POLITECNICO
DI MILANO

School of Industrial and Information Engineering

Master of Science in Mechanical Engineering

Tracking and Calibration for Spatial Augmented Reality systems with manipulable objects

Supervisor: Prof. GIANDOMENICO CARUSO

Master Thesis of:
JITHIN VARGHESE
Matr. No.: 850461

Academic Year 2017-2018

Table of contents

List of Figures

Abstract

Augmented Reality based on projection, called Spatial Augmented Reality (SAR), is a technology that can produce environments by overlapping virtual and real objects. It has been used extensively for applications related to military, industry, medical, commercial, and entertainment. SAR systems with manipulable objects, requires robust and accurate tracking of texture-less white objects. Optical tracking systems are identified as the best solutions available today. However, due to their high cost, presence of markers and the complex calibration procedure required, the need for the development of a new tracking approach, which solves the fore-mentioned issues, is necessary. Hence, this thesis aims at developing a low cost tracking solution that can meet the challenges of tracking in SAR systems with manipulable objects. An attempt to solve one of the major drawback of existing optical tracking system, the complex calibration procedure, is also performed as part of this thesis.

An approach, which utilizes the advantages of marker based tracking without any markers on the object, has been developed using a flat base tray with markers. Despite of the advantages of low cost and no markers attached to the object, the output of the system has not been good enough yet to use it for practical applications. The disadvantages of the system overcomes the advantages due to the limitations in the calibration procedure and rendering tools available. A new calibration method or a properly designed tray are suggested to solve the issues.

Since the object tracking with RGB-D cameras is identified as a promising tracking approach, an attempt to develop a tracking system based on point clouds analysis has been performed. This solution with an easy calibration procedure is able to track the target object without any markers or tray attached to it. However, the low stability and accuracy of the tracking results makes it unsuitable to implement on SAR systems with manipulable objects. Optimization of the tracking data and utilization of advanced devices are suggested to reduce the fluctuations and to improve the accuracy.

In order to reduce the complexity, effort and time for calibration in SAR systems with optical tracking, a simplified calibration procedure has been developed based on RGB-D sensors. Pro-Cam calibration with Kinect v2 and RGB based region growing segmentation using Point Cloud Library (PCL) enables the identification of calibration parameters and relative positions. Although the methodology saves significant time and effort of calibration, the proposed methodology suffers from the issues due to the absence of verification methods and errors in the final results.

Sommario

La realtà aumentata basata sulla proiezione, denominata Spatial Augmented Reality (SAR), è una tecnologia in grado di produrre ambienti sovrapponendo oggetti virtuali e reali. La SAR è stata ampiamente utilizzata per applicazioni relative a settori militare, industriale, medico, commerciale e di intrattenimento. I sistemi SAR, che prevedono l'uso di con oggetti manipolabili, richiedono il rilevamento robusto e accurato di oggetti bianchi. I sistemi di tracciamento ottico sono la migliore soluzione oggi disponibili. Tuttavia, a causa del loro costo elevato, della presenza di marcatori e della complessa procedura di calibrazione richiesta, è necessario sviluppare un nuovo approccio di tracciamento, che risolva i problemi sopra menzionati. Pertanto, questa tesi mira a sviluppare una soluzione di monitoraggio a basso costo in grado di soddisfare le sfide del tracciamento nei sistemi SAR con oggetti manipolabili. Un tentativo di risolvere uno dei maggiori inconvenienti del sistema di tracciamento ottico esistente, la complessa procedura di calibrazione, viene eseguito anche come parte di questa tesi.

Un approccio, che sfrutta i vantaggi del tracciamento basato su marker senza alcun marker sull'oggetto, è stato sviluppato utilizzando un supporto planare tracciato. Nonostante i vantaggi del basso costo e nessun marker collegato all'oggetto, l'output del sistema non è ancora stato abbastanza buono da poter essere utilizzato per applicazioni pratiche. Gli svantaggi del sistema superano i vantaggi dovuti alle limitazioni nella procedura di calibrazione e agli strumenti di rendering disponibili. Un nuovo metodo di calibrazione o un supporto progettato correttamente sono i suggerimenti identificati per risolvere l problema.

Poiché il rilevamento degli oggetti con le telecamere RGB-D è identificato come un promettente approccio di tracciamento, è stato effettuato un tentativo di sviluppare un sistema di tracciamento basato sull'analisi della nuvola di punti. Questa soluzione con una semplice procedura di calibrazione è in grado di tracciare l'oggetto target senza alcun marker. Tuttavia, la scarsa stabilità e accuratezza dei risultati di tracciamento rende inadatto l'implementazione su sistemi SAR con oggetti manipolabili. L'ottimizzazione dei dati di tracciamento e l'utilizzo di dispositivi avanzati sono suggeriti per ridurre le fluttuazioni e migliorare l'accuratezza.

Al fine di ridurre la complessità, lo sforzo e il tempo per la calibrazione nei sistemi SAR con tracciamento ottico, è stata sviluppata una procedura di calibrazione semplificata basata su sensori RGB-D. La calibrazione Pro-Cam con la segmentazione crescente della regione basata su Kinect v2 e RGB tramite la libreria Point Cloud (PCL) consente l'identificazione dei parametri di calibrazione e delle posizioni relative. Sebbene la metodologia consenta di risparmiare tempo e

sforzi significativi di calibrazione, la metodologia proposta soffre dei problemi dovuti all'assenza di metodi di verifica ed errori nei risultati finali.

Chapter 1

Introduction

Augmented Reality (AR) is a technology that integrates synthetic information into the real world. It provides an interactive experience of a real-world environment whose elements are "augmented" by computer-generated perceptual information, sometimes across multiple sensory modalities, including visual, auditory, haptic, and olfactory. The primary value of AR is that it brings components of the digital world into a person's perception of the real world, not just as a simple display of data, but through the integration of immersive sensations that are perceived as natural parts of an environment. However, the definition of AR is not restricted to particular display technologies, such as a Head Mounted Display (HMD) or hand held displays. The images could be projected onto real objects using projectors. Thus the Spatial Augmented Reality (SAR) which is also known as projection based AR can be achieved. The key difference in SAR is that the display is separated from the users of the system. Because the displays are not associated with each user, SAR scales naturally up to groups of users, thus allowing for collocated collaboration between users. SAR can display on any number of external surfaces inside a place at once. Users are able to feel the physical material in such a way that it provides a passive tactile sensation. Due to the decrease in cost and availability of projection technology, personal computers, and graphics hardware, there has been a considerable interest in exploiting SAR systems in universities, research laboratories, museums, industry, and the art community. The main use of SAR is in as much as military purpose, industrial purpose, medical purpose, commercial purpose, and entertainment. In many situations, SAR displays are able to overcome technological and ergonomic limitations of conventional AR systems.

A correct and consistent registration between synthetic virtual elements and the real environment is one of the most important tasks for AR. The precise, fast, and robust tracking of the observer, as well as the real and virtual objects within the environment, is critical for convincing AR applications. A wide variety of methods like sensor based and vision based are developed to track the objects in AR. Nevertheless, for SAR, the tracking system required to be able to track texture-less plain white objects so as to improve the quality of the projection. Although, user tracking methods are adopted for some applications, the object needs to be tracked for SAR systems with manipulable objects. Thus the tracking and registration of texture-less objects is one of the most fundamental challenges in SAR. Optical tracking with small infra-red markers is the most stable and effective approach used today. However, the cost, presence of markers and complex calibration procedure of this method makes it necessary to put efforts on the development of new tracking solutions. Thus, this study proposes two tracking solutions with the aim of meeting the challenges of tracking in SAR systems with manipulable objects. Similar to tracking, calibration is also a key challenge in SAR, as the SAR systems consists of separate projection and tracking units. The study also proposes an new calibration method that enables the simple and fast calibration in SAR systems equipped with optical tracking.

This thesis starts by considering the results of the previous studies carried out as part of the SPARK project (spa,). The SPARK project aims at developing a SAR platform to support and encourage collaborative creative thinking in the design process by reducing language barriers due to diversity of background and sketching skills of the design team members. It has been schematized as a set of modules that perform specific functions. The SAR module enables the users to build a mixed prototype of the proposed creative concepts, by allowing the combination of 3D shapes, textures, images and sketches on a physical object. Hence, it integrates technologies for visualization, tracking and interaction. The studies carried out and the results obtained as part of the tracking section sets the base of this thesis.

Prior to the implementation of the proposed approaches, an extensive literature review was performed on state of the art AR tracking systems and tracking systems in general with the aim of proposing a new tracking solution for SAR. In chapter 2, a review of SAR systems, explaining the tracking systems used and their drawbacks can be found at the beginning followed by a broad classification and review of state of the art of each tracking methods. A review of the calibration methods used in SAR systems can also be found at the end.

In chapter 3, the optical tracking system used in the SPARK platform is analyzed focusing on the challenges and difficulties. Then the two proposed tracking solutions, flat base tray with markers and the particle filter tracking using PCL are explained in detail. The concept, implementation and analysis of the two proposed methods can be found in sections. This includes the calibration procedure, technical aspects and the analysis of the results. Comparison and discussion of the two proposed methods and comparison of the proposed systems with the existing optical tracking system can be found at the end pointing out the advantages and draw backs.

In chapter 4, the implementation and analysis of a simplified calibration procedure utilizing the depth and color information from Kinect v2 sensor are explained. Similar to the previous chapter, the currently adopted calibration method using cameras for the SPARK system is evaluated focusing on the difficulties. The four steps of the proposed calibration method is explained in sections. The obtained results and the problems found are mentioned and the reasons behind the issues and corrections made is discussed at the end.

Chapter 2

Literature Review

Due to its importance, tracking and registration has been a popular topic among the augmented reality(AR) conferences over the years (Zhou et al., 2008). Researches by (Van Krevelen and Poelman, 2010), (Zhou et al., 2008), (Agarwal and Thakur, 2014), (Siltanen, 2012), (Ahad and Hossain, 2004), (Baillot et al., 2001) etc. have shed light on to possible AR tracking techniques. However, a very few studies focus on the identification of methods and challenges for tracking in spatial augmented reality(SAR).

Among the literature on SAR, the book by (Bimber and Raskar, 2005) gives a collective information on different possible methods for tracking in SAR. The authors consider infrared tracking as high precision and high speed tracking system and marker-based camera tracking as a low cost option. The tracking systems are classified as inside-out and outside-in. Inside-out is a system that is attached to the moving object and tracks relative position with the sensors fixed in the environment. Outside-in is a system fixed in the environment that tracks the emitters on a moving object. According to the authors, marker-less tracking is the most challenging and promising technique.The authors also propose a possible solution that combines GPS with gyroscope and accelerometers for outdoor applications.

Most of the SAR systems focus on tracking objects rather than the user movements. However, systems like (Porter et al., 2010), (Voida et al., 2005) are based on tracking the motion of a body part of the user.(Voida et al., 2005) proposes methods to interact with the object by recognizing gestures or by utilizing hardware devices such as wireless gyroscopic mice. On the other hand, (Porter et al., 2010) tracked the finger of the user for rapid interactive interface prototyping. A colored thimble recognized by a camera enables the tracking of finger as shown in figure 2.1. Even though this system is usable for the fore-mentioned application, the delay and the inability to detect fast motions, makes it unsuitable for many other applications. (Bimber and Raskar, 2005) mentions the possibility of head-tracking for spatial augmented reality. With the introduction of Microsoft Kinect sensors, the body tracking became more popular. Works like (Johnson and Sun, 2013) (Cebulla, 2013) (Benko et al., 2014) (Mast et al., 2017) used body tracking with Kinect for their SAR systems.

