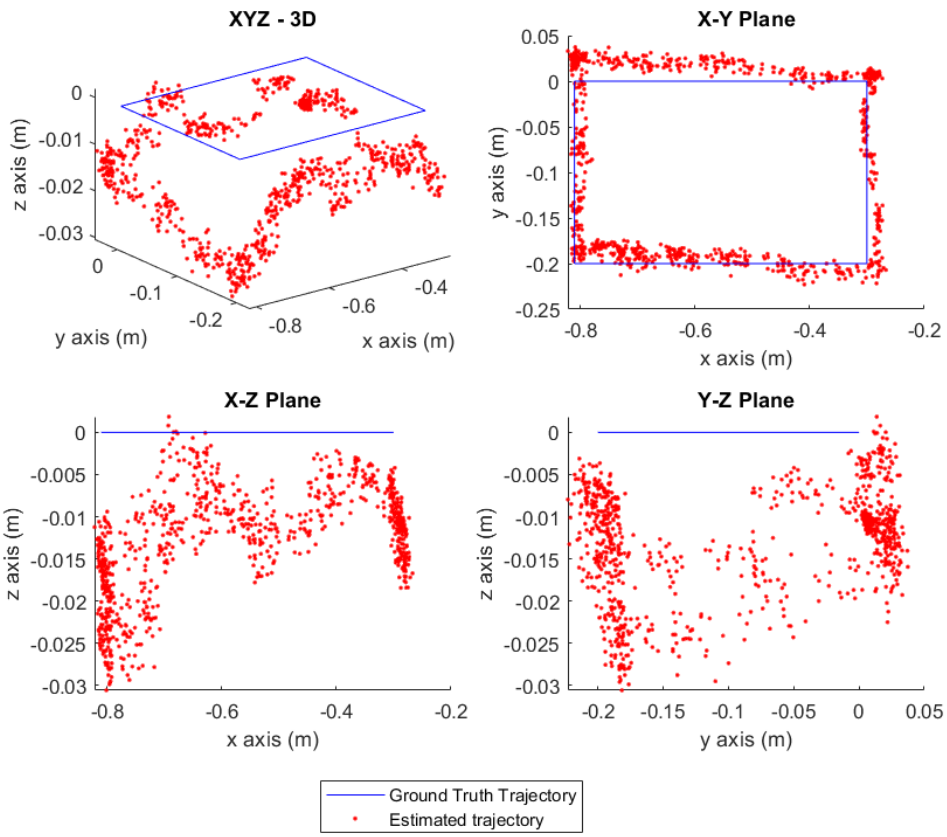


Figure 5.31: Scenario-3 Dense Scene - Freehand movement of Kinect-2

The figure shows the ideal ground truth trajectory and estimated trajectory for the dense scene in different view planes. The starting and end point is (-0.3, 0, 0). The total ground truth trajectory length is 1.42m while rms-translation error is 0.116m and rms-rotational error is 0.0088 radians. Average velocity of ground truth trajectory is around 0.01m/s.

From Figure 5.31, the Z- and Y-error are minimum which lies within 3cm, hence providing above 85% accuracy in Y direction. However, as explained in Section 4.3.1.3, there can be significant human errors while trying to move the Kinect along ideal ground truth trajectory. Also, the experiment was conducted by a single person, so there were considerable delays between data collection start and Kinect-2 movement start. Likewise, at the end of the experiment the delays

happened between stop moving Kinect-2 to end of data collection. This can be seen in X-Y and X-Z sub-plots having a thick cloud of points at the starting position (-0.3, 0, 0). These factors were not considered while estimating the average speed of ground truth trajectory. The speed measurement here gives an understanding of how fast a robot can move relative to the stationary sensor. Average speed was estimated by taking the trajectory length and the overall experiment time. Therefore, delays at the start and the end could cause the speed calculation hence this calculated speed is only an approximation.

5.1.3.2 Scenario - 3 Sparse Scene

The experimental results of the estimated trajectory when moving Kinect-2 along the perimeter of a rectangle on X-Y plane in front of the sparse scene is shown in the below Figure 5.32.

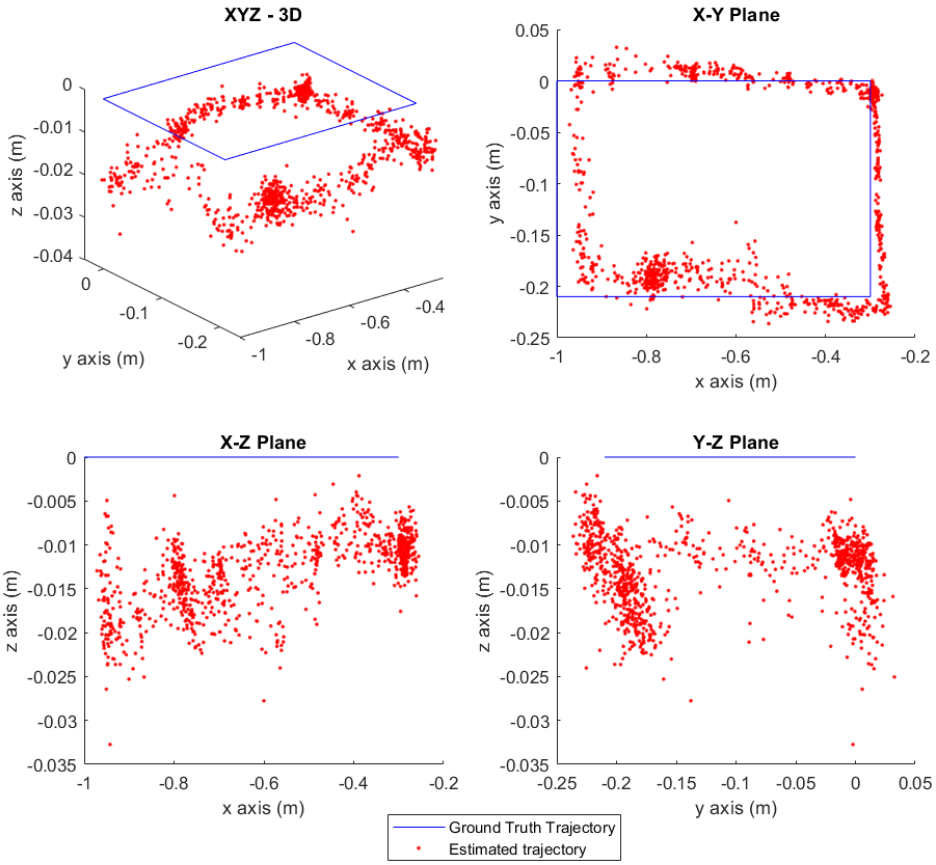


Figure 5.32: Scenario-3 Sparse Scene - Freehand movement of Kinect-2

The figure shows the ideal ground truth trajectory and estimated trajectory for the sparse scene in different view planes. The starting and end point is (-0.3, 0, 0). The total ground truth trajectory length is 1.82m while rms-translation error is 0.189m and rms-rotational error is 0.0122 radians. Average velocity of ground truth trajectory is around 0.01m/s.

The ground truth trajectory for sparse scene is longer than the dense scene trajectory, moving Kinect-2 more towards (-X) axis. However, the translation error along Z axis still lies within 3.5cm. There is a significant increase in translation error along X-axis compared to dense scene when Kinect-2 was moved far away from Kinect-1. The same human errors explained in dense scene

experiment of this scenario also applies to this sparse scene experiment and above results could be affected by these errors.

5.1.3.3 Qualitative Comparison with TUM Dataset

Since the ground truth information for above dense and sparse scene approaches are prone to human errors, we use TUM Benchmark dataset [43] to qualitatively evaluate the accuracy of our proposed approach. This benchmark contains multiple real datasets captured with an RGB-D camera. Every dataset accompanies an accurate ground truth trajectory obtained with an external motion capture system which we were unable to acquire in the previous laboratory experiment methods. The purpose of evaluating against such a dataset is to examine the robustness of the proposed approach in practical applications such as 6-DOF localization of robots and AR applications. However, data collected from a single Kinect may not entirely simulate the real environment for Scenario-3 in which one Kinect moves relative to a stable Kinect. Since there are no such multiple Kinect datasets available in the recent research, this is one option we could consider evaluating our approach.

We use dataset sequences from "Testing and Debugging" and "Handheld SLAM" categories to test on our algorithm. These datasets consist of color and depth images captured from a single moving Kinect. The approach in Scenario-3 is to estimate the pose of a moving Kinect relative to a stationary Kinect. Therefore, when using TUM dataset sequences for Scenario-3, we consider estimating the pose of each successive Kinect frame relative to the first Kinect frame in the sequence. However, most of these TUM dataset sequences have

lengthier trajectories and sudden camera motions which makes it challenging to estimate pose in this way due to the reason that a significant overlap is required between the two keyframes being matched. When the Kinect is moved or turned too far away from the first reference view, the proposed algorithm fails due to lack of inlier matches. Therefore, one solution to address this issue in real world scenario would be to relocate the reference Kinect to a known pose so that it enhances the overlapped view with the moving Kinect. With the benchmark dataset, this can be achieved by shifting the reference frame to a further point along the trajectory to have a good number of inliers.

In this evaluation, we consider two methods to decide on which data frame to be shifted as the new reference frame. One method is to shift the reference frame based on the number of inlier-matches. When the number of inlier matches drop below a certain threshold ($tr_{inliers}$) the reference frame is being shifted/replaced by the currently moving frame. This threshold value is guessed based on the average number of inliers which varies for different sequences.

As the second method we consider running the dataset by shifting the reference frame after every fixed number of frames (tr_{num_frames}). With this approach, the algorithm might still fail or misbehave based on the number of inliers. The quality of the results depends on the nature of the trajectory, i.e. the angular and translational difference between the frames, the speed of moving Kinect etc. We run each sequence given below based on these two different methods.

- **Comparison against Freiburg2 xyz**

For our first evaluation we use fr2/xyz sequence from Freiburg2 dataset. This sequence has simple motions of a hand-held Kinect roughly along X-, Y-, Z-axes. The movement along three axes remains approximately within 0 - 1.5m and the speed of moving Kinect is relatively slow compared to fr1/xyz sequences. The slow camera motion basically ensures that there is (almost) no motion blur and rolling shutter effects in the data. Hence the results of this sequence show the behavior of our approach in the best-case. We shifted the Kinect reference frame every time when the inlier number of matches drop below 200 (*i.e* $tr_{inliers} = 200$). The following figures show an example situation where the reference frame has shifted when the number of inliers reduces.

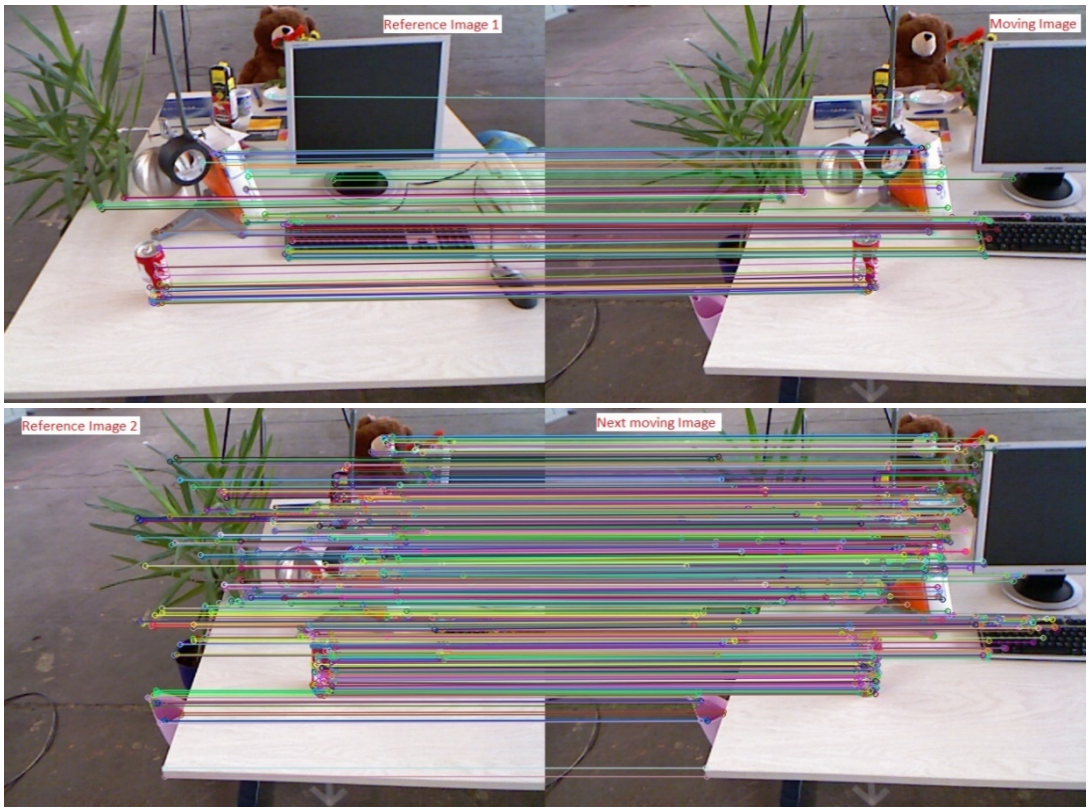


Figure 5.33: Shifting the reference frame when number of inliers drops below threshold

When the number of inliers drop below threshold, the moving image on top-right becomes the new reference frame as shown on bottom-left.

The following Figure 5.34 shows the estimated trajectory for the sequence fr2/xyz at different view angles. The estimated trajectory for the same data sequence when evaluated by shifting the reference frame with $tr_{inliers} = 100$ is shown in the following figures overlaid with the trajectory estimated with $tr_{inliers} = 200$.

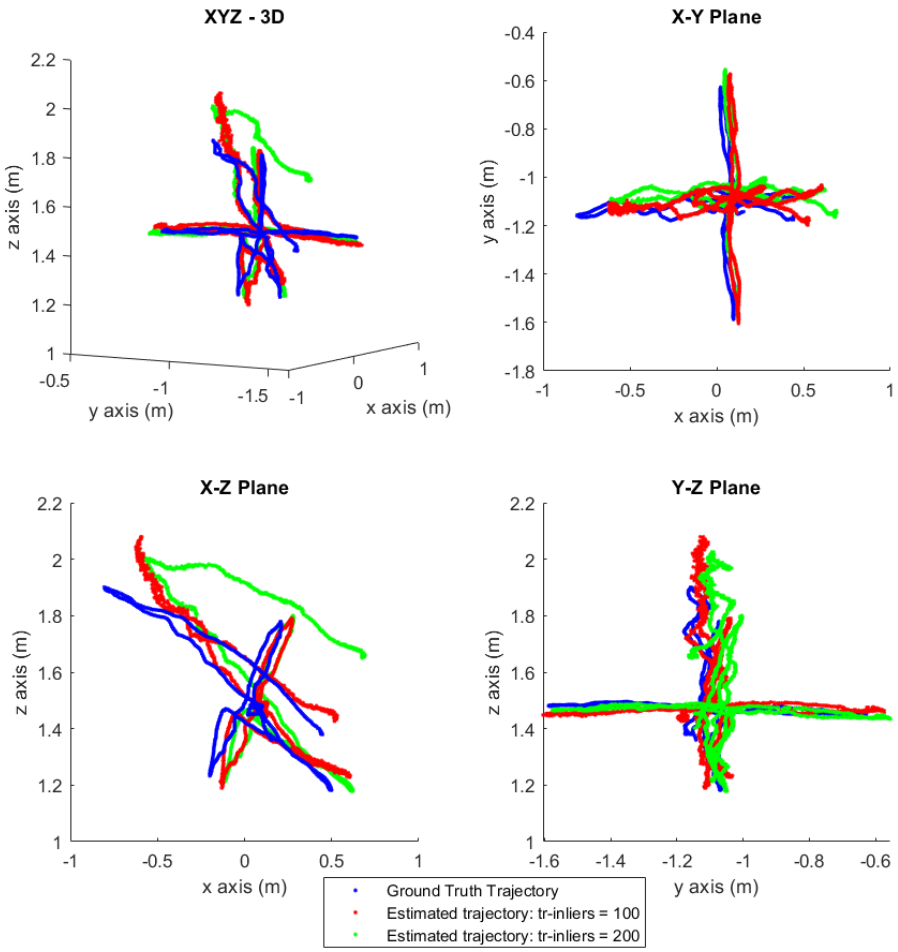


Figure 5.34: Estimated Trajectory for fr2/xyz sequence with $tr_{inliers} = 100$ and $tr_{inliers} = 200$

Ground truth trajectory length=7.029m, Average translational velocity=0.058m/s, Average angular velocity = 1.716deg/s, Trajectory dimensions: 1.30m x 0.96m x 0.72m.

The trajectory when $tr_{inliers} = 100$ seems very much aligned with ground truth trajectory while $tr_{inliers} = 200$ trajectory shows some misalignment in the last section. The following Figure 5.35 shows the frequency of changing the reference frame along the trajectory and the number of moving data frames relative to each reference frame.

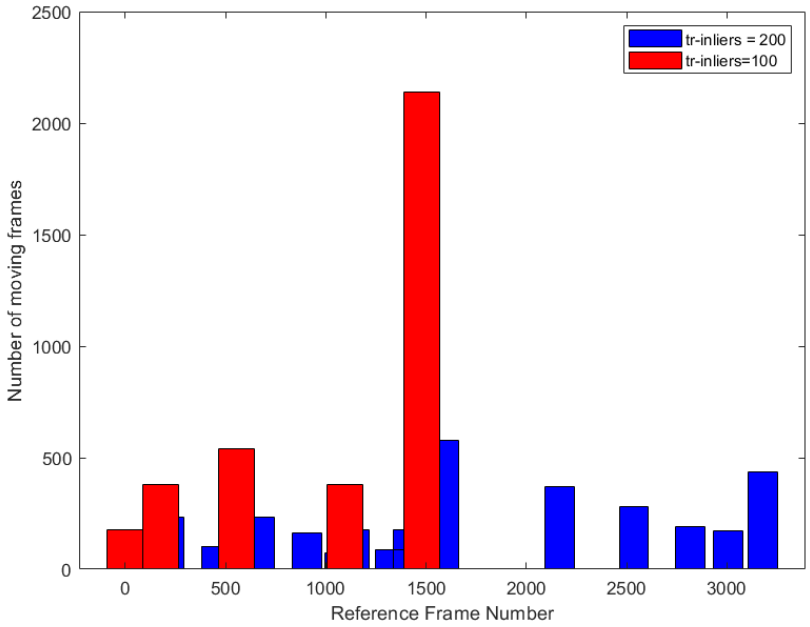


Figure 5.35: Histogram plot of number of moving data frames relative to each reference frame for fr2/xyz sequence with different $tr_{inliers}$ values

There are a smaller number of histograms in above plot for $tr_{inliers} = 100$ analysis compared to $tr_{inliers} = 200$ case due to changing the reference frame only a few times. The number of inliers never drop below 100 for the second half of the trajectory in $tr_{inliers} = 100$ case. But it is different with $tr_{inliers} = 200$ situation where the reference frame changed many times in the second half.

The estimated trajectory when the same data sequence is run by shifting the reference frame after every 300th frame ($tr_{num\_frames} = 300$) is shown in the following figure.

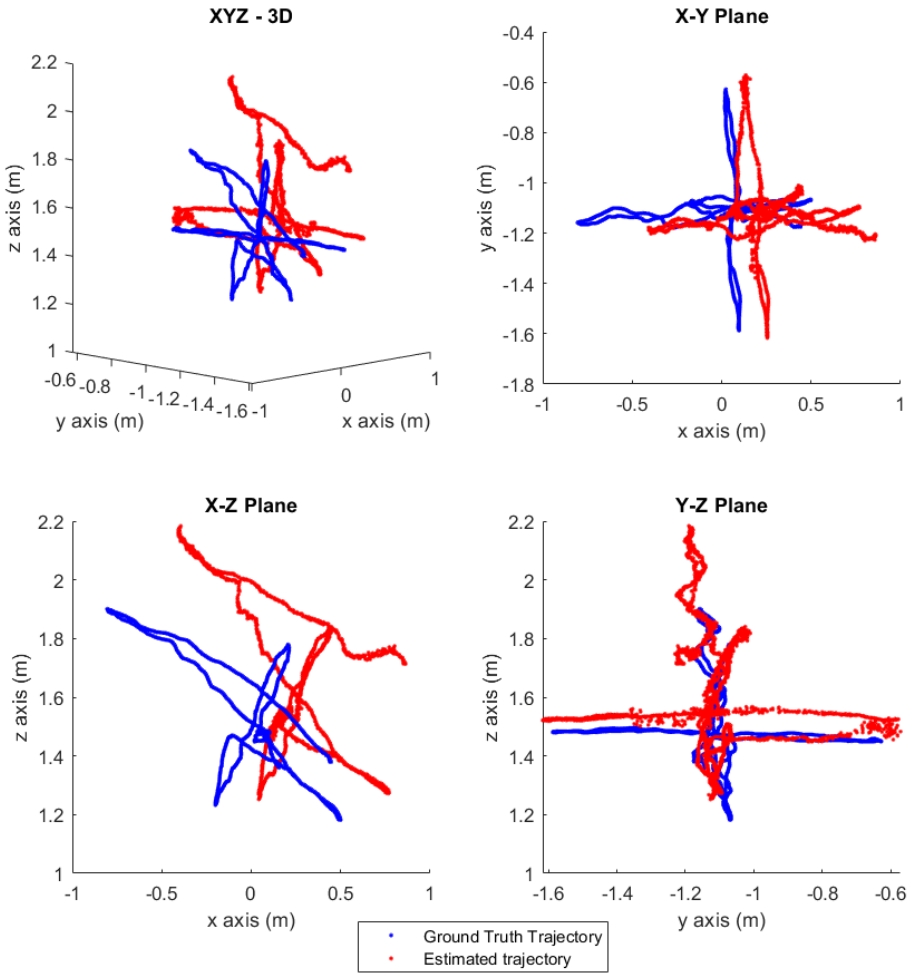


Figure 5.36: Estimated Trajectory for fr2/xyz sequence with $tr_{num\_frames} = 300$

The estimated trajectory has the similar ground truth trajectory shape but have shifted off significantly. The following figure shows the number of inliers variation along the trajectory for above analysis.

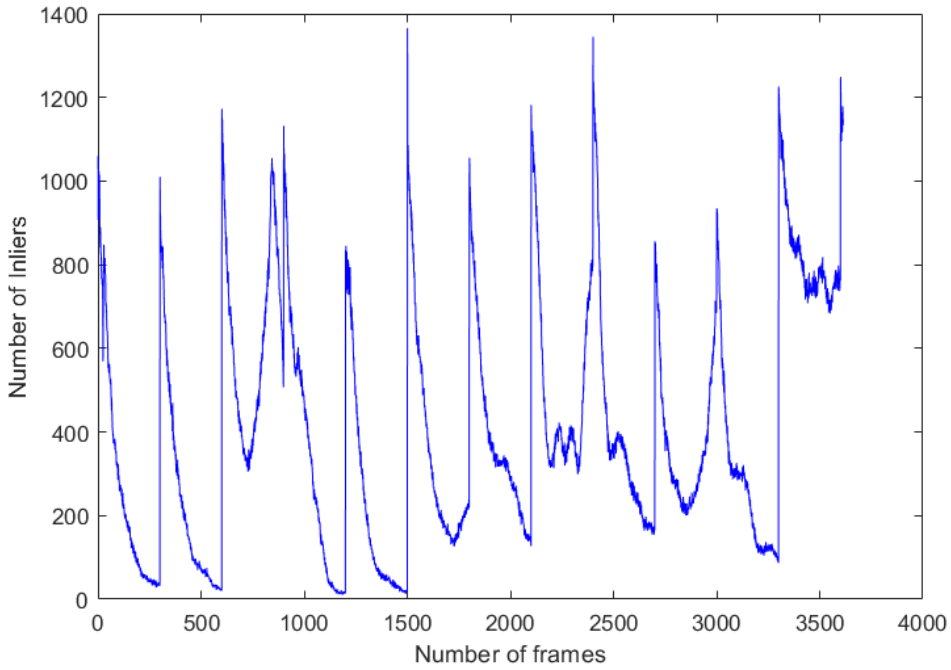


Figure 5.37: Inliers variation for fr2/xyz sequence with $tr_{num\_frames} = 300$

It can be seen that the inliers have dropped significantly before the reference frames 300, 600, 1200 and 1500. The incorrect pose estimated with these few inliers might cause the offset in the predicted trajectory.