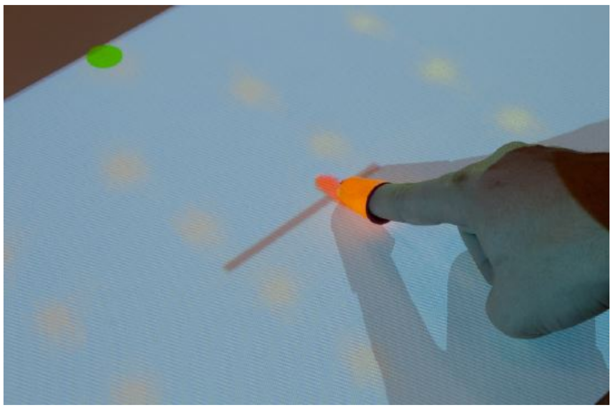
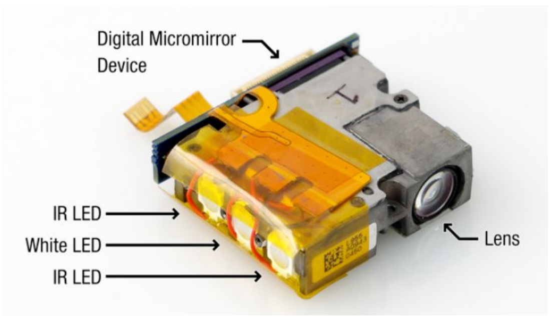
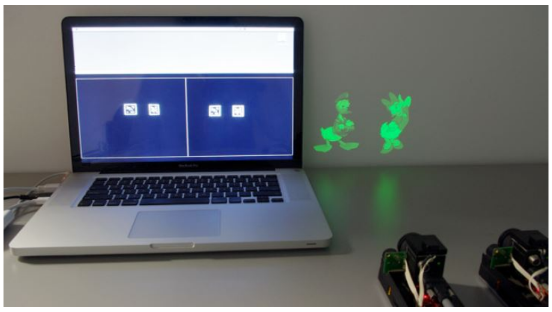


Figure 2.1: Finger tracking using thimble

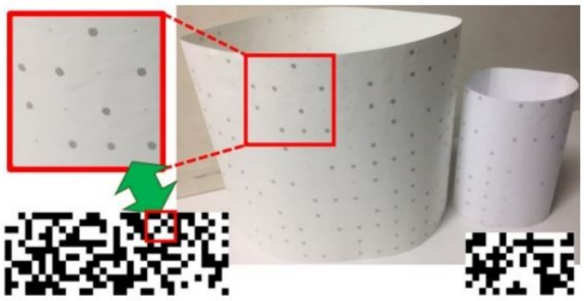
(Calife et al., 2007) combined robotics with SAR. A robot has been used as an interaction tool in the SAR environment. A colored marker-based tracking using web camera and OpenCV library has implemented to recognize the fiducial markers (Legos). The tracking algorithm determines the center of the marker by segmentation of the image using HSV color system and calculates the position using pixel to millimeters ratio after transforming it with respect to the origin.The orientation is determined from the vector between the markers. However, This system only determines X and Y positions of the markers. It also requires colored fiducial markers with high saturation. Whereas, (Zhou et al., 2011) used Intersense black and white fiducial markers. Despite of the fact that the marker-based camera tracking with fiducial markers are widely used due to its lightweight and robust performance, it usually requires good, predictable ambient light. On the other hand, SAR projections appear better with low light and these projections can change the appearance of the markers themselves (Marner et al., 2011). An interesting solution to this problem is implemented by (Willis et al., 2011) using an invisible IR fiducial marker projector. Light sources of a hand held projector is modified and an IR projection channel is added as shown in figure 2.2a. This allows independent visible and invisible content to be combined in a single projected light stream as shown in figure 2.2b. An on-board IR sensitive camera constantly identifies and tracks the position and orientation of all fiducial markers projected using ARToolkitPlus library. Nevertheless, this system solves some issues with visible markers, it suffers from low accuracy and natural light issues. Another solution that can be used is suggested by (Miyazaki and Hashimoto, 2018). The solution uses IR dot markers (Figure 2.2c) implemented by retro-reflective material or IR ink, which are not disturbed by the visible light in marker detection with an IR camera and does not degrade the visibility of the projected images. The marker recognition is performed using depth data and IR images from Kinect RGB-D camera.This system is designed for the projection mapping on non-rigid objects. Projection regions are correctly identified and adjusted for external occlusions and the images can be correctly projected with this method , however the markers implemented by retro-reflective material is not as effective as the author stated. The retro-reflective material does interferes with the projected image as it appears to be reflective to visible light as well. On the other hand the IR ink turned out to be effective for the purpose, nevertheless the volatile behaviour of the ink causes the need for re-implementation of the markers in short periods.



(a) Modified DLP pico projector engine to project both IR and visible light



(b) Projecting visible and IR images in a single stream



(c) IR dot markers

Figure 2.2: Solutions to marker projection interference issue

The projection surface required to be texture-less in order to achieve better quality image projection and to avoid poor marker detection issues of marker based tracking. Therefore, a marker-less tracking approach is developed by (Morikubo and Hashimoto, 2017). In this approach the contours obtained from the monocular image of the object is matched with its 3D shape model having contours and estimates the posture with a linearly predicted initial pose. The correspondence between the 3D shape model and the contour of the monocular image is estimated from the predicted initial pose. Unlike the previous use of this idea, the concept of ICP algorithm has applied to the alignment between the 3D point clouds for the combination of the 3D shape model and the monocular image.This method achieved high accuracy of both the position and the orientation tracking with a tracking speed of 120fps by using the GLSL shader to generate the edge points. However,this method hasn't been tested with objects that are used for practical SAR applications and the accuracy of tracking still needs to be improved.

Optical tracking has become popular in SAR enviroments by the introduction of motion capture systems like Vicon and Optitrack. In the work of (Marner et al., 2009) a six camera Vicon MX motion capture system is used to track the position and orientation of objects. The system is based on infrared markers attached to the objects and to the hand-held tool for interaction. Whereas , (Hartmann and Vogel, 2018) used ten-camera Vicon motion tracking system for real time tracking of a mobile phone and a person's head. (Chan and Lau, 2010) also used infrared optical tracker where it tracks the user as well as the objects. The head of the user is tracked by the markers fixed on a pair of glasses that the user has to wear.(Lindlbauer et al., 2016) also used the same methodology. The user's head is tracked for perspective correction and the devices to enable interaction. A large number of SAR systems have been implemented with Optitrack system due to its high accuracy and fast tracking. For the development of a tangible user interface to provide an interactive workflow for the industrial design process, (Irlitti and Von Itzstein, 2013) used 6 OptiTrack Flex:V100R2 cameras to track the stylus like tangible input tools(Figure 2.3a). Each tool is attached with 4 optical marker balls. Whereas, (Marner and Thomas, 2014) used 8 Optitrack cameras for their kitchen design and interior architecture SAR system. The cameras track the interaction tools and the kitchen cabinet walls where the image is being projected (Figure 2.3b). Other optical tracking systems like ART(Advanced Realtime Tracking:https://ar-tracking.com/) is also is being used (Menk and Koch, 2013). Even though the optical tracking systems provide accurate and fast tracking, requirement of markers attached to the object makes it undesirable for many applications.

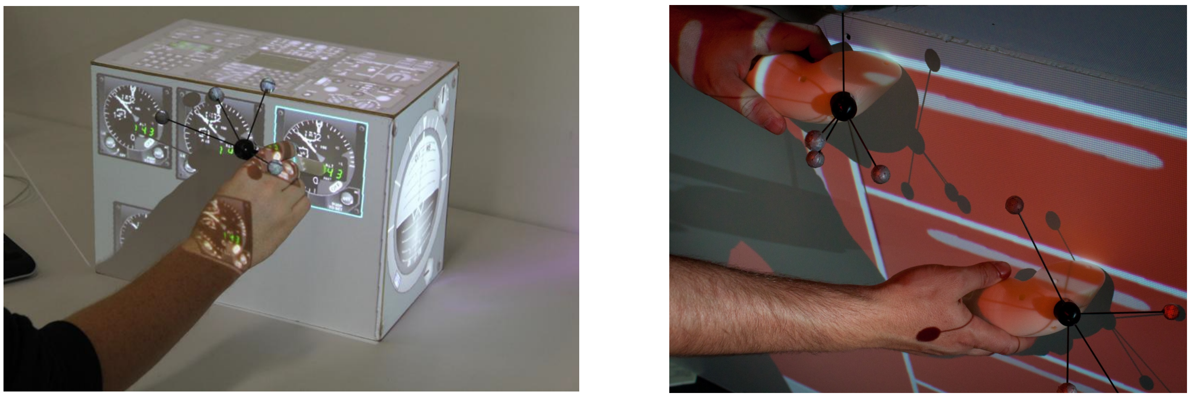

(a) **(b)**

Figure 2.3: Examples of optical tracking using IR cameras and spherical markers

Some SAR systems use two or more tracking systems together to achieve robust tracking. For example, (Ridel et al., 2014) combines optical tracking system with electro-magnetic. In this work the author used Razer hydra electro-magnetic tracking system to track the object with certain shape and a Leap motion optical tracker to track the finger. Even though this system intend to provide robust tracking, the leap motion tracker suffers from limited interaction volume and less stability of tracking.

Attempts have been made to develop tracking techniques that can meet the challenges of tracking in SAR, but no single solution hasn't developed yet. Hence, a detailed Systematic Literature Review (SLR) on 3D object tracking techniques has performed following the SLR protocol suggested by (Rabbi et al., 2012). This enables the identification of available methods of tracking that can be used for SAR. (Dünser et al., 2008) has provided a properly classified resource to identify related references.

A broad classification for AR tracking technique was proposed by (Rabbi and Ullah, 2013). The trackers are classified into three groups namely sensor based, vision based and hybrid. Further level of classification is illustrated in figure 2.4. The optical tracking and vision based tracking techniques are classified in entirely different classes although they are considered as similar by other authors.

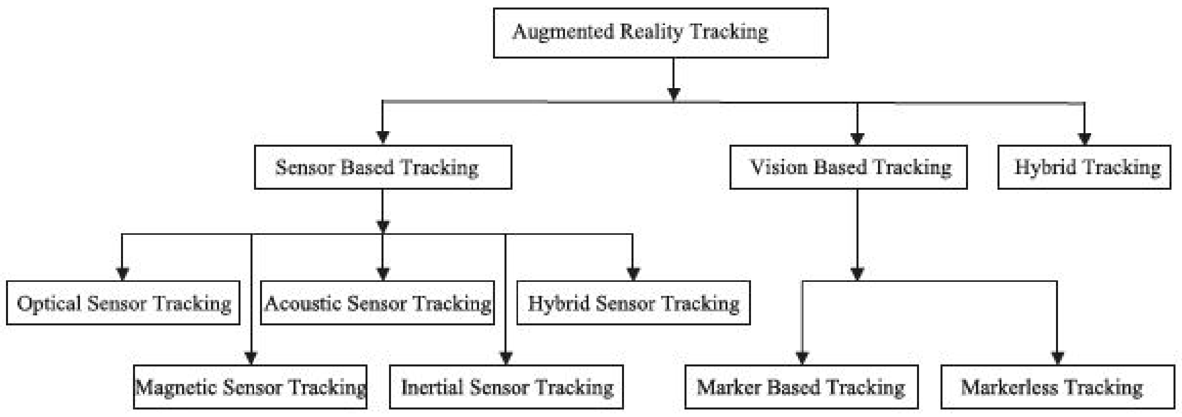


Figure 2.4: Classification of AR tracking techniques

Technological advancement and researches over the years opened the door to new possibilities for tracking in augmented reality. The first HMD by Sutherland (Sutherland, 1968) was first tracked mechanically by motors and ceiling trolley. Later, (Raab et al., 1979) introduced Polhemus magnetic trackers that measures distances within electromagnetic field. Advancements in this technology made it small, compatible and cheaper to implement but is still less accurate and prone to errors than other systems available in the market. Magnetic tracking method is mainly based on sensing of the magnetic field generated by the magnetic source. Usually, a small permanent magnet is used as the magnetic source that doesn't require power supply to maintain the magnetic field hence, wireless tracking can be achieved. A magnetic dipole model is used to estimate the magnetic field. Magnetic tracking is popular in medical devices since it can be integrate to small objects (figure 2.4) and gives absolute tracking measurement with respect to the calibrated tracking reference. A detailed review can be found in (Franz et al., 2014). Magnetic sensors were designed to measure only 5 degree of- freedom (DoF) pose information can be estimated, with the self rotation information missed until (Song et al., 2017) proposed a novel 6-D (3-D position and 3-D orientation) pose detection method, which is based on an opposing-magnet pair system. Two magnets of different sizes with opposite magnetic directions have been combined together to serve as the tracking target. The missing spin rotational information is estimated according to the relative movement between these two magnets. However, since magnetic system measures the position and orientations by identifying the variations in the magnetic field created, external disturbance in the magnetic field can cause large errors especially on the orientation information. The presence of a small electronic device nearby can result in a large deviation from the actual result (Ullah, 2011). Magnetic trackers are even vulnerable to distortion by even the presence of metal that exists in many desired AR application environments (Azuma, 1997). Early attempts to correct the distortions in the magnetic systems were using least-squares polynomial fit, linear look-up calibration, and bump look-up calibration (Bryson, 1992) and also using ultrasonic calibration (Ghazisaedy et al., 1995). Later researches have succeeded at correcting the gross error but there hasn't been any improvement in reducing the mean error to a value that makes it suitable for SAR systems.

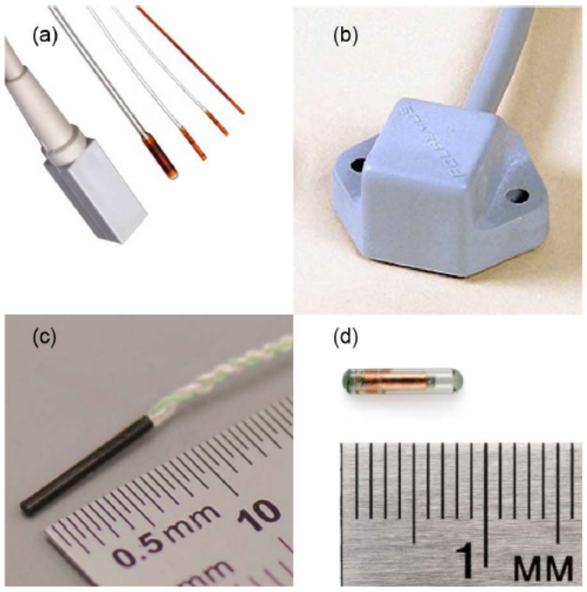


Figure 2.5: Examples of magnetic sensors.(a) Ascension DC magnetic sensors (b) Polhemus AC magnetic sensors (c) Six DoF Sensor for the NDI Aurora AC tracking system (d) Passive EM transponder of the Calypso GPS for the Body system

Accelerometers and gyroscopes provide the possibility to track objects without any source. They are called source-less inertial sensors (Van Krevelen and Poelman, 2010). A mechanical gyroscope is based on the principle of angular momentum that stated that an object rotated at high angular speed in the absence of external moments, conserves its angular momentum (Baillot et al., 2001). The orientation of the target can be computed from the rotational encoder angles. Each gyroscope gives on reference axis in space so at least two gyroscopes are needed to find the orientation of an object in space. The main advantage is that it doesn't require an external reference. However, the small friction between the axis of the wheel and the bearing causes the momentum to be non-parallel to the axis of rotation which in turn causes drift in the direction of wheel axis with time. Due to this problem, the system suffers from various numerical errors but periodic re-calibration or combining with other sensors could alleviate the issue. On the other hand the accelerometer measures the linear acceleration of the object relative to the free fall from which the position can be accurately calculated. In a 3-axis configuration, it can measure the orientation with respect to the direction of gravity. That is the rotations perpendicular to this direction cannot be determined. Magnetometers can be used to measure the yaw rotation which is not possible with accelerometer. The motion of the object produces a pressure of the crystal because of the inertial of the mass. The resulting force, proportional to the acceleration can be evaluated by measuring the voltage. The sensor is light weight and reference free. However, the mixture of forces develops during the motion makes the separation of the acceleration components difficult. In order to solve the issues of inertial trackers, a hybrid sensor tracking method can be used. GPS (Global Positioning System) is a promising technology for long range outdoor AR tracking but it is not suitable for indoor applications due to weakness and blockage of signal. However, GPS combined with inertial sensors have been used for a number of applications. A recent application is found in the work of (Chen et al., 2018) where the author uses the attitude from IMU(Inertial Measurement Unit) and GPS to track the motion of a moving object from a drone. Combination of both the inertial sensors were also used in several cases. For example, (Lang et al., 2002) and (Foxlin, 1996) have used both the inertial tracking sensors for their application. Hybrid sensors available in the market today is a combination of accelerometers,gyroscopes and magnetometers. A few of such devices are show in figure 2.5. Even though these devices are small in size, when it is attached on top of objects with considerably small size, makes them bulky in appearance. An absolute reference is also required which is to be approximated manually. Hence, within these limitations, these sensors could be used in combination with other tracking sensors for SAR systems with large simple shaped objects.

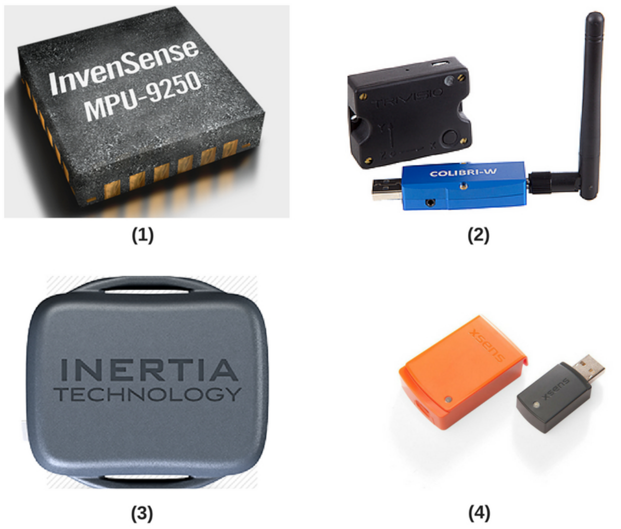


Figure 2.6: Examples of Inertial sensors. (1) InvenSense MPU-9250 Nine-Axis (Gyro + Accelerometer + Compass) MEMS Motion Tracking Device (2) Colibri Wireless IMU (3) ProMove-mini MEMS (4) MTw Development Kit Lite

In general, the basic concept of optical sensor based tracking systems is the use of a camera that uses either visible light or infra-red light. 2D tracking is possible with a single camera but at least two cameras are required for 6DOF 3D tracking (Ullah, 2011). The position and orientation of the object is calculated using the epipolar geometry between the two planes of the images captured by the two cameras. Optical sensor based tracking system has been used for several applications over the years. State of the art optical tracking systems use infra-red lights due to high precision and low latency and is identified as the best tracking system available in the market today for SAR systems. Infra-red cameras uses active or passive (reflective) IR markers in the shape of balls that can be detected by the cameras. A marker body is a cluster of at least four markers rigidly connected but sufficiently separated to allow tracking as shown in figure 2.6. Having marker spheres attached to object can be a disadvantage in some SAR applications. IR flash lights that are usually integrated into the cameras are illuminated to the physical space and the 3D position and orientation of the target is identified with respect to a calibrated reference system using detected marker spheres. The working space can be expanded by free positioning of the cameras which also reduces the occlusion problems. Due to advantages of this system it has used in several SAR applications mentioned before including the SPARK platform. Despite of the functional advantage, currently available systems like Optitrack and A.R.T. IR tracking systems are comparatively expensive. The optical tracking system is robust and accurate is very sensitive to lighting conditions. Parasite reflections and light sources can cause instability and errors. Full or partial occlusions of tracking bodies can also cause issues. To improve robustness, the number of observing cameras could be increased with which increases the total cost. In SAR systems like SPARK, the biggest disadvantage is that tracking bodies have their marker spheres mounted on (fragile) pins of several cm in length. This can provoke issues during manipulation and limits the usability.

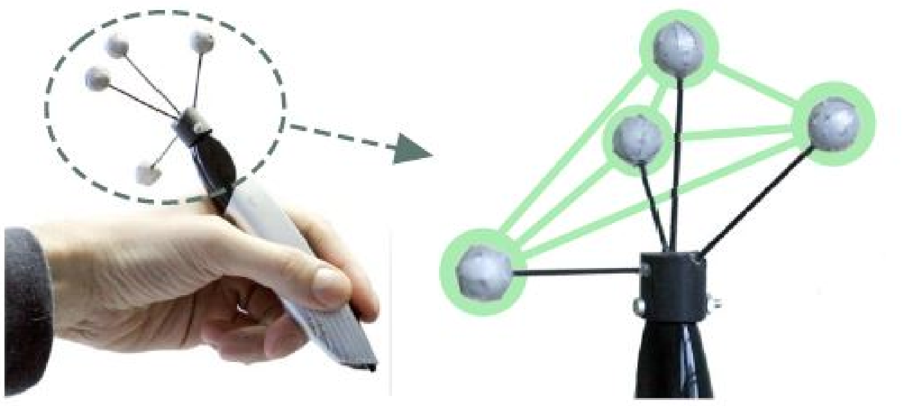
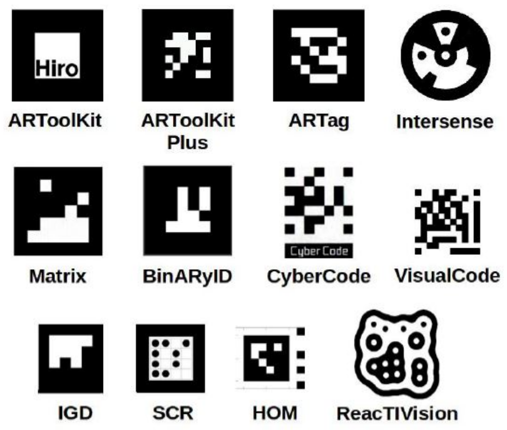


Figure 2.7: IR four marker target

Vision based tracking also known as video based tracking or image based tracking is the most active research area in AR. This method is identified as the low cost option (Bimber and Raskar, 2005). It uses image processing methods to calculate the camera position relative to real world objects. This method is a closed-loop tracking since the digitized images provides a mechanism for bringing feedback into the system (Azuma, 1997). Early computer vision based systems like (Mellor, 1995),(Hoff et al., 1996),(Livingston et al., 1996), (Neumann and Cho, 1996) etc. used fiducial (artificial) markers for tracking. Visual markers are placed in the scene and the specific property of these markers are identified by the camera. (Hoff et al., 1996), (Neumann and Cho, 1996) used black and white markers whereas (Livingston et al., 1996) used coloured fiducial markers but they all failed to produce robust tracking. When the camera moves too fast the image processor has to search the whole image to find the markers to detect. In order to solve this issue, (Naimark and Foxlin, 2002) presented a circular 2D bar-coded fiducial. Examples of several fiducial markers that are proposed in the literature is shown in figure 2.6b. It allows to have thousands of different codes that enables uninterrupted tracking over a large area at reasonable cost. The same year (Prince et al., 2002) proposed a method to track corners of a 2D photo using fast and robust homography calculations. This system has the limitation of working when the camera view contains large blank surface or highly dynamic scene such as crowds. (Kato et al., 2000), (Narzt et al., 2006) and (Dunston et al., 2008) also used fiducial markers for tracking. A highly reliable fiducial marker system under occlusion is presented by

(Garrido-Jurado et al., 2014). A detailed review of fiducial markers can also be found in this paper. A recent system developed by (Laviole et al., 2018) used color tracking algorithm along with feature tracking of OpenCV and pre-existing markers from ARtoolKitPlus. State of the art AR tools like Vuforia (vuf,) and EasyAR (fre,) use images especially having patterns with rich features as targets (Figure 2.7). Despite of the improvements made and resulting advantages of this technology, one of the obvious downfalls that still exits is that the markers have to be seen and cannot be obscured by other object during augmentation.



(a) Example of fiducial markers

(b) Examples of Image target (Stones)

Figure 2.8: Examples of image based tracking targets

Marker-less tracking became popular since (Harris and Stennett, 1990) developed RAPID (real-time attitude and position determination) algorithm that uses selected control points on both objective and profile edges of the 3D object. This system uses single cycle convergence for edge detection. Later many researches developed marker-less tracking systems that uses iterative algorithms to detect point or line features in order to accurately determine the camera pose. According to (Viyanon et al., 2017), marker-less tracking can be performed by two methods. A model based approach that uses prior knowledge of 3D objects in the environment along with their appearance, and the natural feature tracking that utilizes objects in real world as markers by recognizing their characteristics. The model approach uses edge detection for construction of 3D models, in some cases the model is provided to track resemblance in relation to its object in the environment, however, this approach requires more processing power. A real-time multiple 3D object tracking method is proposed by (Kim et al., 2010) by running object detection and feature tracking run in parallel. A foreground thread tracks feature points from frame-to-frame to ensure real-time performances, while a background thread aims at recognizing the visible targets and estimating their poses. The object to be tracked is selected from a captured frame by selecting a region that lies on the object. The second approach can be called feature-based tracking which uses only features already available in the scene. Visual marker-less pose trackers mainly rely on natural feature points (often also called interest points or key points) visible in the user's environment. In order to have an accurate pose determination, this approach requires fast computational time and robustness with respect to changing lighting conditions and image blurring. In order to solve the performance issue with the feature-based approach

(Sánchez et al., 2010) developed a fully parallizable algorithm based on monte carlo simulations on GPU. (Choi and Christensen, 2010) utilized both the methods together to get robust tracking for robotic manipulations. However, these methods have poor performance under occlusion and non smooth camera motion. An algorithm to accurately track camera position solving the aforementioned issues is developed by (Ababsa and Mallem, 2011). This simple and flexible approach estimated camera pose real time by particle filtering frame work with points and lines model-based tracking.3D detection of texture less objects were popular in early computer vision topics,but the complexity of the algorithm forced to use textured objects. A prior CAD model were required for these methods. (Park et al., 2011) proposed a texture-less object tracking with on-line training using depth camera. This method eliminates the requirement of prior object model, since any data for detection and tracking is obtained on the fly, which enhances the depth map. Tracking by synthesis is another promising method for marker-less vision based camera tracking. This system can run at high speed by combining fast corner detection and pyramidal blurring (Simon, 2011). These marker-less tracking algorithms were implemented in well-developed AR development tools like Vuforia and ARToolKit, EasyAR etc. A recent method based on natural features specifically for Augmented reality is proposed by (Li et al., 2017) that uses SIFT algorithm to improve the matching accuracy and improved Lucas-Kanade method for real-time tracking.