From above figures it can be seen that $tr_{inliers}$ method is more stable than tr_{num_frames} method. The reason for the instability of the latter approach is that in fr2/xyz sequence the Kinect changes the direction of motion frequently and there is more possibility of not having the current reference frame and the moving frame overlapped in many times. Also, a significant drift can be observed in the estimated trajectory due to accumulating the error in estimated relative pose.

- **Comparison against Freiburg2 desk**

For the second evaluation, we use fr2/desk sequence where the images are recorded in a typical office environment with two desks, a computer monitor, keyboard, phone, chairs, etc. The Kinect is moved around the two tables so that the loop is closed. This sequence consists of relatively long trajectory approximately around 18m. The movement of Kinect along Z-axis is less significant compared to movement along X- and Y-axes. Even though the speed of Kinect is slow, there is a significant translation along X- and Y- axes. Due to this nature of the data sequence, reference frame needs to be shifted more frequently than that of in fr2/xyz sequence. The number of inlier feature matches also tend to drop drastically along the trajectory. Therefore, for this sequence we considered to evaluate against $tr_{inliers} = 100$, $tr_{inliers} = 50$ and $tr_{num\_frames} = 50$.

The following Figure 5.38 shows the estimated trajectory for fr2/desk sequence for $tr_{inliers} = 100$ and $tr_{inliers} = 50$ on the same plot.

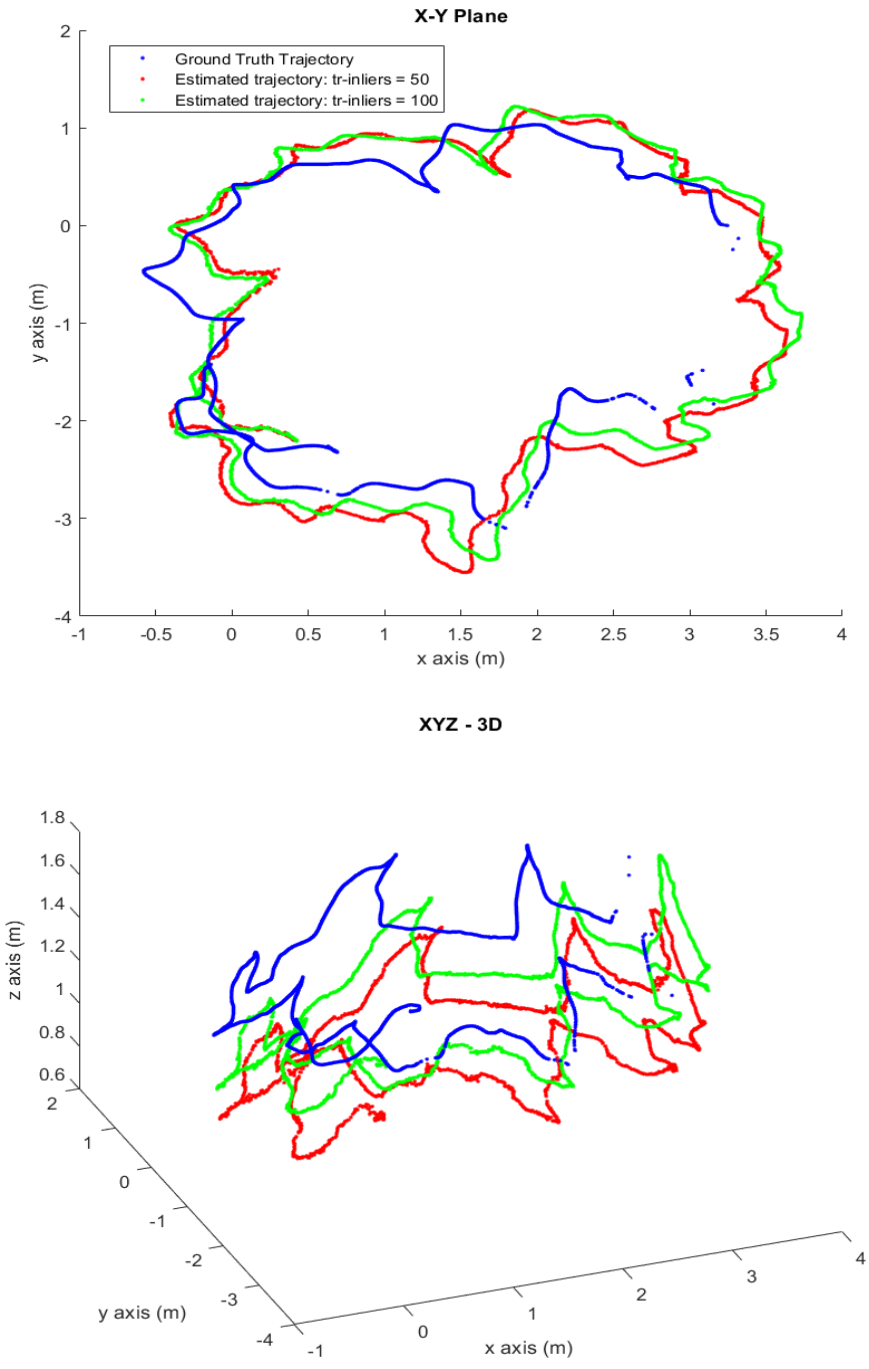


Figure 5.38: Estimated Trajectory for fr2/desk sequence with different $tr_{inliers}$values

Ground truth trajectory length=18.880m, Average translational velocity=0.193m/s, Average angular velocity = 6.338deg/s, Trajectory dimensions: 3.90m x 4.13m x 0.57m.

The following figure shows the frequency of changing the reference frame along the trajectory and the number of moving data frames relative to each reference frame.

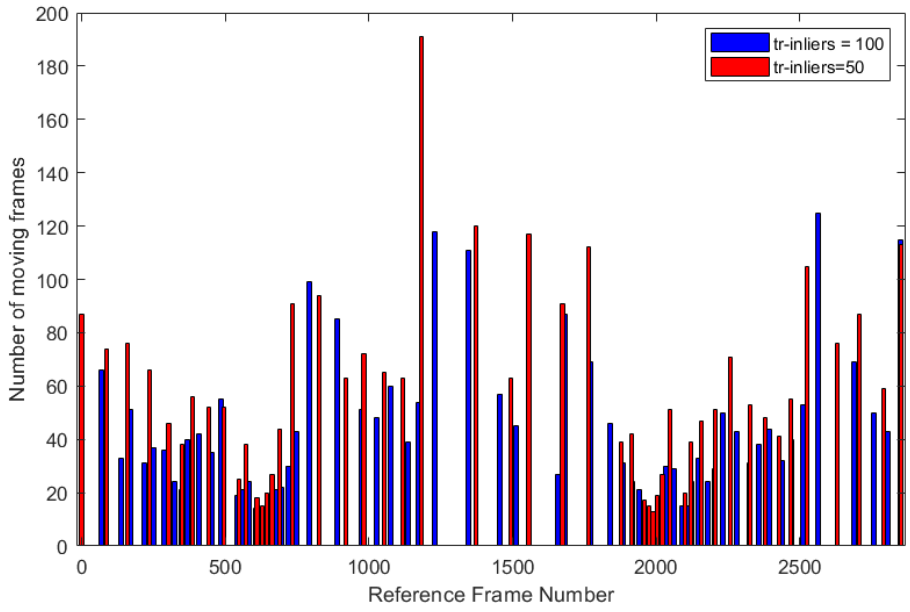


Figure 5.39: Histogram plot of number of moving data frames relative to each reference frame

From above graph it can be seen that the reference frame has frequently changed between the frames (~600-700) and (~1900-2100) for $tr_{inliers} = 100$ case. The red histograms being mostly taller than the blue histograms proves that the reference frame has changed less frequently in $tr_{inliers} = 50$ experiment than $tr_{inliers} = 100$ experiment.

We ran our algorithm on the same data sequence by shifting the reference frame after every 50th frame, i.e. $tr_{num\_inliers} = 50$ and the estimated trajectory is shown in the below Figure 5.40.

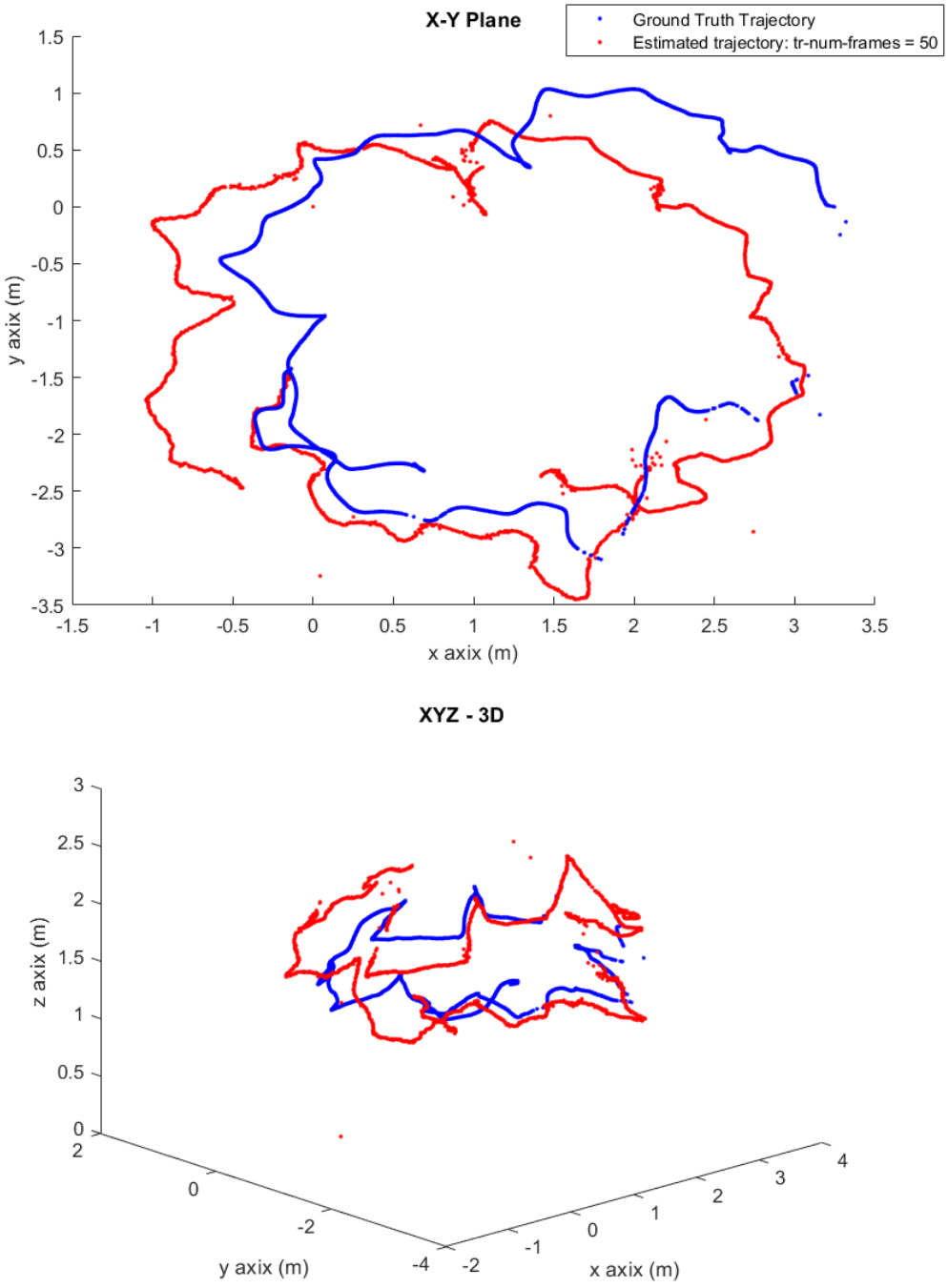


Figure 5.40: Estimated Trajectory for fr2-desk sequence with $tr_{num\_frames} = 50$

The variation of the number of inlier matches along the trajectory is show in the below graph.