Although many tracking methods were implemented based on camera based feature detection, use of depth and IR image information became popular with the introduction of low cost RGB-D cameras. The most existing algorithm for tracking 3D objects using RGB-D data is the Iterative Closest Point(ICP).(Held et al., 2012) used ICP to track hand-held rigid 3D puppets by inputing RGB-D imagery from a Kinect sensor. Robust and real-time performance is obtained by this method, nevertheless the objects are required to have rich color texture and the occlusion by the hands has to be carefully managed through a colour-based pre-segmentation phase. Systems like (Kim et al., 2010) also uses color features to estimate the pose of the object. (Newcombe et al., 2011) used ICP in Kinectfusion for the estimation of pose and alignment. Entire scene structure is reconstructed point correspondences are established using Ray-casting after which estimation of alignment or pose is achieved with ICP. However, a key requirement when tracking with KinectFusion is that the scene moves rigidly with respect to the camera, a condition which is obviously violated when the device is fixed and calibrated with respect to the projector. Particle Swarm Optimization was used by (Kyriazis and Argyros, 2013) to follow the interaction between a hand and an object. This system tracks the motion of the object by the interaction of the hand however it only allows the interaction by a single hand and fails to track fast motions. Another popular algorithm used is the particle filter algorithm. (Azad et al., 2011) used this algorithm to match 2D image edges with those rendered from the model. In contrast to conventional approaches,it can detect arbitrary geometries using accurate 3D polygon model and it doesn't require local features.On the other hand (Choi and Christensen, 2010) added 2D landmark points to the edges. These systems works well with occlusion though,it suffers slow read-back from the frame buffer.

Looking at recent researches on tracking, a tendency towards the use of point cloud based tracking using RGB-D cameras can be observed. Availability of low cost devices like Kinect and powerful libraries like PCL provide the possibility of easy development and implementation of tracking systems for different applications. A number of approaches like (de Figueiredo et al., 2013) (Held et al., 2013),(Yuheng Ren et al., 2013) etc. utilized the tracking algorithms on point cloud data from RGB-D cameras like Microsoft Kinect, ASUS Xtion etc. (Kyriazis and Argyros, 2013) developed an approach that solves the issue of occlusion caused by the hand while interacting with the object using single actor hypothesis. Works by (Ren et al., 2017) (Akkaladevi et al., 2016) (Choi and Christensen, 2013) etc. used tracking functions in Point Cloud Library (Rusu and Cousins, 2011) for the implementation of their tracking approaches. However, all these approaches were developed for robotics and computer vision applications. No such systems were implemented in SAR environments to track non-static objects.

In general, the main challenge of an AR tracking system is to produce fast and accurate tracking with minimal efforts and costs in the settings and changes in the environment (Rabbi and Ullah, 2013). According to (Bimber and Raskar, 2005) precise, fast, and robust tracking of the observer, as well as the real and virtual objects within the environment is critical for a convincing AR application. The tracking systems that are used even today fails to satisfy these criteria. In his paper (Azuma, 1993) mentioned that Augmented Reality demands much more accurate tracking than the virtual environments. According to the aforementioned work, an AR tracking system must be accurate to a small fraction of degree in orientation and a few millimetres in position. The combined latency of the tracker and the graphic engine must be very low. I.e. The time delay between the measurement by tracker and the appearance of virtual object on desire position of the environment must be very low in order to avoid the visual-kinaesthetic and visual proprioceptive conflicts. These conflicts can cause motion sickness (Pausch et al., 1992). For outdoor applications the tracking system is required to work at long ranges. In addition to these requirements, a perfect calibration of the devices are necessary to avoid misalignment in the AR output (Azuma, 1997). The transformation between the tracking device's frame of reference to user's viewing coordinate system is the critical calibration measurement. Once calibrated, some techniques give absolute measurements (electromagnetic sensors) in which further transformations are not required but some techniques makes the process even difficult by reporting relative measurements (Gyroscope). Unlike the traditional HMD and hand-held displays, Calibration process is much more complicated in SAR. The calibration in SAR systems deals with parameter estimation, geometric registration and intensity normalization (Bimber and Raskar, 2005). (Bajura and Neumann, 1995) mentioned the typical errors due to incorrect calibration and tracking.

Projectors in early SAR systems are calibrated manually(Zhou et al., 2011)(Raskar et al., 2001). It is performed by moving a projected cross-hair in the projector image-space so that its center coincides with the known fiducials.The 3x4 perspective projection matrix and its decomposition into intrinsic and extrinsic parameters of the projector are computed using Matlab. This process is cumbersome as the number of projectors increases. Later (Lee et al., 2004) developed an automated projector calibration method using structured light patters. In this method, light sensors are embedded in the target surface and Gray-coded binary patterns are projected to discover the sensor locations. Then the image is prewarped to accurately fit the physical features of the projection surface. (Nakamura et al., 2012) introduced a free-calibration projector-camera system for spatial augmented reality with a planar surface. Although this method can project a pattern on a moving textured planar surface using an uncalibrated projector-camera system,the process is slow and cannot be used for non-planar surfaces. The method proposed by (Moreno and Taubin, 2012) provides an accurate and robust calibration that is simple implement. This method uses a full pinhole model including radial and tangential lens distortions to describe both projector and camera behaviors and computes sub-pixel resolution 3D point projections from uncalibrated camera images. An improved process of automatic multiple projector calibration is proposed by (Smith et al., 2013). The technique uses larger surface area photo detector to locate sub-pixel position and has demonstrated improvements in resolution of pixel measurement over that achieved using a point sized photo detector and the traditional Gray-code projection sequence. However, the maximum pixel size is limited to the area of the photo detector and algorithm is sensitive to short term ambient light level fluctuations. SAR systems with camera based tracking follows the automatic calibration method using cameras, and the same camera is used for tracking objects in the environment. (Jones et al., 2014) introduced a new method of calibration using Kinect RGB-D cameras. This approach utilizes gray code patterns to get camera-projector pixel correspondences and Kinect's depth image to obtain 2D to 3D point correspondences. RANSAC (random sample consensus) algorithm is used to increase robustness. Singular value decomposition and pairwise iterative closest point (Newcombe et al., 2011) methods enables the calibration of multiple projector-camera units. (Benko et al., 2014) used this method to setup the SAR system with 3 cameras and 3 projectors. All the above mentioned works utilized the same calibrated devices for tracking as well as projection. However, for SAR systems with a tracking system other than camera based requires the knowledge of relative position and orientation of both tracking and projector coordinate systems. This process is challenging and time consuming and sometimes error prone. There is no standard procedure available to perform the calibration of the whole SAR system as each system differs in the tracking methodology.

Chapter 3

Tracking

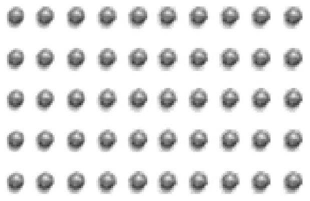
The study proposes two solutions for the tracking in SAR considering the requirements of the SPARK platform and requirements of SAR systems in general that have non-static objects. This chapter deals with the evaluation of the current best method and a detailed explanation regarding the development and analysis of the proposed tracking solutions.

3.1 Evaluation of tracking in SPARK platform

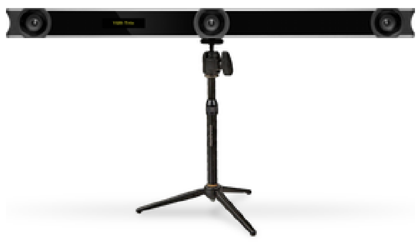
The SPARK platform utilizes the optical tracking system as it is the best tracking system available today for SAR systems with non-static objects. Previous attempts made during the SPARK project to implement tracking systems with inertial and magnetic sensors and the literature review performed for this study proves the same. The platform is equipped with OptiTrack motion capture system. The idea is to track non-static objects white texture-less objects. Hence, 3 mm facial markers (Figure 3.1b) are used to create a rigid body for the cameras to track. Since the markers are small compared to the size of the object, it doesn't interfere significantly with projection. However, the small size of the markers makes it easily occluded by the hands while manipulating. Increased number of cameras can solve occlusion issue as the other cameras can view the markers even though it is masked from one camera during manipulation. Hence, six OptiTrack Flex 3 (Figure 3.1a) cameras were used in this platform.

The number of cameras has a significant effect in the total cost of the system. The optical tracking system is comparatively expensive than other approaches due to the high cost of cameras and markers. This is the reason why only six cameras are used for the project, even though more number of cameras improve tracking on occlusion. However, a combination of six cameras add around 8000 € to the total cost of the SPARK platform. The Flex 3 cameras are of 0.3 MP (640x480) resolution and can capture at 100 fps. It has a latency of 10 ms which is acceptable for this kind of application. More advanced cameras are also available with higher cost. The cost is the main disadvantage of the system. The 3 mm facial markers used is also expensive. A set of 50 markers costs around 85 €.

(a) OptiTrack Flex 3 Cameras



(b) 3 mm facial marker set



(c) OptiTrack V120 Trio

Figure 3.1: OptiTrack motion capture devices

Although the markers are small, its presence is not negligible while projecting images on to the object. Hence, an attempt to use invisible infra-red ink (Miyazaki and Hashimoto, 2018) is made. It is found that, the invisible infra-red ink is effective than the facial markers. The presence of the marker is no visually detected while projecting images. Nevertheless, the infra-red ink is found to be highly volatile and re-implementation of the markers is required at certain intervals of time. This adds maintenance effort and costs to the requirements of the system.

The setup and calibration of the system is highly complex and time consuming. The fixing and calibration of the cameras alone require significant human effort. A predefined procedure suggested by the manufacturer is to be done carefully in order to calibrate the cameras accurately. Adding projectors to the setup requires further complex procedure. The details of the projector calibration can be found in chapter 4. OptiTrack V120 Trio (Figure 3.1c) is installed to avoid issue with calibration and fixing of the cameras. It is basically a bar with 3 motion capture cameras. Since the position of the cameras are already known, the calibration of camera system can be avoided. The fixing of the device is also easier. Nevertheless, this device costs around 4000 € and could be still expensive for some design company. Although, this new device avoids calibration efforts, it may cause occlusion issues since the only 3 cameras are used and are placed close to each other. Considering all the above mentioned issues, efforts must be put into the development of an alternative tracking solution.

3.2 Flat base tray with markers

Marker-based tracking or image based tracking is a popular tracking method in AR. Popular AR platforms like Vuforia (vuf,) and EasyAR (fre,) have efficient image tracking algorithms. It is identified as the least expensive solution for tracking. However, textures or markers reduces the quality of the projection and projection itself causes changes in the texture which results in poor recognition of the markers by the camera. Manipulation of the objects causes occlusion to the markers and the camera fails to track the marker when the object moved fast. Hence, it is suggested to use texture-less objects for SAR systems that makes the use of image based methods unsuitable to track objects in SAR. Nevertheless, a possible solution has identified to utilize the image tracking for non-static objects. The proposed method will be explained in detail in the coming sections.

3.2.1 Concept

From the work of (Zhou et al., 2011), it can be understood that it is not necessary that the marker needs to be on the object. In the fore-mentioned work, the authors placed markers around the object on which the image is being projected (Figure 3.2), so that the markers doesn't interfere with projection and to avoid poor recognition of the markers. However, the object is static in this application. Similar idea can be utilized for non-static objects by using a flat base tray with markers. Instead of tracking the object itself, the object can be placed on the tray and the tray can be tracked using image feature recognition algorithms.

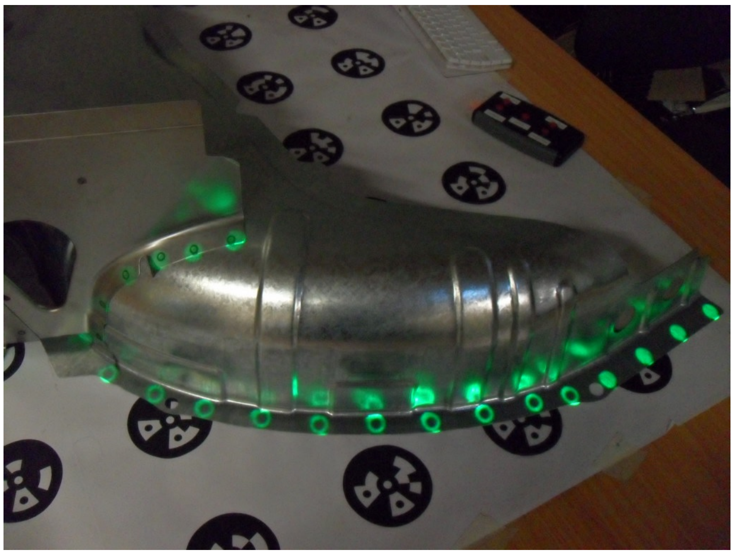


Figure 3.2: Projection of multiple welding spots beside the markers

Since the object is being manipulated by users, the size of the tray should allow easy manipulation of the object as well as the recognition of the markers. The markers should be placed in such a way that the occlusion by the object and the hand during manipulation doesn't affect the tracking. Any kind of markers can be used however, black and white coded ring markers or ARToolKit markers are suggested to be used as it is small and can be placed anywhere over the tray. Since the marker should be small in order to avoid occlusion by the object, images with features cannot be used. An example of the tracking tray prepared is shown in figure 3.3.