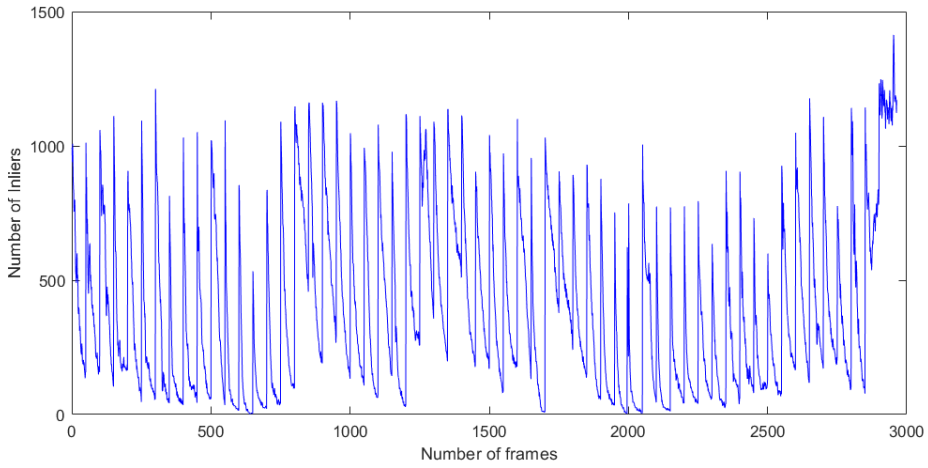


Figure 5.41: Variation of Number of inliers along the trajectory when $tr_{num\_frames} = 50$

From above figure it can be seen that the number of inliers reduces drastically along the trajectory even after the reference frame is shifted every 50th frame. Compared to above Figure 5.40 X-Y Plane trajectory, there is a significant drift in the trajectory at around 650th frame and this drift accumulates along the trajectory. The reason for this can be the number of inliers being approximately zero around this frame number according to Figure 5.41.

Since fr2/desk sequence has images captured when the Kinect is moved around two tables so that the loop is closed, this gives us opportunity to test the robustness of proposed approach in a different way. When the trajectory closes the loop, it comes to a point where the images are overlapped with the first data frame of the sequence. This is equivalent to a scenario where the second camera suddenly comes to a point so that it shares some view with the still Kinect. So

far in our analysis we considered the situations where the second Kinect always moves away from the first Kinect. However, with this new analysis we could test the robustness of our approach in the opposite way.

The following figures show the last part of the trajectory which was estimated when there is a reasonable overlap between the moving data frames and the first data frame.

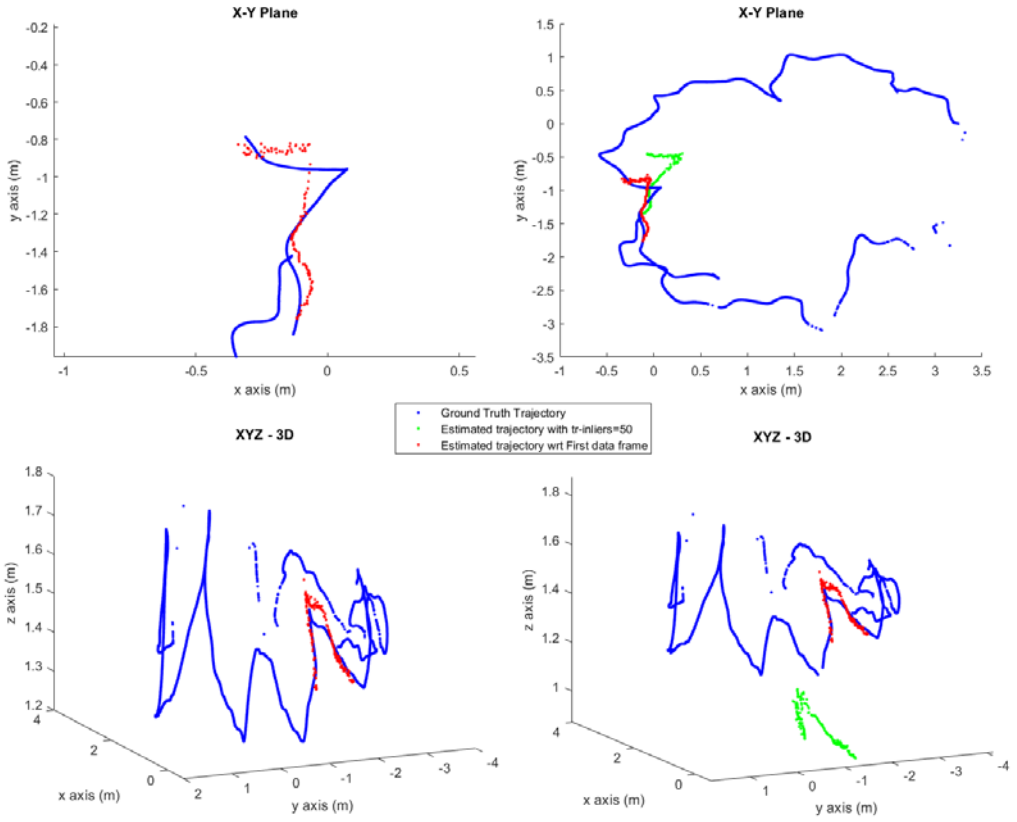


Figure 5.42: The last section of the trajectory estimated wrt First data frame

A zoomed-out view is shown on the top-left sub-plot and it can be seen that the error in estimated trajectory is minimal. When the same part of the trajectory is compared with the previously analyzed $tr_{num\_inliers} = 50$ trajectory, a large

error can be observed, due to error accumulation over the loop. Having a minimal error in this scenario, we prove the robustness of our approach when the second Kinect moves towards the static Kinect.

- **Comparison against Freiburg1 xyz**

To investigate the behavior of the proposed work in the worst-case scenario, we used fr1/xyz and fr1/desk sequences to analyze the performance. The following figure shows the estimated trajectory for fr1/xyz in the two situations where the reference frame is changed when number of inliers reduced less than 100 and 200.

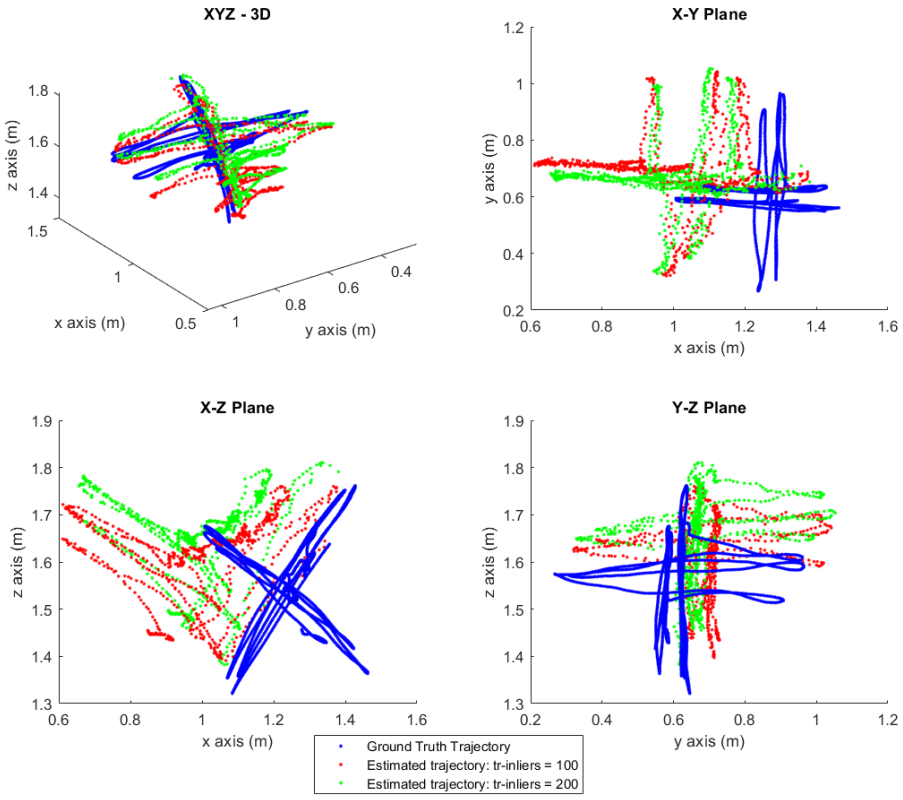


Figure 5.43: Estimated Trajectory for fr1/xyz sequence with $tr_{inliers} = 100$ and $tr_{inliers} = 200$

Ground truth trajectory length=7.112m, Average translational velocity=0.244m/s, Average angular velocity = 8.920deg/s, Trajectory dimensions: 0.46m x 0.70m x 0.44m.

Above plots in Figure 5.43 shows the estimated trajectory in the worst-case scenario because the camera has moved very fast compared to other two data sequences analyzed before. According to the sample images shown in the following figure, it can be clearly seen that there is a significant difference in the images due to motion blur. This may cause the large errors observed in the estimated trajectory. However, the trajectory when the inlier threshold is 200 seems slightly improved and slightly closer to ground truth trajectory when seen from X-Y, Y-Z and X-Z plane plots.

Figure 5.44: Sample Images from fr1/xyz and fr2/xyz Sequences

Top row: fr1/xyz sequence images with motion blur due to fast camera movements. Bottom row: Images from fr2/xyz and the images are fairly clear.

The histogram plot of where the reference frame has changed along the trajectory is shown in the below figure. Compared to the same plot for fr2/xyz and considering the number total data frames in two sequences, we can observe a significant growth in changing the reference frame. While the number of data frames is about 78% less in this sequence compared to fr2/xyz, there is a frequency

increment of three times per every 100 frames in the reference frame change for $tr_{inliers} = 100$ and six times increment when $tr_{inliers} = 200$ case.