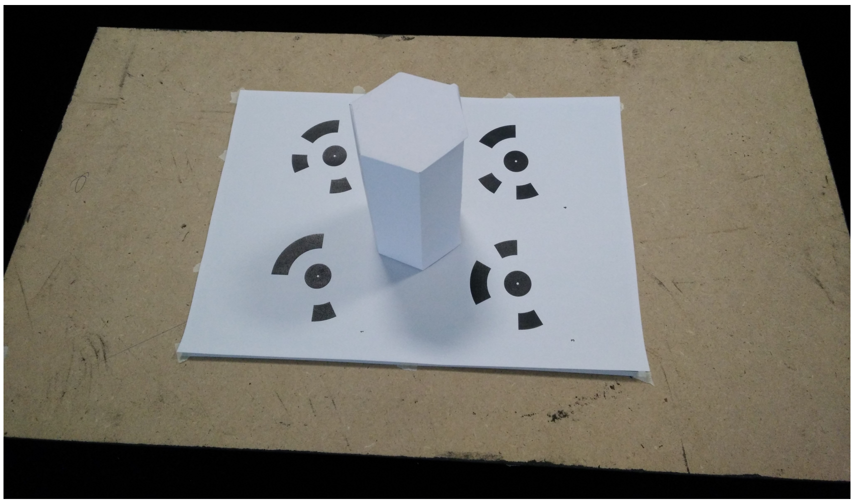


Figure 3.3: Example of tracking tray

3.2.2 Setup and Calibration

The setup consists of a projector to project the images on to the object and a camera to track the markers on the tray. Like every SAR systems, the setup needs to be calibrated in order to project the images at the desired positions on the object. The calibration gives the intrinsic parameters of the projector and the position of the projector with respect to the camera.

A SANYO XGA projector with a resolution of 1024x768 and a Logitech web camera with a resolution of 2592x1944 is used for the setup. The camera is attached to the projector as shown in figure 3.4 in order to get better calibration results.

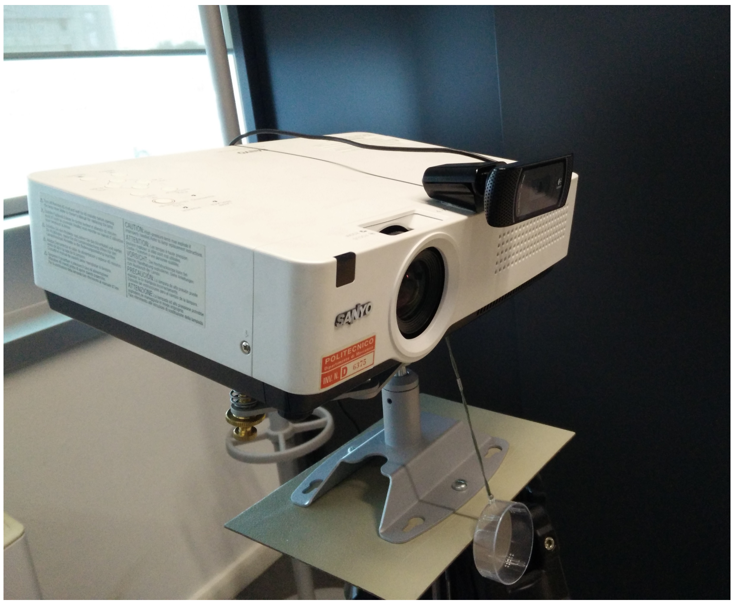


Figure 3.4: Camera-Projector arrangement

Although different Camera-Projector calibration methods can be used for the purpose, the projector-camera calibration method for the SPARK platform is applied in this study due to the availability of the open source calibration application and previous successes in calibration of similar setup. Open-source software scan3d-capture (pro,) is used to get the calibration parameters and a specially developed python script is used to verify the calibration. A 7x10 checker board is used for the purpose. The details of this calibration method is explained in chapter 4 calibration.

Although Vuforia is the most popular AR platform, it doesn't support windows standalone builds. Hence, it cannot be used to develop SAR system. Therefore, EasyAR is used for this study as it supports windows standalone builds. Since the EasyAR platform is being used to implement the AR system, prototype of a hexagonal cylinder which has its prefab already available is selected as the object. The tray is prepared with black and white coded ring markers and the object is fixed at the center of the tray as shown in figure 3.3.

3.2.3 Principle and Algorithm

The recognition and tracking of the markers are done by EasyAR image tracker. Image Tracker is able to recognize and track image targets by analyzing the contrast based features of the target that are visible to camera. Since EasyAR is not an open source platform, the algorithm for recognition and tracking is confidential. However, since the recognition is improved when there are more vertices or lines in the high-contrast image, the algorithm can be considered similar to SIFT (Scale-Invariant Feature Transform) (Lowe, 2004).

The target image is to be loaded to the tracker before hand. The algorithm first identifies key locations in scale space by looking for locations that are maxima or minima of a difference-of-Gaussian function. Each point is used to generate a feature vector that describes the local image region sampled relative to its scale-space coordinate frame. The resulting feature vectors are called SIFT keys. The feature extraction can be computed very efficiently by building an image pyramid with re-sampling between each level. Once the features are extracted, keypoint matching is performed using Euclidean distance between descriptors' vectors and a distance ratio, namely ratio of closest neighbour distance to that of the second-closest one, that can be checked against a threshold to discard false matches. More information on the algorithm can be found in (Lowe, 2004) and some applications are described in (Hu et al., 2008) and (Zhou et al., 2009). The process of tracking is represented in figure 3.5. The features of the target image and the features of each frame of the video feed is extracted and the feature vectors generated are matched to track the image.

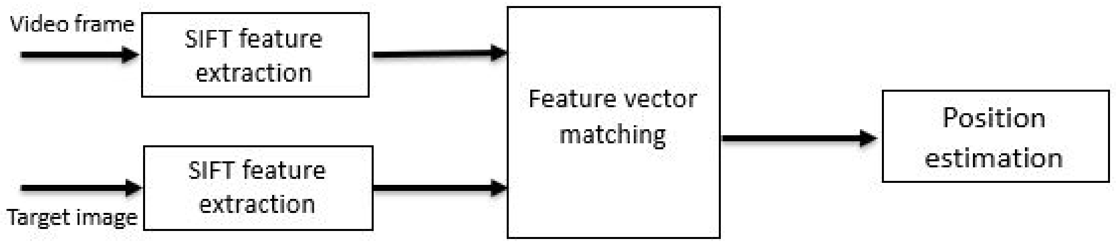


Figure 3.5: Image tracking process

3.2.4 Registration and Rendering

The development of the SAR scene is done in Unity 3D with EasyAR package. Apart from the AR camera which is the camera used to track, a projector camera is added to render the scene from the point of view of the projector (Figure 3.6). The camera is transformed to the projectors position with respect to the camera using the output of the calibration performed.

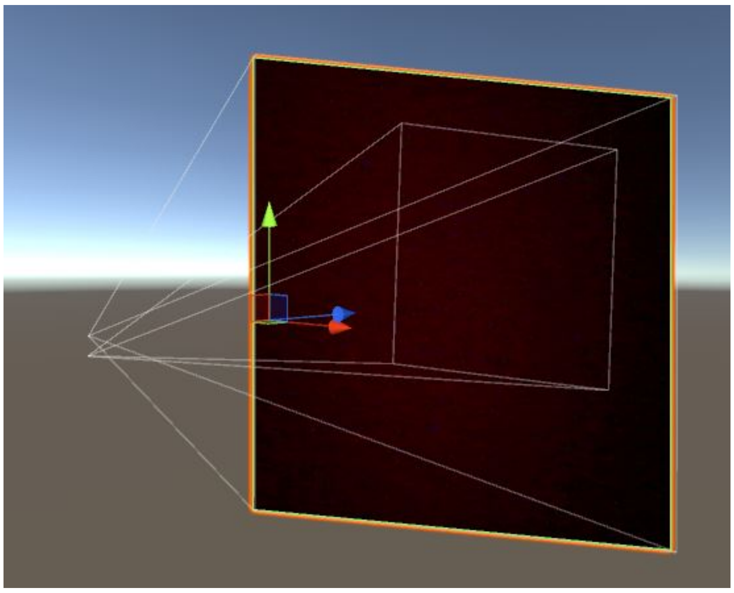


Figure 3.6: Projector and render cameras

For conventional hand held displays or HMDs (Head Mounted Displays), the rendering of the AR scene is performed by overlaying the virtual object on the video feed. In Unity 3D, the video feed is being displayed at the far plane of the render camera and the video frame at this location is processed as an image for feature extraction and matching. Hence, the output of the tracker is a position where the virtual marker overlays the real one on the far plane of the render camera (Figure 3.6) which is also called as the reality plane. Initially, the virtual object is placed at the center of the markers similar to the real object. Nevertheless, the scale of the object will be different since scale in Unity environment is different from the real world. Hence, while running the scene, the virtual object will be transformed to a position where it overlays the real object on the reality plane by homography. Since the overlaying is based on homography but not based on real world coordinates, this cannot produce a SAR scene. Hence, the calculation of the real position and scale using the homography of the camera is performed.

By knowing the ratio of the height of the virtual object to the height of frustum plane at which the center of the virtual object lies, the height of the frustum plane on which the scale of the virtual object is the same as the real one can be found. Once the height of this second frustum plane is found, the position of this plane can be calculated using the distance to height ratio of the camera frustum. Hence, the location of the frustum plane where the center of the real object lies is obtained. This doesn't give the correct position of the center of virtual object so that it perfectly overlaps with the real one. In order to overlay the virtual object on to the real one, the position of the object's center in the plane of interest has to be found. This is done with help of ray-casting feature of Unity 3D. A ray is set to originate from the camera origin and pass through the center of default virtual object. A plane with a mesh collider (Figure 3.7) is created at the previously calculated plane position. The point where the ray collides should be the real world position of the center of real object.

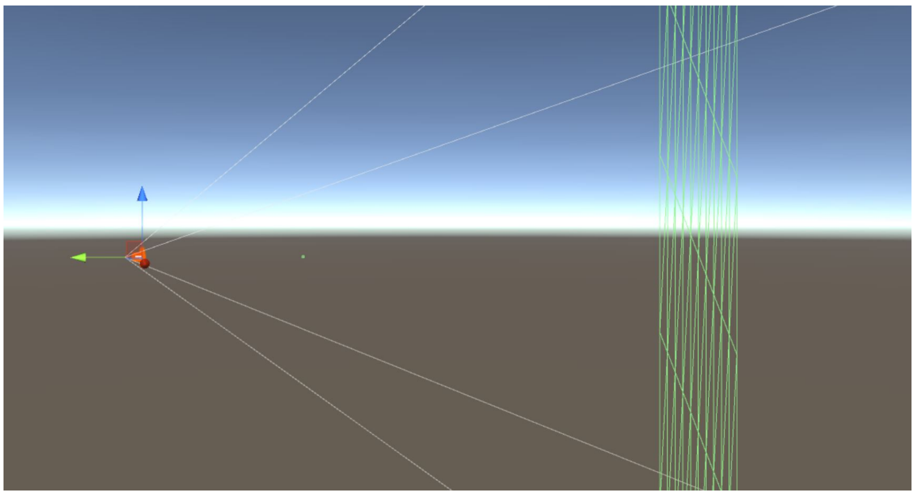
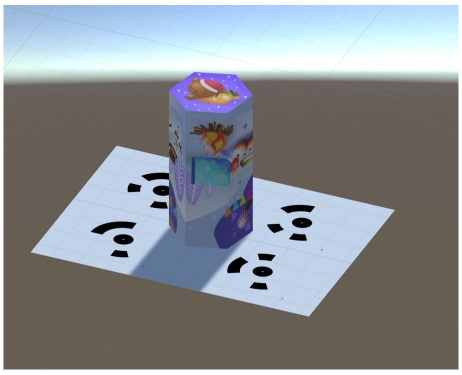


Figure 3.7: Plane with collider to obtain real world position

3.2.5 Results and Inferences

The SAR scene is built and run to test. Since the camera is positioned on top of the projector and the projector is placed at a small angle with the respect to the surface, poor initial recognition of the markers is observed. The tray is required to be lifted and positioned at angle facing the camera in order to detect the markers. Once the marker is detected, the camera keeps tracking the markers unless the tray is moved fast. Hence, the system suffers from poor initial detection and loss of tracking on fast movements. The virtual object overlays the real one with slight errors in scale and position. This could be due to the approximations taken during the calculations. Manual adjustments can correct these errors. The result obtained after fine adjustments are shown in figure 3.8.


(a) Virtual object and markers

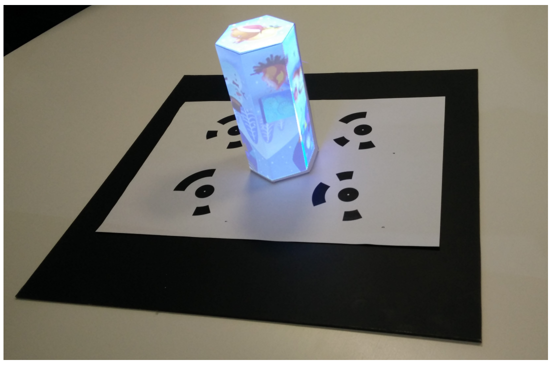

(b) Virtual object projected on to real object

Figure 3.8: Virtual and real SAR scenes

In order to solve the issues with initial detection, the camera can be placed at different position where the markers are detected easily. However, the calibration method used doesn't not allow the camera to be positioned far from the projector. As the camera moves away from the projector, the error in the calibration result increases. The ideal position where the camera can detect the markers perfectly doesn't succeed the calibration. The use of small markers also causes poor detection and loss of tracking sometimes. The presence of tray is a disadvantage considering the comfort of the user. It makes the manipulation of the object difficult. Hence, either a detailed study to solve the issues has to be performed or other tracking solutions has to be considered.

3.3 Particle filter tracking using PCL

From the literature review performed, it is understood that the point-cloud based tracking is a promising tracking solution and a large number of researches are performed in this methodology. However, such solution is neither implemented nor mentioned for any SAR systems. As the current tracking solutions for SAR systems are expensive and challenging to implement, point cloud based tracking with low cost RGB-D cameras provide easy and low cost tracking options. In this study, an attempt is performed to implement the particle filter tracking approach using point cloud library and Kinect V2.

3.3.1 Concept

In order to recognize and track a marker-less plain white 3D object, conventional computer vision methods that uses colors or textures alone are not suitable. Hence, the possibility of utilizing both the color and depth data from and RGB-D sensor has considered. A popular tracking approach available is the particle filter tracking using point-cloud libraries. A low cost RGB-D sensor like Kinect v2 can be used to extract depth and color information of the environment in real time. This data is then used in point cloud library to recognize the target object and to track it. A properly scaled point cloud of the target object is prepared beforehand using PCL functions and this data is compared with the live streaming point cloud data and

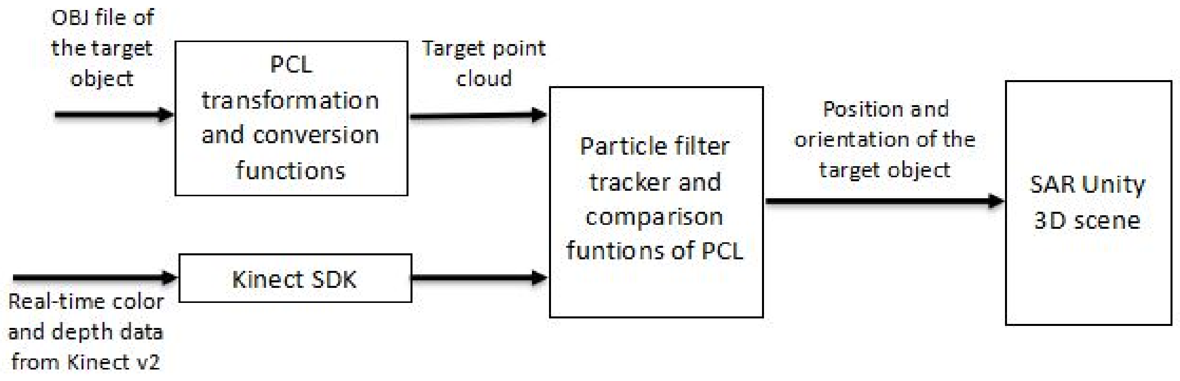

OBJ file of the target object
PCL transformation and conversion functions
Target point cloud
Particle filter tracker and comparison funtions of PCL
Position and orientation of the target object
SAR Unity 3D scene
Kinect SDK
Real-time color and depth data from Kinect v2

Figure 3.9: Concept diagram of PCL particle filter tracking methodology for SAR

the cluster of point cloud that matches with the target is being tracked. The particle filter algorithm is used to obtain real-time performance. The tracking data can be sent to Unity 3d and the custom prefab in the scene transforms relative to the Kinect sensor origin. As the origin of both tracking and projection units is the Kinect sensor origin, no coordinate transformation is required. A diagram showing the general concept of this methodology is shown in figure 3.9. Detailed explanation of the implementation and principle of operation can be found in the following sections.

3.3.1.1 Kinect V2

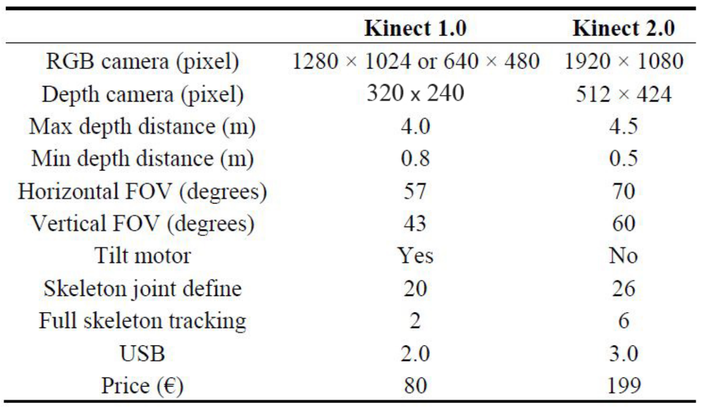
In the field of computer vision, RGB-D cameras have huge impact on the research related to robotics and image processing since it can provide dense depth estimations together with color images at a high frame rate. Kinect is one of the common and less expensive RGB-D cameras available in the market. With the introduction of first generation Microsoft Kinect device (Figure 3.10a), research fields such as 3D reconstruction, Simultaneous Localization and Mapping (SLAM), gesture and object recognition, bilateral filtering has been pushed forward. A Kinect sensor is made up of an RGB camera, a depth sensor, an IR emitter, and a microphone array, which consists of several microphones for sound and voice recognition. The Kinect drivers and software, which are either from Microsoft or from third-party companies, can even track and analyze advanced gestures and skeletons of multiple players. The camera detects the red, green, and blue color components as well as body-type and facial features. With the release of the Microsoft Kinect v2 (Figure 3.10b) which uses a new Time-of-Flight (ToF) method that improves accuracy and quality, an extensive use of these devices can be observed. The new version has a wide-angle time-of-flight camera, and processes 2 gigabits of data per second to read its environment. The depth sensor contains a monochrome CMOS sensor and infrared projector that help create the 3D imagery throughout the room. It also measures the distance of each point of the player's body by transmitting invisible near-infrared light and measuring its "time of flight" after it reflects off the objects. Although both the devices are available in the market today, this study chose to use Kinect version 2 for obvious reasons. A detailed comparison of both these devices can be found in (Wasenmüller and Stricker, 2016).

(a) Kinect V1

(b) Kinect V2

Figure 3.10: Kinect RGB-D sensors

The Kinect v1 measures the depth with the Pattern Projection principle. The depth is computed by projecting known infrared patterns into the scene. On the other hand, Kinect v2 contains a Time-of-Flight (ToF) camera and determines the depth by measuring the time emitted light takes from the camera to the object and back. Therefore, it constantly emits infrared light with modulated waves and detects the shifted phase of the returning light. Comparison of the specifications of the two devices are shown in figure 3.11.



	Kinect 1.0	**Kinect 2.0**
RGB camera (pixel)	1280 × 1024 or 640 × 480	1920 × 1080
Depth camera (pixel)	320 x 240	512 × 424
Max depth distance (m)	4.0	4.5
Min depth distance (m)	0.8	0.5
Horizontal FOV (degrees)	57	70
Vertical FOV (degrees)	43	60
Tilt motor	Yes	No
Skeleton joint define	20	26
Full skeleton tracking	2	6
USB	2.0	3.0
Price (€)	80	199

Figure 3.11: Specifications of Kinect sensors

The version 2 is more accurate than version 1 due to the hardware improvements. The version 2 device shows less deviations in measurements which are taken at a distance of 0.5 to 1.5 meters which is suitable for the intended application. However, while capturing, the accuracy of Kinect v1 decreases exponentially where Kinect v2 has a constant accuracy in form of an offset -18mm (Wasenmüller and Stricker, 2016). This is a very important fact for this study since a constant offset can be easily modeled. (Wasenmüller and Stricker, 2016) figured out a strong correlation of depth accuracy and temperature for the Kinect v2. Hence, the authors recommend to pre-heat the device for at least 25 min in order to achieve reliable results. This inference is also taken into account in this study. Kinect v1 incorporates the stripe pattern in the depth images, which is difficult to compensate. On the other hand, for Kinect v2 all central pixels show a similar accuracy. Only the image corners deviate. Hence, the placement of the sensor is in such a way that the projection area is in the center of the field of view of the depth sensor.

3.3.1.2 Point Cloud Library(PCL)

Point cloud library is a comprehensive free, BSD licensed(family of permissive free software licenses, imposing minimal restrictions on the use and redistribution of covered software) library for n dimensional point clouds and 3d geometry processing (Rusu and Cousins, 2011). It is a fully templated modern C++ library integrated with ROS (Robot Operating System). Point clouds can be acquired from hardware sensors such as stereo cameras, 3D scanners, or time-of-flight cameras, or generated from a computer program synthetically. PCL supports natively the OpenNI 3D interfaces, and can thus acquire and process data from devices such as the Intel PrimeSensor 3D cameras, the Microsoft Kinect, or the Asus XTionPRO etc. The PCL framework contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. To simplify development, PCL is split into a series of smaller code libraries, that can be compiled separately. The graph of the code libraries are shown in figure 3.12. PCL comes with its own visualization library, based on VTK providing a comprehensive visualization layer for n-D point cloud structures.

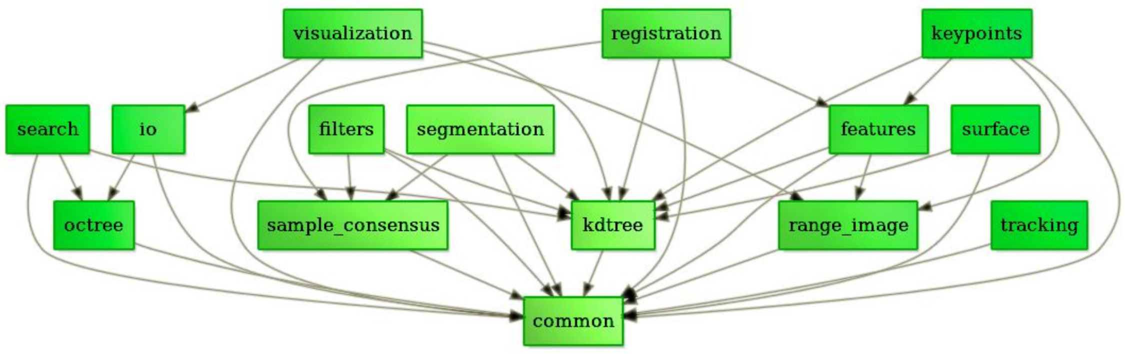


Figure 3.12: PCL code libraries

In this study, latest version of PCL (1.8.1) has been used and the following code libraries were utilized to implement the tracking system.

- libpcl filters: implements data filters such as downsampling, outlier removal, indices extraction, projections,etc.

- libpcl tracking: implements a tracker which is optimized to perform computations in real-time, by employing multi CPU cores optimization, adaptive particle filtering (KLD sampling) and other modern techniques.

- libpcl io: implements I/O operations such as writing to/reading from PCD (Point Cloud Data) files.

- libpcl segmentation: implements cluster extraction, model fitting via sample consensus methods for a variety of parametric models (planes, cylinders, spheres, lines, etc), polygonal prism extraction, etc.

3.3.2 Setup and Calibration

The setup consists of HITACHI CP-WU8600W projector with a resolution of 1920x1200 pixels and Kinect v2 RGB-D camera. The projector is fixed in a position where it can project the images with minimum interference by the users while manipulating the object and unreachable in order to avoid changes after calibration by accidental touching (Figure 3.13a). The zoom and focus of the projector is set to obtain clear and quality projections of images on the objects over the interaction area. The Kinect v2 RGB-D camera placed on a tripod at a position where it doesn't interfere with the projection and both the depth and color camera view most of the projection area. This is a necessary condition for successful calibration.