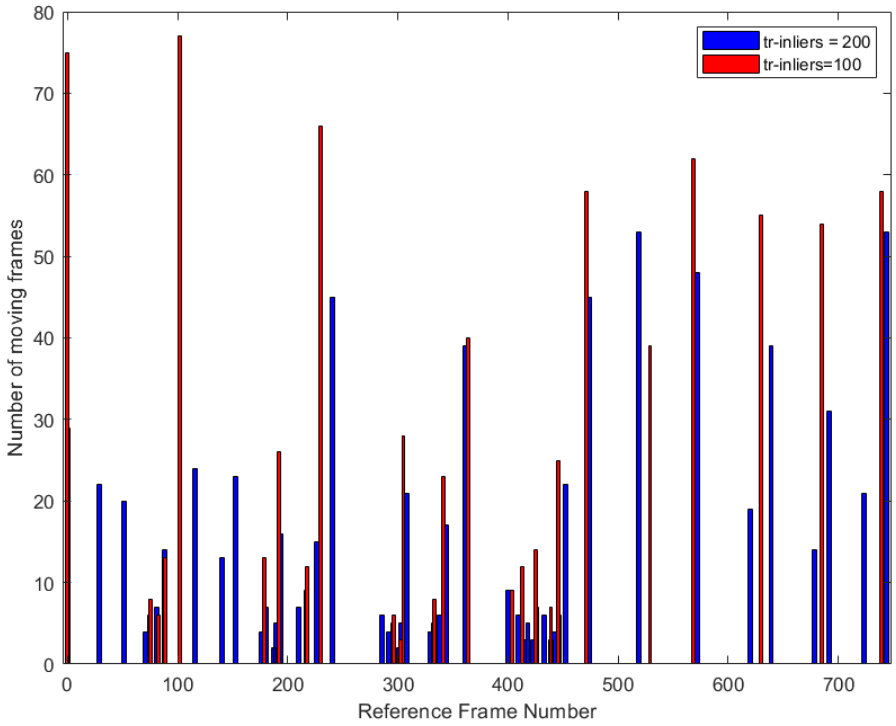


Figure 5.45: Histogram plot of number of moving data frames relative to each reference frame for fr1/xyz sequence with different $tr_{inliers}$ values

- **Comparison against Freiburg1 desk**

To study the behavior of proposed work further in worst-case scenario, we compared the accuracy against fr1/desk sequence, which is nearly two times faster in translational velocity and three times faster in angular velocity compared to

fr1/xyz sequence. The resulted trajectory is shown in the below figure for $tr_{inliers} = 50$ and $tr_{inliers} = 100$ analysis.

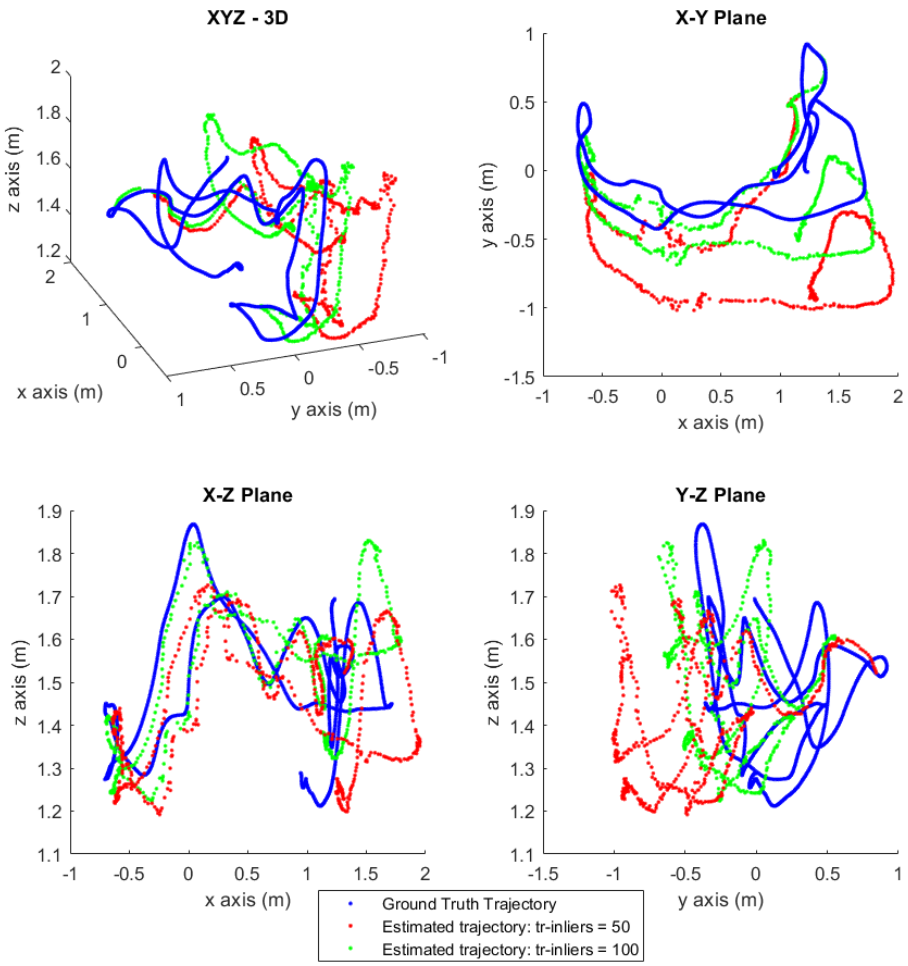


Figure 5.46: Estimated Trajectory for fr1/desk sequence with $tr_{inliers} = 50$ and $tr_{inliers} = 100$

Ground truth trajectory length=9.263m, Average translational velocity=0.413m/s, Average angular velocity = 23.327deg/s, Trajectory dimensions: 2.42m x 1.34m x 0.66m.

Even with the worst-case scenario, the results seem still acceptable. When $tr_{inliers} = 100$ case is considered, the drift in estimated trajectory is relatively higher along Y axis while it is very minimal along X- and Z- axes. From above plots, the Y-error lies within about 40cm for $tr_{inliers} = 100$ case while it is twice higher for $tr_{inliers} = 50$ scenario. The X- and Z- axes error for $tr_{inliers} = 50$ trajectory is also considerably larger.

These large errors in $tr_{inliers} = 50$ trajectory can be because the number of inlier features drastically reduces from frame to frame and the average number of inliers are less than 150 in most cases as shown in Figure 5.47.

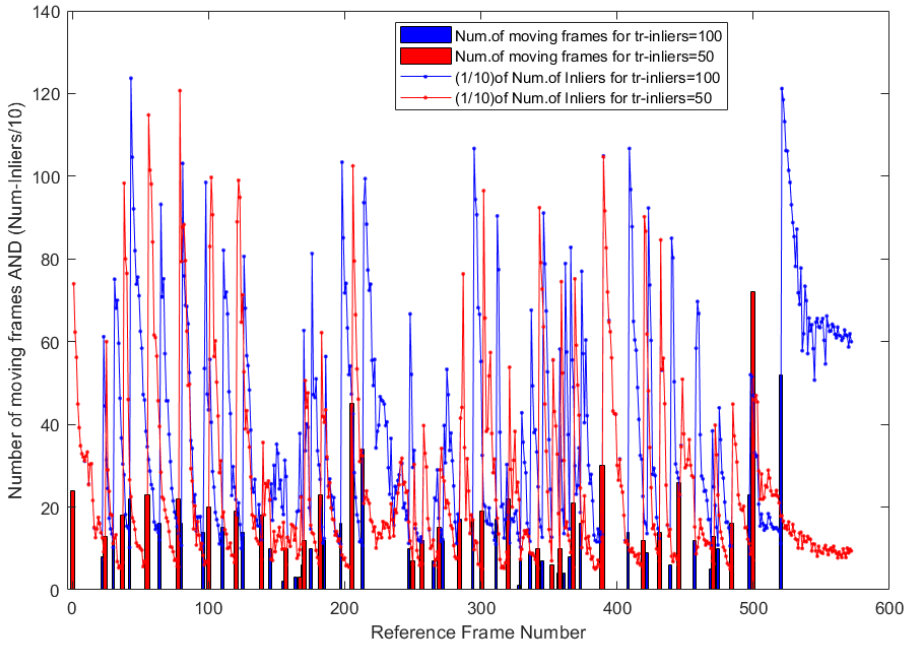


Figure 5.47: Frequency of changing the reference frame and the number of Inliers

Frequency of changing the reference frame is given for both $tr_{inliers} = 50$ and $tr_{inliers} = 100$ cases. The map of the number of inliers are

overlaid on the same plot. Please note that the number of inliers scaled down by 10.

However, it can be seen that there is substantial amount of motion blur and rolling shutter effect occurred in the color images due to fast camera movements as shown in Figure 5.48. Due to image distortions when the camera moves quickly, the estimated pose could be severely affected. Hence, we can observe the significant drift in the trajectory.

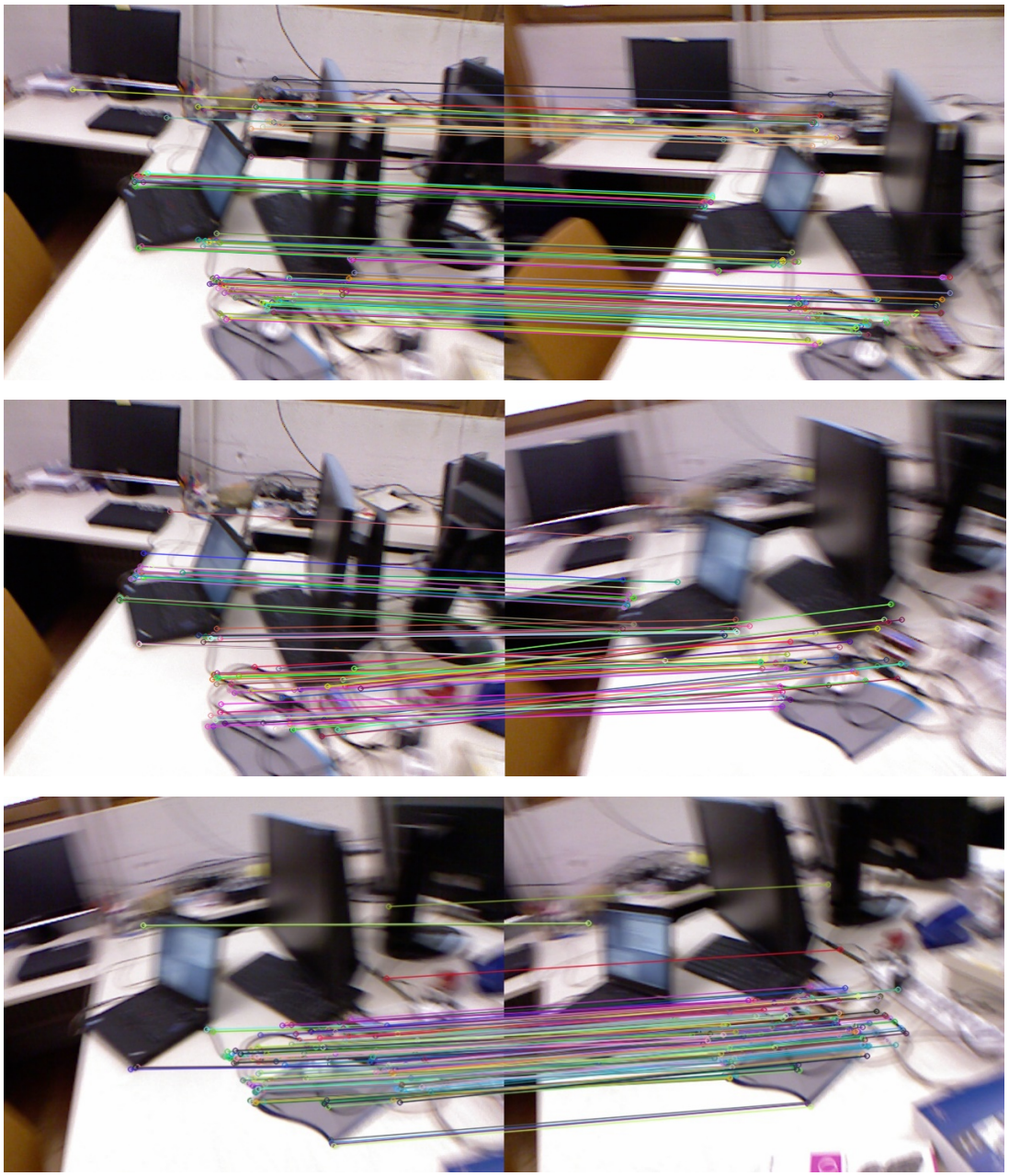


Figure 5.48: Sample images analyzed for fr1/desk sequence

6 Discussion

The experiments described in the previous chapter laid out the success and failures of the proposed work. This chapter discusses and evaluates the results of each experiment methodically.

6.1 Systematic Movements

The purpose of analyzing the proposed approach with systematic movements as explained in Scenario-1 and 2 is to investigate the behavior in the best-case. Scenario-1 and 2 are considered as the best-case scenarios because no camera motion considered in these experiments and the data was collected when the Kinects are moved to a certain position and when they are stationary. On the other hand, Scenario-3 presents freehand movement of Kinects hence can be considered the worst-case compared to Scenario-1 and 2. These experiments help to identify how far the Kinects can slide away without a rotation and up to how far away they are able to maintain a reasonable accuracy in the pose if rotated about Y-axis and translated along X-and Z-axes. These scenarios can be applied when 3-DoF localizing mobile robots in indoor environments whose translation is constrained along X- and Z- axes while able to rotate only about Y-axis.

6.1.1 Scenario-1: Systematic Translation only of one sensor relative to a stationary sensor

Our experiments show that a robot can slide along X axis with respect to a stable Kinect or a helper robot up to 1.5m. However, this distance depends on

the scene they observe and how rich the features extracted from the scene. According to our sparse scene experiment, a robot can move 1.5m with a reasonable accuracy of localization but with the dense scene it is restricted to 1.4m. One reason for this difference would be some low-quality features with incorrect depth values get removed while occluding the scene for sparse scenario which resulted in more accurate pose estimation. We prove that this method performs better than our ACRA method in Scenario-1 experiments by being able to maintain an accurate localization for even longer distances.

If a helper robot is used as the reference for localization, some sort of coordination between two robots can be employed to move the second robot further away along the trajectory. If they coordinate to move the helper robot to a closer position so that their share of FoVs increase, then the second robot can explore further areas in the environment. It is important to relocate the helper robot to an accurate position, otherwise this may affect the global localization accuracy. If a swarm of robots are to be localized and a large area to be explored, then instead of a single helper robot relocates every time, the reference role can be switched among the robots so that a part of the swarm can navigate so that they share their FoVs. Nevertheless, our experiments for translation provide only a systematic analysis and not tested on robots in real scenario. The localization accuracy for such scenarios may be affected by image blur, rolling shutter effects and even by robot's odometry errors. The accumulated error could affect the global localization of robots.

6.1.2 Scenario-2: Systematic Translation and Rotation of one sensor relative to a stationary sensor

Experiment for Scenario-2 allows us to further analyze the performance of our approach taking the rotation into consideration. The turn table experiment proves the accuracy of our method if implemented on mobile robots. The experiment is equivalent to localizing a mobile robot who is freely moving on the floor with translations and rotations. In our experiment, there is a vertical gap between the cameras, and this can be assumed a fixed gap between the two robots. The results show that our method can perform well with the rotations up to 45^0 between the Kinects with only 3^0 average rotation error. However, the translation error is significantly high for both dense and sparse scene experiments. In terms of rotation between the Kinects, our method shows significant improvement over ACRA method which could only support angles up to 25^0. Incorporating depth information in the proposed method helped with the localization improvement. Another reason for our method to succeed over ACRA method would be the ability to extract good keypoints even the images are being rotated significantly. Because we used oriented FAST corner detector and BRIEF feature descriptor by having ORB in our implementation, the algorithm could maintain localization with larger rotations as well.

Scenario-2 experiment considers a scenario with more rotation than translation, but the localization accuracy is still acceptable. When consider the angular errors, the rotation error about Y-axis is negligible for the dense scene and even for sparse scenes the error is minimal for rotations up to 25^0. The percentage accuracy of Y-axis rotation is around 93.5% and 85% for dense and sparse scenes. Y- axis

rotation is important in our analysis because, it is the only rotation that a mobile robot can perform. If the robots are used in these experiments to localize each other, the results could affect with motion blur as explained before.

The results prove that our approach can be used in 3-DoF localization of mobile robots. However, the experiments are systematically analyzed offline and not tested on robots moving in real scenarios which can be more challenging. Nevertheless, the ground truth information obtained for both scenarios might not be the precise values and there could be human errors in estimating the ground truth.

6.2 Freehand Movements

The freehand moving experiments are useful to analyze the impact of our approach in 6-DoF localization of robots including UAVs. Pose estimation conducted for dense and sparse scene laboratory experiments provide a qualitative analysis of the data.

6.2.1 Scenario-3: Freehand movement of one sensor relative to a stationary sensor

The localization accuracy towards Y direction is kept above 85% and there is minimal error in X- and Z- directions for dense and sparse scenes compared to ground truth trajectory. However, the accuracy of estimated trajectory is very much prone to human errors because the obtained error values are relative to ideal ground truth trajectory which was not possible to achieve in these experiments. When moving the Kinect, there could be considerable amount of

drift-off from the ideal trajectory which are not accounted in the final trajectory. In both experiments, the Kinects could maintain the localization without losing shared field of view because the trajectory was relatively short compared to previous scenarios. Also, the average speed of moving the Kinect is very slow, but there were significant delays at the start and end points which are not accounted for speed calculation. However, an analysis with the Kinect speed of 1-2cm/s would not be an adequate measure for the robots who operate in 6-DoF. Further experiments with precise ground truth measurements could help to analyze such scenarios.

6.2.1.1 Qualitative Comparison with TUM Dataset

On the other hand, the qualitative analysis against TUM Benchmark dataset is suitable for more practical scenarios. Since the dataset comes with precise ground truth information this evaluation is more advance than above analysis. Not only that the evaluation against this benchmark sequences provides more thorough evidencing of the robustness of our approach because the data sequences are fairly challenging hence demonstrate more practical usage of our work. The only drawback of this dataset is, being captured using a single Kinect which would have been a more appropriate dataset if comes with sequences captured with two or more Kinects. Since the work proposed here is based on using multiple Kinects, our results might negatively affect with other factors when using single Kinect data such as the reference frame not being updated real time with the change of environmental lighting condition etc.

The analysis against TUM benchmark is considered to be a more genuine evaluation of our method in the usage of AR applications and robot localization. We present a fair evaluation of our approach against single RGB-D odometry. The proposed work here is suitable to use on localization of robots either on 6-DoF or 3-DoF. TUM benchmark analysis supports our evaluation for using on 6-DoF robot localization which can then be used to predict the analysis against 3-DoF localization. Evaluation based on the number of inliers is proven to produce better results than using a fixed number of frames. Based on the pre-predicted nature of the trajectory, a suitable method can apply as the metric for helper robot relocation. When comparing the results analyzed for fr2/xyz and fr2/desk sequences based on the number of frames, fr2/xyz sequence produced better results even with 300 frames offset than fr2/desk which could barely survive the localization even with 50 frames offset. Therefore, when implemented on robot localization and navigation applications it is useful to choose the best metric so that the frequency of helper robot relocation is minimum. Making this choice is easier if the expected trajectory is not random and possible to guess beforehand. According to fr2/xyz analysis, applications where the surveilling robot navigates closer to the helper robot with minor rotations, can relocate the helper robot based on a fixed number of frames or can use a lower inlier threshold as the metric. This way surveillance process can be speedup by reducing the delay in frequently relocating the helper robot.

With the results from fr2/desk sequence we prove that our method can be applied on robots navigate in long trajectories when they use a suitable inlier threshold to coordinate between themselves. The results show that the shape of

estimated trajectory is very much accurate in shape except the accumulated drift in the localization. When the moving robot follows a long trajectory navigating further away from the helper robot, the number of inliers is a more suitable measure to decide on when to relocate the helper robot. This is because the occurrence of localization failure is uncertain, and it entirely depends on environmental features and the speed of the moving robot. However, this can result in more frequent relocations leading to delays in the overall navigation.

Through fr2/desk sequence analysis, we also simulate a situation that a moving robot suddenly comes to a point where it can share the view with a stable helper robot. We prove that our method can perform well in these situations as well.

Referring to the analysis in worst-case scenario according to results from fr1/xyz and fr1/desk sequences, our algorithm could still maintain the trajectory shape up to an acceptable accuracy, except the significant localization offset from ground truth trajectory. We suggest using the number of inliers as a measure to decide on relocating the moving robot or to pass on the reference role to another robot in a swarm because the relocation based on a fixed number of frames could affect more negatively in worst-case scenario.

The errors in the results could be affected by the interference of depth images and not having a dense depth map. As a result of this a significant number of good feature matches could be lost during 3D point cloud estimation due to having zero depth. Therefore, the accuracy of estimated pose depends on the accuracy available depth values at remaining features. However, combining both

depth map and point features in is a good approach we implemented in this work rather than using either of them alone.

None of the above described analysis are conducted on real robotic platforms which could be more demanding. We always considered static environments in our experiments and the proposed work is not being tested for dynamic real-world environments. The scenes used for our experiments are rich in features hence our work has not been tested against environments with featureless planar surfaces which are common in most indoor environments. Therefore, more real-world experiments and analysis need to be conducted to address these scenarios.

7 Future Work and Conclusions

In this section we discuss about suggestions and further developments to be added into the proposed work in terms of improving the accuracy, efficiency and robustness. Later we summarize the conclusions made by this work.

7.1 Future Work

The proposed RGB-D localization approach is implemented and tested systematically but needs further investigation and development before it could be a real practical tool. The analysis presented in this thesis is not sufficient to investigate the behavior of proposed work in challenging real-world applications. Therefore, as the next step, an implementation on robotic platforms would help to understand the behavior of proposed work. This localization approach can be implemented and used more efficiently on holonomic robots because they can easily maintain their FoVs while navigating apart from each other. All the experiments presented in this research are based on analyzing offline data hence the evaluation is lack of real-time experiments. Implementing this approach on a swam consisted of two or three robots employing in a surveillance application would be helpful to evaluate the real-time performance. This way a quantitative analysis of the proposed work can be conducted.

To examine the usage of proposed work in augmented reality applications, a simple AR application can be implemented. An example scenario would be, two users holding two RGB-D sensors pointing at the same scene, while the second user can see in his or her camera RGB image the direction or location where the

first user is pointing at. This can be done by showing the location where the 3D ray from first user's crosshair in the middle of his image appears in second user's RGB image. Implementing such scenarios would help to investigate and improve the accuracy and efficiency of proposed work by testing on further pose optimization methods.

The system can be improved to endure worst-case application scenarios such as 6-Dof localization of fast-moving UAVs and uncontrolled hand-held RGB-D sensor movements in AR applications. In order to survive such worst-case scenarios, the system can be improved by integrating with other methods of localization such as fusing with IMU sensor information.

7.2 Conclusions

The work presented here introduces a novel localization approach for RGB-D sensors to use in indoor robotic and AR applications. The proposed localization approach estimates the pose of a moving or static RGB-D sensor with respect to a fixed RGB-D sensor in the environment as long as they share a part of their fields of view. We evaluated the proposed localization approach qualitatively and quantitatively by conducting systematic experiments and comparing with our previous localization methods. We also analyzed the behavior of our approach when used with publicly available RGB-D benchmark datasets.

We investigated the accuracy of estimated pose when the moving RGB-D sensor is moved with translations and rotations relative to the stationary RGB-D sensor in feature rich and feature less environments. The results prove that our approach can be used to localize a moving RGB-D sensor up to 1.5m away from the fixed sensor with an average translational accuracy of 96.7% and average per centimeter rotational accuracy of 98%. When the sensors are moving together with a rotation component, our method could achieve average rotational accuracy of above 87% over 45^0 angle. The corresponding translational accuracy was 83%.

The comparison of this work with our previous localization approach (ACRA) produced better-quality results due to incorporating depth data and the use of more robust corner detectors and feature descriptors. The results of analysis against ACRA method evidenced that the new method can maintain the accurate localization while enhancing the operating range. The results verified that this method enhances the translation domain by 36% and rotational domain by 80%

over ACRA method. The average translational and rotational accuracy of our method over ACRA method remains almost similar most of the time giving slightly higher accuracy.