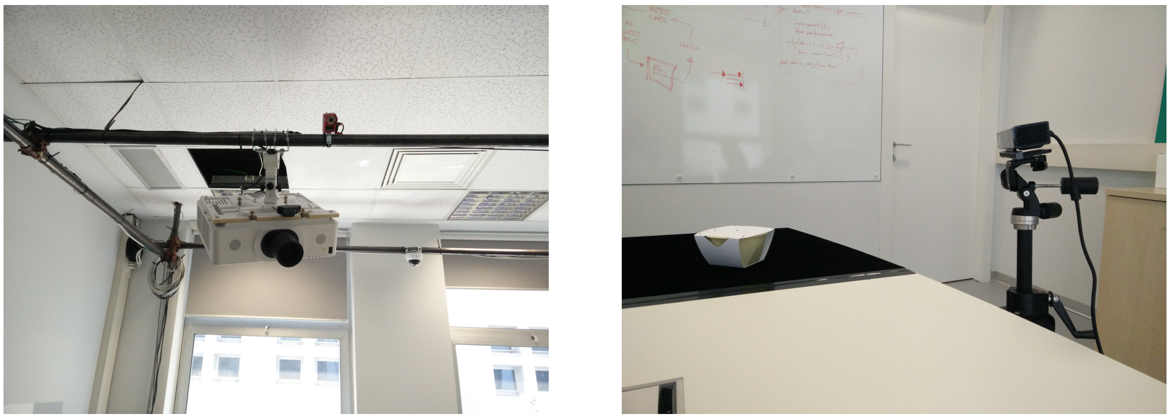
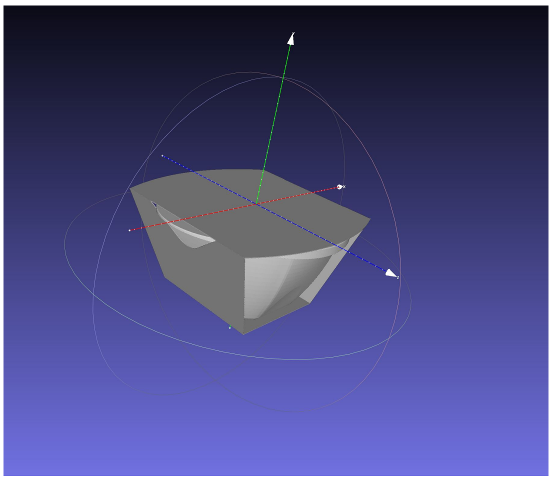

(a) Projector fixed on an upper beam **(b)** Kinect fixed on a tripod

Figure 3.13: SAR setup

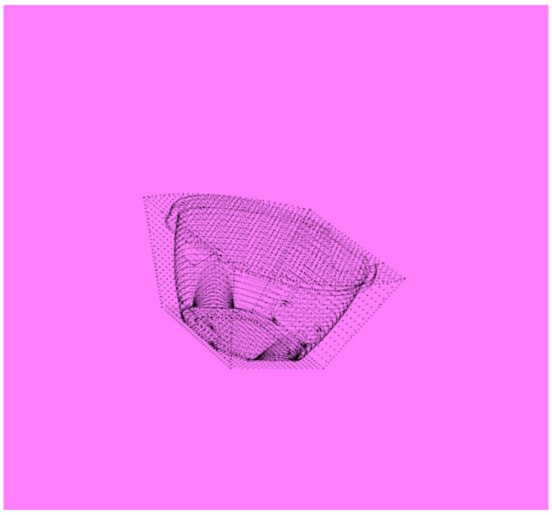
The Kinect v2 and the projector is calibrated using Pro-Cam calibration tool of the Microsoft Room-Alive Toolkit. First the projection area is made non-flat and the first calibration is performed in order to obtain the intrinsic parameters of the projector. These values were being used for the second calibration to get the position of projector with respect to the Kinect camera. RMS error of 0.6753 is obtained after calibration. Detailed explanation about this calibration methodology can be found in chapter 4 calibration.

The object to be tracked should be prepared before hand. In this study, the prototype of a soup packet which is being used in the SPARK platform is taken as the object. The object is white and texture-less as shown in figure 3.14c. The predefined model file of the object is already available. In order to use it as a target in the PCL tracking program, First the OBJ/PLY model file (Figure 3.14a) has to be converted to a point cloud using PCL's .OBJ to .PCD converter. The program uses vtkOBJReader (for .obj files) function or the vtkPLYReader (for .ply files) function in the `vtk_lib_io` libray to open the model file and coverts it into point cloud using vtkPolyDataToPointCloud function. The commonly used PCDWriter function in the `pcd_io` library is used to write the point cloud data into a .PCD file. The result is obtained as in figure 3.14b.

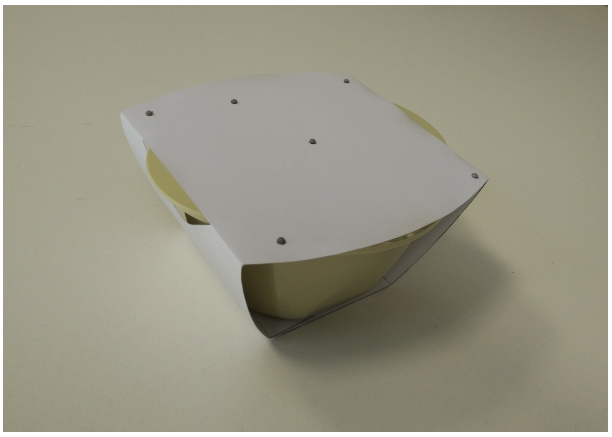
The scale and approximate initial position of the object should be assigned to the model using matrix transformation program. Since the scale of the .OBJ file is 1 i.e 1 meters, the scale



(a) Soup packet CAD model



(b) Converted point cloud of the model



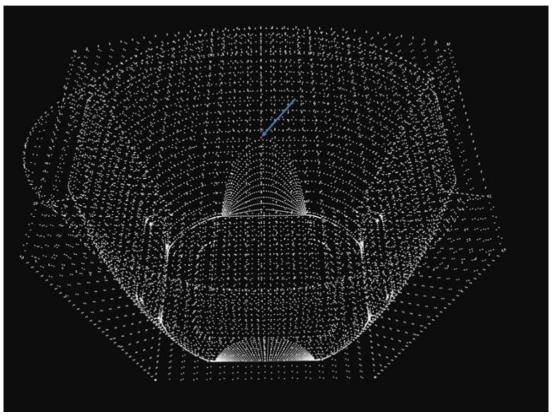
(c) Prototype of the soup packet

Figure 3.14

has to be reduced by a factor of 1/1000. The initial position of the object will be at (0,0,0) which is the origin of Kinect depth sensor. Hence, considering the object is placed on a table as show in figure 3.13b, the coordinates of the initial position of the target can be approximated to (0.0,-0.1,0.05). The translation and the scaling is done by setting the 4x4 transformation matrix as below given the sensor position as (0,0,0,0) and orientation as an identity matrix of order 3.

$$\begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & -0.01 \\ 0 & 0 & 0.001 & 0.05 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The diagonal values except for the last row represents the scale of the point cloud and first three rows of last column represents the X,Y and Z translations from the origin. The position doesn't need to be accurate. Since the tracking is done by predicting the position in the next frame using the particles, it is better to place the obj file close to the position of the real object in order to make the initial detection fast. The transformed and original models are shown in figure 3.15. The red colored points represents the original model and the white colored points represents the transformed point cloud.


(a) Scaling of the model

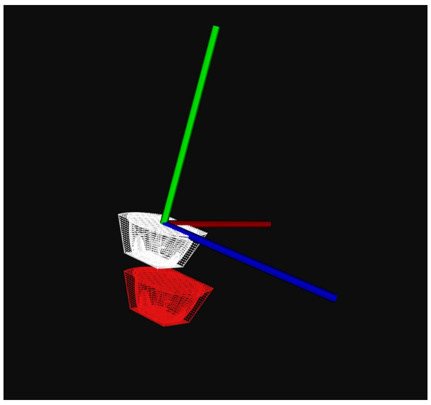

(b) Translation of the model

Figure 3.15: Transformation of the model

3.3.3 Principle and Algorithm

The `pcl_tracking` library contains data structures and mechanism for 3D tracking which uses Particle Filter Algorithm. This tracking enables the implementation of 6D-pose (position and rotation) tracking which is optimized to run in real time. Particle filter is a Bayesian filtering method based on sequential simulation from posterior probability distributions. It can handle nonlinear and non-Gaussian dynamics, so it has often been considered as a good alternative to the filtering methods designed upon Gaussian probability distributions for the state space. In the 6-DOF pose tracking problem, a particle filter is employed to sample the trajectory of an object of interest over time. In a particle filtering framework, the posterior probability density function of the object pose $p(X_t|Z_{(1:t)})$ at the current time t is represented as a set of weighted particles by the following equation:

$$S_t = (X_t^{(1)}, \pi_t^{(1)}), ..., (X_t^{(N)}, \pi_t^{(N)})$$

Where the particles $X_t^{(n)} \in$ SE(3) represent samples, the normalized importance weights $\pi_t^{(n)}$ are proportional to the likelihood function $p(Z_t|X_t^{(n)})$, and N is the total number of the particles. At each time t, the particle filter performs the sequential importance sampling with re-sampling. The current state χ_t could be estimated by the weighted particle mean as below:

$$\chi_t = \varepsilon[S_t] = \sum_{n=1}^{N} \pi_t^{(n)} X_t^{(n)}$$

For more detailed information regarding particle filter tracking approach, refer the works of (Choi and Christensen, 2013), (Bukey et al., 2017) and (Azad et al., 2011).The tracking process is graphically represented in figure 3.16. Initially, the algorithm generates a number of sample (predicted position) near the centroid of the target object. Prediction of each particle's position and rotation at the next frame is done using previous particle's information (At t = t - 1) about position and rotation. The likelihood function is then being checked for color and distance coherence and weights are assigned to the particles. Finally, the evaluate function compares real point cloud data from depth sensor with the predicted particles, and re-samples the particles.

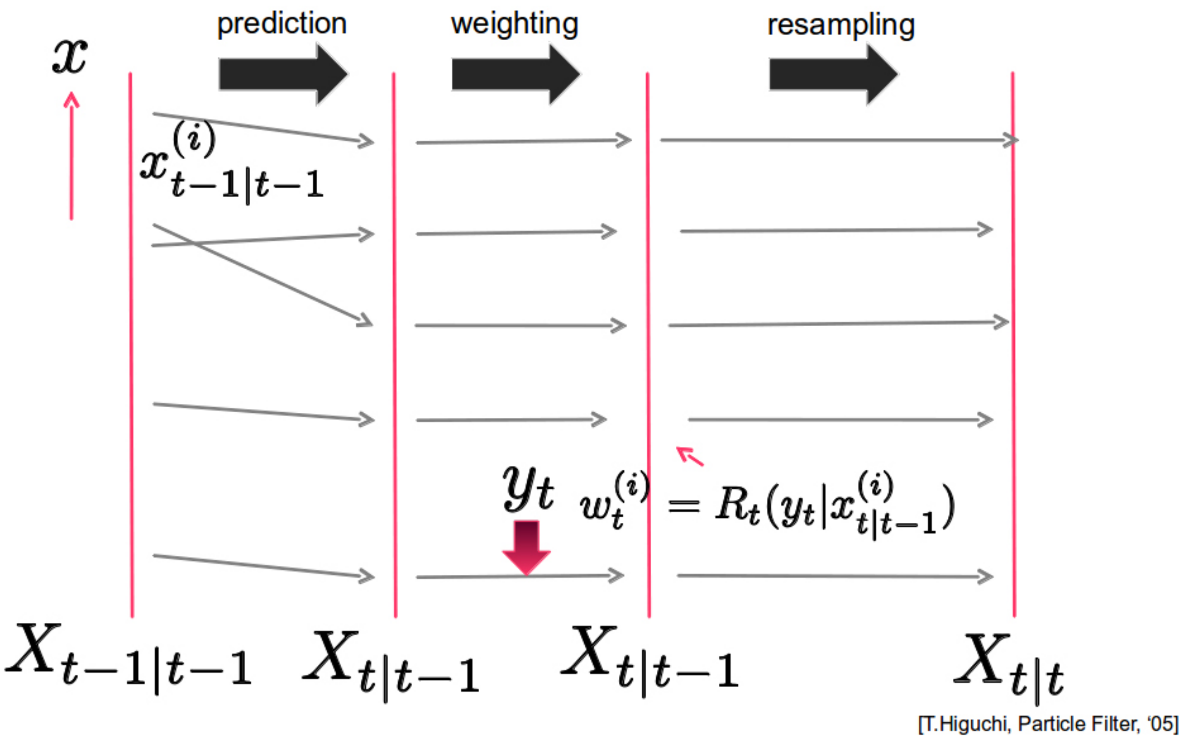


Figure 3.16: The process of tracking with Particle Filter algorithm

The tracking program is constructed from the default 6D object tracking example code of PCL source. In order to implement a tracking system with Kinect v2, the original code has to be modified and rebuild as it is scripted for the first generation Kinect sensors. The color and depth data can be extracted and imported real-time using grabber functions that comes with the official SDKs or 3rd party driver's. In this study, Openni2Grabber based on OpenNI2 is used for the purpose since the original code utilized previous version of it. Default values of the parameters for the tracking is set as below:

```
//Set all parameters for  KLDAdaptiveParticleFilterOMPTracker
 tracker->setMaximumParticleNum (1000);
 tracker->setDelta (0.99);
 tracker->setEpsilon (0.2);
 tracker->setBinSize (bin_size);

 //Set all parameters for  ParticleFilter
```

```
tracker_ = tracker;
tracker_->setTrans (Eigen::Affine3f::Identity ());
tracker_->setStepNoiseCovariance (default_step_covariance);
tracker_->setInitialNoiseCovariance (initial_noise_covariance);
tracker_->setInitialNoiseMean (default_initial_mean);
tracker_->setIterationNum (1);
tracker_->setParticleNum (600);
tracker_->setResampleLikelihoodThr(0.00);
tracker_->setUseNormal (false);
```

The maximum particle indicates the limiting number of particles (samples) employed to approximate the true posterior probability distribution and is set to 1000 considering computational cost. When setting the particle number, trade-off between computational cost and the variance of the resulting estimates must be considered. As the number of particles or sample size increases the variance decreases. Since in this study a color likelihood function is used, use of normals for the calculation is set to false. The likelihood functions are set by initializing a Coherence Class and adding the Coherence instance to coherence variable with addPointCoherence function. A color likelihood function can be set by defining the reference point type as XYZRGBA that can take positional and color data.