We showed our approach can be used in AR applications and localizing robots operate in 6-DoF. Even though our freehand moving experiments does not provide enough evidence, the analysis against public dataset proves the robustness of our approach to be used in such applications. When used on robots our method requires the helper robot to be relocated or moved closer to maintain the localization accuracy. The analysis shows that our method works fairly well with short trajectories and a reasonable accuracy can be maintained over the long trajectories. We showed that a threshold number of inliers is a good indicator for helper robot relocation during navigating on long trajectories. In our analysis we verified that our method works well on the trajectories with up to 0.2m/s translational speed and up to 7deg/s angular speed. We also prove that our method can still work without failing on the trajectories up to 0.4m/s translational speed and 23deg/s angular speed. However, the resulted trajectory was significantly affected by the accumulated error of estimated pose. Incorporating information from sensors such as IMU would help improving the accuracy of estimated pose over long trajectories.

This work presented a detailed analysis of the proposed RGB-D localization approach targeting to use on AR and multi-robotic applications. However, to be used on such applications, it requires some more experiments conducted in real-time on robotic platforms.

8 Appendix A: Microsoft Kinect Sensor (Version 1)

Microsoft Kinect sensors are primarily used in the gaming industry as a Natural User Interface (NUI) for Microsoft X-Box 360 gaming platform to capture 3D perception of humans' motions. Kinect sensor has also become a widely used 3D measuring device in indoor robotics [74, 75], object recognition, 3D mapping [28, 37], SLAM [42-44] and 3D reconstruction [50, 51] due to its low cost, reliability and the speed of measurement.

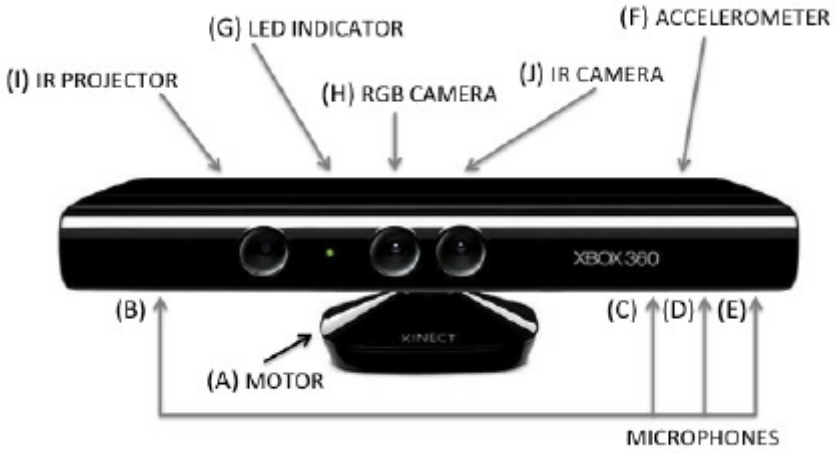


Figure 8.1: Microsoft Kinect with IR Projector, RGB Camera, IR Camera, Accelerometer and Microphone array (Illustration by [76])

Microsoft Kinect provides the color-depth (RGB-D) data having benefits of laser and vision sensing together. The Kinect sensor features a regular monovision RGB camera and a depth sensor, consisted of an infrared (IR) projector and an IR camera pair. This IR camera and the projector are used as a stereo pair which

helps each other to see depth by using infrared vision by triangulating infrared laser points in 3D space.

The Kinect also has an array of four microphones that allows the players to use voice control with noise cancellation when used in gaming. The accelerometer is used for inclination and tilt sensing and for image stabilization while the motorized base is used to rotate the Kinect to track players in gaming.

The technical specifications of Microsoft Kinect version 1 as in [74] and [77] are as below.

Horizontal field of view	57^{0}
Vertical field of view	43^{0}
Frame rate (Depth and color stream)	30 frames per second
Default resolution, depth stream	VGA (640 x 480)
Default resolution, color stream	VGA (640 x 480)
Depth sensor range	1.2m - 3.5m
Physical tilt range	± 27 degrees
Audio format	16-kHz, 16-bit mono pulse code modulation (PCM)

Table 8.1: Microsoft Kinect Specifications

The Microsoft Kinect also has its own System-On-Chip (SoC) special purpose processor made by Primesense which processes the captured images by color and depth cameras. The Kinect system architecture is shown in the below Figure 8.2.

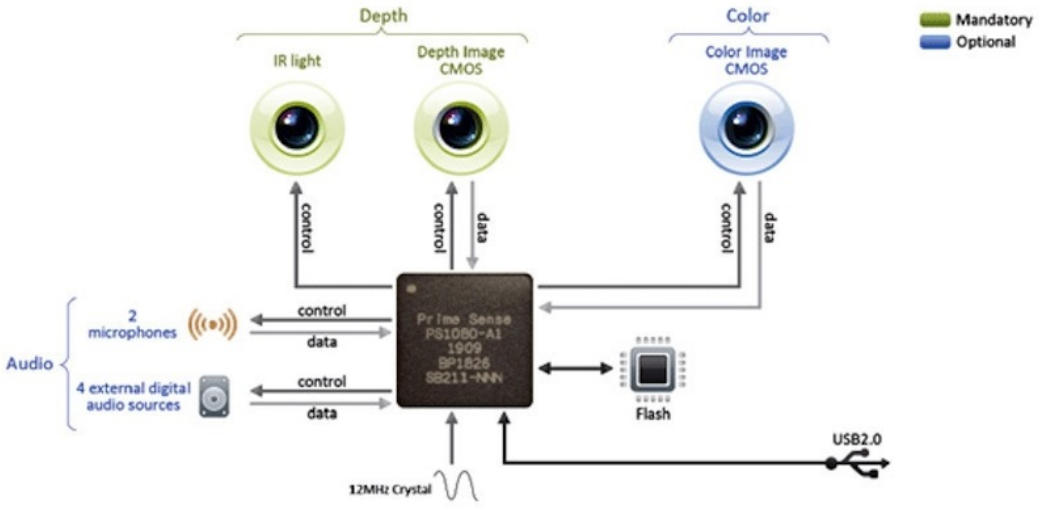


Figure 8.2: Microsoft Kinect system architecture (The image is taken from [78])

Kinect's IR projector sends out a light and dark speckle pattern as shown in below Figure 8.3. The IR camera then captures the projected pattern and for each pixel in IR image, depth is calculated by comparing the local pattern at that pixel with a previously captured memorized pattern from the projector at the same pixel with a selected window. The disparity is taken as the offset from the known depth in pixels and it is refined further with sub pixel accuracy. With the memorized depth and disparity, triangulation is used to estimate the depth at each pixel as given in [79]. These depth data are then correlated to a calibrated RGB camera to obtain RGB image with depth associated at each pixel.

However, being a consumer RGB-D sensor, there are limiting factors of Kinect sensor compared to other mapping specialized sensors. One of those limiting factors is having a smaller field of view compared to other 3D mapping sensors hence limit their usage in mapping. Moreover, the consumer RGB-D sensors for NUI applications are designed in a way that the user always stay within a certain depth range which avoids the chance of inaccuracies of estimated depth information when the user moves away from the accepted range. This is not a crucial factor in gaming however, the range of accurate depth data is an important factor when using these sensors in mapping. Hence the accuracy of depth data being deteriorated when the objects in the scene move further away from the sensor is a significant limiting factor for consumer RGB-D sensors as Kinects when using in 3D mapping applications.

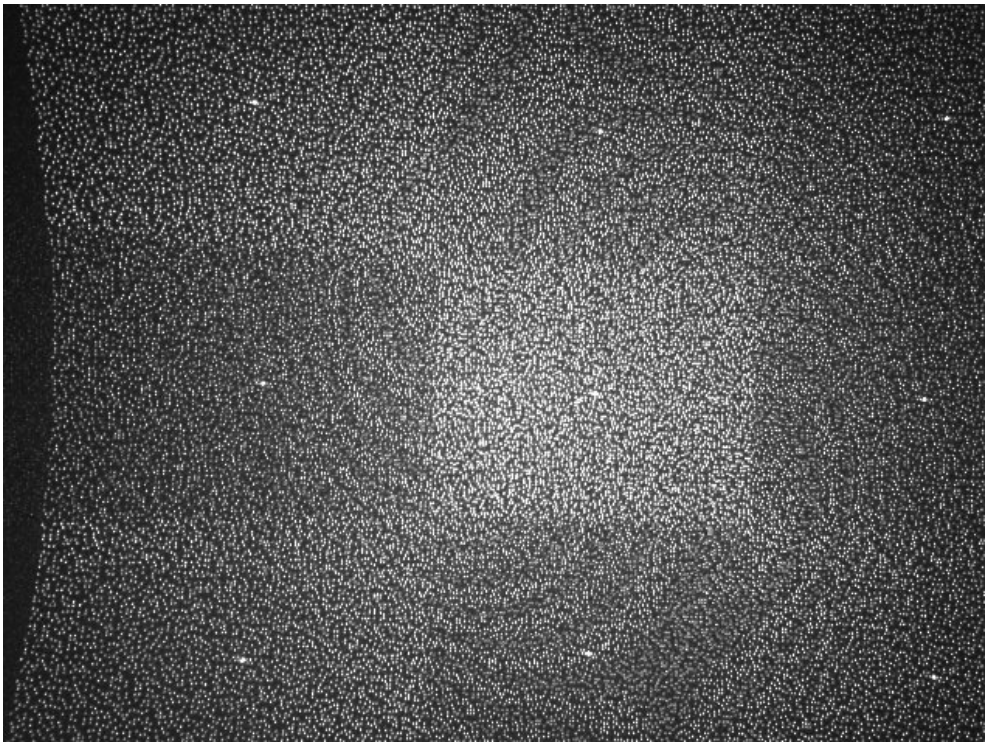


Figure 8.3: Kinect IR speckle pattern (Image taken from [79])

There are several open source software packages available to interface Kinect sensor for robotic applications in different platforms. OpenKinect LibFreenect [80] supports the Kinect with Linux, OS X and Windows. Later on, Microsoft officially released Microsoft Kinect Software Development Kit (Kinect SDK) using Visual Studio 2010 express as their own software package. Kinect SDK enables developers to create applications using Kinect sensor technology on computers running Windows operating system.

In the experiments explained in this thesis, we used Microsoft Kinect for Windows SDK [69] version 1.8 to interface multiple Kinects on a single PC running Windows 7 platform.

9 Bibliography

[1] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A Stable Tracking Control Method for an Autonomous Mobile Robot," in *IEEE International Conference on Robotics and Automation,*, Cincinnati, OH, USA, 1990, pp. 384-389.

[2] B. Barshan and H. F. Durrant-Whyte, "Inertial Navigation Systems for Mobile Robots," *IEEE Transactions on Robotics and Automation,* vol. 11, pp. 328-342, June 1995 1995.

[3] J. J. Leonard and H. F. Durrant-Whyte, "Mobile Robot Localization by Tracking Geometric Beacons," *IEEE Transactions on Robotics and Automation,* vol. 7, pp. 376-382, June 1991.

[4] S. Panzieri, F. Pascucci, and G. Ulivi, "An Outdoor Navigation System Using GPS and Inertial Platform," *IEEE/ASME Transactions on Mechatronics,* vol. 7, pp. 134-142, June 2002.

[5] M. DrumHeller, "Mobile Robot Localization Using Sonar," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-9, pp. 325-332, March 1987.

[6] L. Kleeman, "Ultrasonic Autonomous Robot Localisation System," in *IEEE/RSJ International Workshop on Intelligent Robots and Systems ' (IROS '89) 'The Autonomous Mobile Robots and Its Applications*, Tsukuba, Japan, 1989, pp. 212-219.

[7] L. Zhang and B. K. Ghosh, "Line Segment Based Map Building and Localization Using 2D Laser Rangefinder," in *2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, San Francisco, CA, 2000.

[8] U. Larsson, J. Forsberg, and A. Wernersson, "Mobile Robot Localization: Integrating Measurements From a Time-of-Flight Laser," *IEEE Transactions on Industrial Electronics,* vol. 43, pp. 422-431, June 1996.

[9] J. A. Castellanos, J. M. M. Montiel, J. Neira, and J. D. Tardos, "The SPmap: A Probabilistic Framework for Simultaneous Localization and Map Building," *IEEE Transactions on Robotics and Automation,* vol. 15, pp. 948-952, Oct 1999.

[10] P. Sunhong and S. Hashimoto, "Autonomous Mobile Robot Navigation Using Passive RFID in Indoor Environment," *IEEE Transactions on Industrial Electronics,* vol. 56, pp. 2366-2373, 2009.

[11] B.-S. Choi, J.-W. Lee, J.-J. Lee, and K.-T. Park, "A Hierarchical Algorithm for Indoor Mobile Robot Localization Using RFID Sensor Fusion," *IEEE Transactions on Industrial Electronics,* vol. 58, pp. 2226-2235, 2011.

[12] M. Betke and L. Gurvits, "Mobile Robot Localization using Landmarks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, Munich, Germany, 1994, pp. 135-142

[13] A. M. Sabatini and O. D. Benedetto, "Towards a Robust Methodology for Mobile Robot Localisation Using Sonar," in *IEEE International Conference on Robotics and Automation*, San Diego, CA, USA, 1994, pp. 3142-3147.

[14] C.-C. Tsai, "A Localization System of a Mobile Robot by Fusing Dead-Reckoning and Ultrasonic Measurements," in *IEEE Instrumentation and Measurement Technology Conference. Where Instrumentation is Going (Cat. No.98CH36222)*, St. Paul, MN, USA, 1998, pp. 144-149.

[15] Puneet Goel, Stergios I. Roumeliotis, and G. S. Sukhatme, "Robust Localization Using Relative and Absolute Position Estimates," in *International Conference on Intelligent Robots and Systems*, 1999.

[16] J. Horn and G. Schmidt, "Continuous Localization for Long-Range Indoor Navigation of Mobile Robots," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, Nagoya, Japan, 1995, pp. 387-394.

[17] A. Curran and K. J. Kyriakopoulos, "Sensor-Based Self-Localization for Wheeled Mobile Robots," in *IEEE International Conference on Robotics and Automation*, Atlanta, GA, USA, 1993, pp. 8-13.

[18] D. Nister, O. Naroditsky, and J. Bergen, "Visual Odometry," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR*, Washington, DC, USA, 2004.

[19] T. Sattler, B. Leibe, and L. Kobbelt, "Fast Image-Based Localization using Direct 2D-to-3D Matching," in *International Conference on Computer Vision*, Barcelona, 2011, pp. 667-674.

[20] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision Based MAV Navigation in Unknown and Unstructured Environments," in *IEEE International Conference on Robotics and Automation*, Anchorage, AK, 2010, pp. 21-28.

[21] I. Ohya, A. Kosaka, and A. Kak, "Vision-Based Navigation by a Mobile Robot with Obstacle Avoidance Using Single-Camera Vision and Ultrasonic Sensing," *IEEE Transactions on Robotics and Automation,* vol. 14, pp. 969-978, Dec 1998.

[22] A. J Davison, "Real-Time Simultaneous Localisation and Mapping with a Single Camera," in *International Conference on Computer Vision*, Nice, France, 2003, pp. 1403-1410.

[23] Seokju Lee, Girma S Tewolde, Jongil Lim, and J. Kwon, "Vision Based Localization for Multiple Mobile Robots Using Low-cost Vision Sensor," presented at the 2015 IEEE International Conference on Electro/Information Technology (EIT), Dekalb, IL, USA, 2015.

[24] Andrew J Davison and D. W. Murray, "Simultaneous Localization and Map-Building Using Active Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, 2002.

[25] W. B. Jiirgen Wolft, Hans Burkhardtt "Robust Vision-based Localization for Mobile Robots using an Image Retrieval System Based on Invarient Features," in *International Conference on Robotics B Automation* Washington, DC, 2002.

[26] S. L. Christian Forster, Laurent Kneip, Davide Scaramuzza, "Collaborative Monocular SLAM with Multiple Micro Aerial Vehicles,"

presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 2013.

[27] Z. Z. Yi Feng , Jizhong Xiao, "Heterogeneous Multi-Robot Localization in Unknown 3D Space," presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 2006.

[28] 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, pp. 647-663, 2012.

[29] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-Time Visual Odometry from Dense RGB-D Images," in *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Barcelona, 2011, pp. 719-722.

[30] S. B. W. Ian Mahon, Oscar Pizarro, Matthew Johnson-Roberson, "Efficient View-Based SLAM Using Visual Loop Closures," *IEEE Transactions on Robotics* vol. 24, pp. 1002 - 1014, 2008.

[31] N. Karlsson, E. di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, "The vSLAM Algorithm for Robust Localization and Mapping," in *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2015.

[32] J. V. Miro, W. Zhou, and G. Dissanayake, "Towards Vision Based Navigation in Large Indoor Environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, 2006, pp. 2096-2102.

[33] J. Engel, J. Stückler, and D. Cremers, "Large-Scale Direct SLAM with Stereo Cameras," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, 2015, pp. 1935-1942.

[34] A. Howard, "Real-Time Stereo Visual Odometry for Autonomous Ground Vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice,, 2008, pp. 3946-3952.

[35] Microsoft. (2010). *Microsoft Kinect*. Available: http://www.xbox.com/kinect

[36] Asus. (2018). *Asus Xtion PRO*. Available: https://www.asus.com/3D-Sensor/Xtion_PRO/

[37] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneau, D. Kim, A. J. Davison, *et al.*, "KinectFusion: Real-Time Dense Surface Mapping and Tracking," in *10th IEEE International Symposium on Mixed and Augmented Reality*, Basel, 2011, pp. 127-136.

[38] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. Leonard, "Kintinuous: Spatially Extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, 2012.

[39] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, *et al.*, "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera," in *International Symposium on Robotics Research (ISRR)*, Flagstaff, Arizona, USA, 2011.

[40] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust Real-Time Visual Odometry for Dense RGB-D Mapping," in *IEEE International Conference on Robotics and Automation*, Karlsruhe, 2013, pp. 5724-5731.

[41] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D Mapping With an RGB-D Camera," *IEEE Transactions on Robotics,* vol. 30, pp. 177-187, 2014.

[42] G. Hu, S. Huang, L. Zhao, A. Alempijevic, and G. Dissanayake, "A robust RGB-D SLAM algorithm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, 2012, pp. 1714-1719.

[43] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, 2012, pp. 573-580.

[44] J. Sturm, S. Magnenat, N. Engelhard, F. Pomerleau, F. Colas, W. Burgard, *et al.*, "Towards a benchmark for RGB-D SLAM evaluation," in *RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf. (RSS)*, Los Angeles, USA, 2011.

[45] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Robotics: Science and Systems*, Seattle, WA, USA, 2009.

[46] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, 2014, pp. 1524-1531.

[47] I. Dryanovski, R. G. Valenti, and J. Xiao, "Fast Visual Odometry and Mapping from RGB-D Data," in *IEEE International Conference on Robotics and Automation*, Karlsruhe, 2013, pp. 2305-2310.

[48] Z. Zhang, "Iterative Point Matching for Registration of Free-Form Curves and Surfaces," *International Journal of Computer Vision,* vol. 13, pp. 119 - 152, Oct 1994.

[49] 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, pp. 643-650, April 2012.

[50] D. S. Alexiadis, D. Zarpalas, and P. Daras, "Real-Time, Full 3-D Reconstruction of Moving Foreground Objects From Multiple Consumer Depth Cameras," *IEEE Transactions on Multimedia,* vol. 15, pp. 339-358, 2013.

[51] D. S. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, realistic full-body 3D reconstruction and texture mapping from multiple Kinects," in *IVMSP 2013*, Seoul, 2013, pp. 1-4.

[52] E. J. Almazan and G. A. Jones, "Tracking People across Multiple Non-Overlapping RGB-D Sensors," presented at the 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2013.

[53] G. Loianno, J. Thomas, and V. Kumar, "Cooperative Localization and Mapping of MAVs using RGB-D Sensors," in *IEEE International*

Conference on Robotics and Automation (ICRA), Seattle, WA. USA, 2015, pp. 4021-4028.

[54] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *9th European Conference on Computer Vision, ECCV 2006,*, Graz, Austria, May 2006.

[55] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision,* vol. 60, pp. 91-110, 2004.

[56] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded Up Robust Features," *Computer Vision and Image Understanding,* vol. 110, pp. 346-359, June 2008.

[57] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *European Conference on Computer Vision (ECCV)*, 2010.

[58] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *International Conference on Computer Vision*, Barcelona, 2011.

[59] G. Bradski. (2000). *The OpenCV Library*. Available: http://www.drdobbs.com/open-source/the-opencv-library/184404319

[60] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM,* 1981.

[61] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," vol. 4, pp. 629-642, 1987.

[62] Tom Drummond and R. Cipolla, "Real-Time Tracking of Multiple Articulated Structures in Multiple Views," presented at the European Conference on Computer Vision, 2000.

[63] J.-Y. Bouguet. (2015). *Camera Calibration Toolbox for Matlab*. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/

[64] E. Rosten. (2018). *TooN: Tom's Object-oriented numerics library*. Available: https://www.edwardrosten.com/cvd/toon.html

[65] E. Rosten. (2017). *libCVD - computer vision library*. Available: https://www.edwardrosten.com/cvd/index.html

[66] G. Loianno, V. Lippiello, and B. Siciliano, "Fast Localization and 3D Mapping using an RGB-D Sensor," in *16th International Conference on Advanced Robotics (ICAR)*, Montevideo, 2013.

[67] W. Yii, N. Damayanthi, T. Drummond, and W. H. Li, "Visual Localisation of a Robot with an external RGBD Sensor," in *Australasian Conference in Robotics and Automation (ACRA)*, Melbourne, 2011.

[68] S. Taylor and T. Drummond, "Multiple Target Localisation at over 100 FPS," presented at the Procedings of the British Machine Vision Conference 2009, 2009.

[69] Microsoft. (2018). *Kinect for Windows SDK v1.8*. Available: https://www.microsoft.com/en-us/download/details.aspx?id=40278

[70] M. Luber, L. Spinello, and K. O. Arras, "People Tracking in RGB-D Data With On-line Boosted Target Models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, 2011.

[71] B. Kwolek and M. Kepski, "Human fall detection on embedded platform using depth maps and wireless accelerometer," *Computer Methods and Programs in Biomedicine,* vol. 117, pp. 489-501, 2014.

[72] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "BigBIRD: A Large-Scale 3D Database of Object Instances," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, 2014.

[73] W. Susanto, M. Rohrbach, and B. Schiele, "3D Object Detection with Multiple Kinects," in *Computer Vision - ECCV 2012. Workshops and Demonstrations*, Berlin, Heidelberg, 2012, pp. 93-102.

[74] R. A. El-laithy, J. Huang, and M. Yeh, "Study on the Use of Microsoft Kinect for Robotics Applications," in *IEEE/ION Position, Location and Navigation Symposium*, Myrtle Beach, SC, 2012, pp. 1280-1288.

[75] W. Garage. (2015). *Willow Garage: Turtlebot*. Available: http://www.willowgarage.com/turtlebot

[76] O. Lopes, T. Martins, V. Carvalho, D. Matos, F. Soares, and J. Machado, "Ergonomics and Usability in the Development of a Portable Virtual Gaming Device Applied in Physiotherapy," *Transactions of FAMENA,* vol. 40, pp. 95-106, 2016.

[77] K. Litomisky, "Consumer RGB-D Cameras and their Applications," 2012.

[78] IFIXIT. (2010). *Xbox 360 Kinect Teardown*. Available: https://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/1

[79] K. Konolige and P. Mihelich, "Technical description of Kinect calibration," 2012.

[80] OpenKinect. (2012). *OpenKinect*. Available: https://openkinect.org/wiki/Main_Page