The tracker requires the target object to be loaded as a point cloud data file (.pcd file). The prepared point cloud of the object is imported using PCL's IO library. In order to improve real-time performance and easy coherence, the target point cloud and the streaming point cloud is down-sampled with a same down-sampling grid size (default 0.01). A voxel-grid filter is used for the purpose. The centroid of the target cloud is also computed in order to define the rotations. A transformation of the target cloud is performed in order to orient it with the streaming cloud. The down-sampled target reference cloud and the streaming cloud is set to to tracker to perform the evaluation.

```
 //prepare the model of tracker's target
 Eigen::Vector4f c;
 Eigen::Affine3f trans = Eigen::Affine3f::Identity ();
 CloudPtr transed_ref (new Cloud);
 CloudPtr transed_ref_downsampled (new Cloud);

//centroid calculation and transformations
 pcl::compute3DCentroid<RefPointType> (*target_cloud, c);
 trans.translation ().matrix () = Eigen::Vector3f (c[0], c[1], c[2]);
 pcl::transformPointCloud<RefPointType> (*target_cloud, *transed_ref,
 trans.inverse());

  //down-sampling
 gridSampleApprox (transed_ref, *transed_ref_downsampled,
 downsampling_grid_size_);
```

```
//set reference model and trans
tracker_->setReferenceCloud (transed_ref_downsampled);
tracker_->setTrans (trans);
```

The grabber function retrieves the full depth field of the Kinect sensor as point cloud. In order to omit the unwanted region of the cloud, a Passthrough filter is being used. Only the region of cloud that lies within (0, 1.2) in the x axis and (-5.0, 0.28) in the y axis is utilized for the calculations since this region represents the projection area of the SAR system. Due to initial noise, first few frames are ignored by setting a counter to 10. The position of the particle is obtained by calling getParticles() function. The position and orientation is then transferred to Unity 3D by setting up a TCP/IP server and client. The result of the tracking in a random frame is shown in figure 3.17.

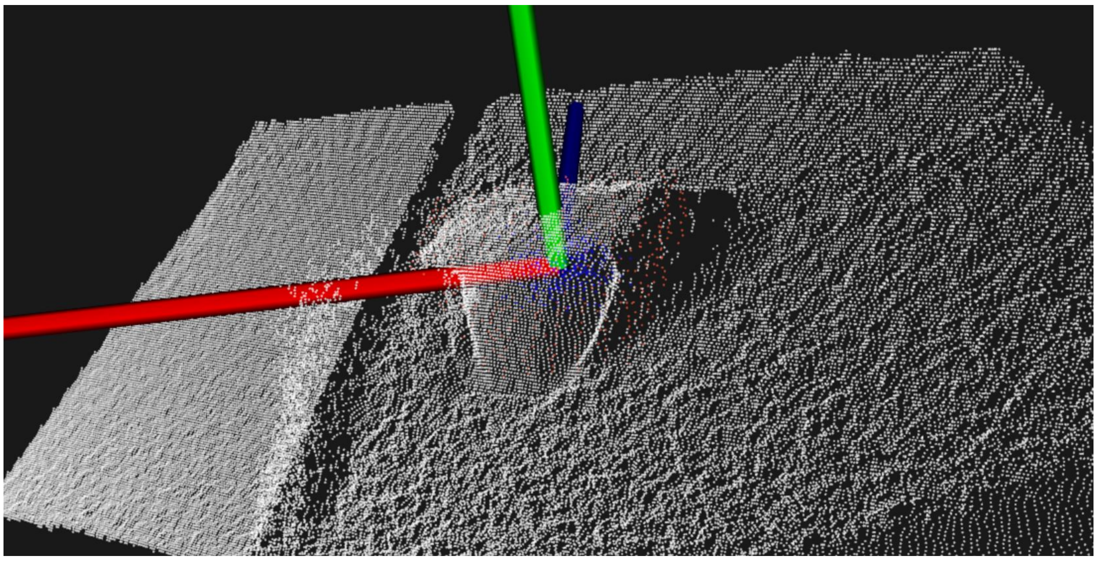


Figure 3.17: Output of the tracking program. The blue points represents the particles,red points represents target point cloud and the white points are streaming from Kinect

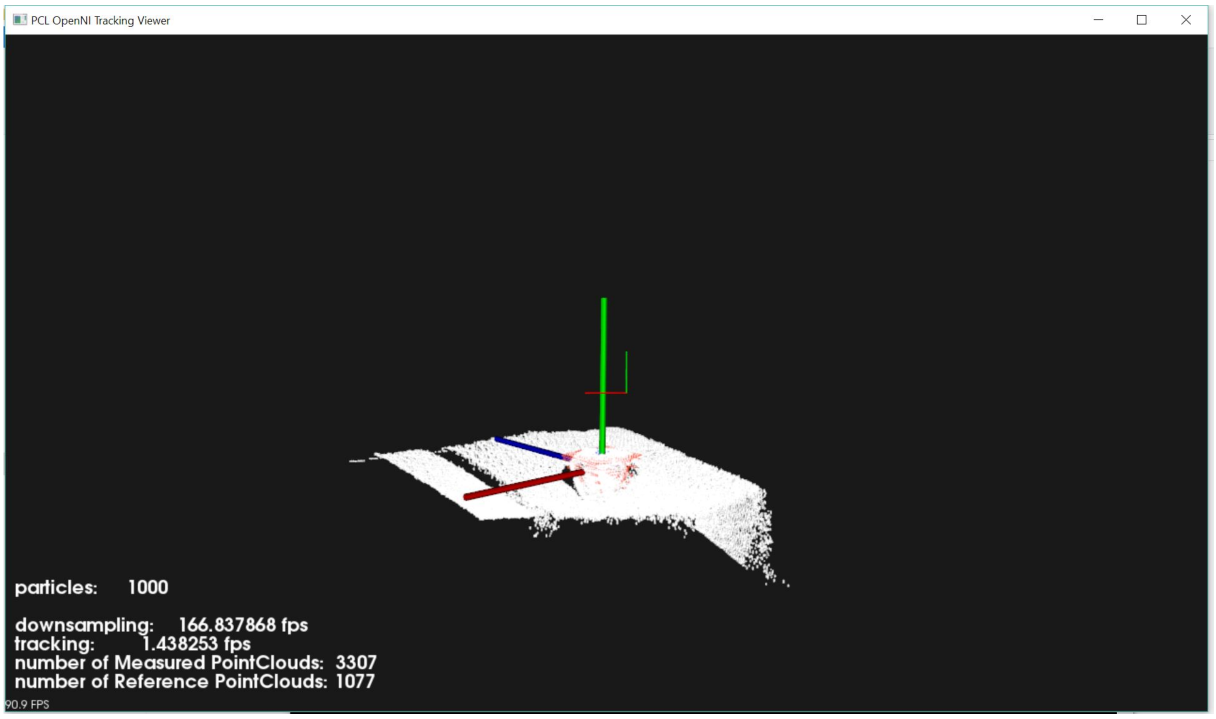
3.3.4 Registration and Rendering

Rendering of the SAR is implemented in Unity 3D. Unity 2017.4.1 free version has been used for the purpose. The scene consists of the calibrated projector-camera system. The projector is considered as a camera since the projector has to display whatever it views from its point of view. The intrinsic parameters of the projector camera matches with the intrinsic parameters of the projector. The setup is generated by Unity 3D plugin of Microsoft RoomAlive using the calibration file.

The tracking data is applied to the virtual model file of the object (soup packet). PCL uses right hand coordinate system,on the other hand unity uses left hand coordinate system. Hence, the x axis needs to be reversed while using the data tracking data for both rotation and translation.

3.3.5 Results and Inferences

Due to high computational requirements, the tracking system is implemented in a PC with Intel Core i7-6800k CPU (12 cores) and 65 GB RAM. The system is able to detect and track the texture-less white object as shown in figure 3.18. However, the system failed to provide desired accuracy, stability and real-time performance of tracking. Deviations in the position and orientation (significant deviations in the orientation) were observed. The virtual object keeps deviating from the real one on every update of the frame even when the real object is stationary. Average variations in position are found to be, 12mm in the x direction, 7 mm in the y direction and 9 in the z direction. Average variations in orientation are appeared to be, 8 degrees around x axis, 4 degrees around y axis and 3 degrees around z axis. A low average tracking speed of 1 to 2 FPS(Frames Per Second) is obtained (Figure 3.18) even with the high PC configuration.



```
particles:     1000

downsampling:    166.837868 fps
tracking:        1.438253 fps
number of Measured PointClouds:  3307
number of Reference PointClouds: 1077
90.9 FPS
```

Figure 3.18: SAR Unity 3D scene components

An attempt is made to improve the speed, accuracy and stability of tracking by changing the tracking parameters. The parameters that can be changed and their default values are:

- Down-sampling grid size of the streaming point cloud ($Ds_{Grabber}$) - 0.01
- Maximum number of particles (N_{max}) - 1000
- Down-sampling grid size of the target object point cloud (Ds_{OBJ}) - 0.01

By properly adjusting the parameters, an optimum performance and accuracy can be obtained. Each parameter is changed keeping the others constant. First the down-sampling grid size of point cloud obtained from the Kinect is changes while keeping other parameters constant.

$Ds_{Grabber}$	N_{max}	Ds_{OBJ}	Average tracking speed (fps)
0.001	1000	0.01	2
0.01	1000	0.01	2.18
0.02	1000	0.01	2.4
0.1	1000	0.01	3.9

It can be observed that, the change in down-sampling grid size of point cloud obtained from the Kinect doesn't significantly affect the tracking speed. Grid size equal to 0.02 is taken for further analysis considering the accuracy of results. The down-sampling grid size of the target object point cloud is then changed keeping the other parameters constant.

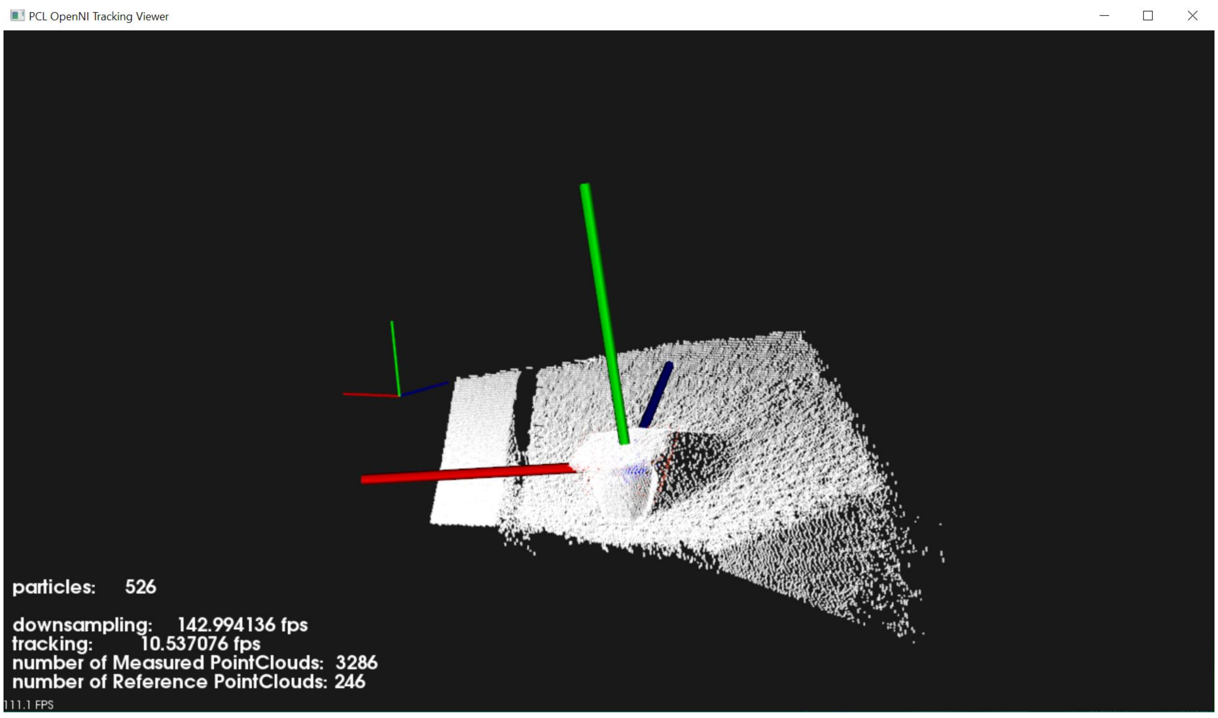
$Ds_{Grabber}$	N_{max}	Ds_{OBJ}	Average tracking speed (fps)
0.02	1000	0.001	0.013
0.02	1000	0.01	2.4
0.02	1000	0.015	6.2
0.02	1000	0.02	13.3
0.02	1000	0.05	15.8
0.02	1000	0.1	not detected

Small increase in down-sampling grid size of the target object point cloud increases the tracking speed significantly. However the accuracy of tracking is decreased due to the decrease in the amount of information. A value close to 0.1 results in poor object detection. Therefore, considering the fore-mentioned issue a grid size of 0.02 is taken for further analysis. Finally, maximum number of particles changed keeping the other parameters constant.

$Ds_{Grabber}$	N_{max}	Ds_{OBJ}	Average tracking speed (fps)
0.02	100	0.02	43.52
0.02	500	0.02	15.16
0.02	1000	0.02	13.3
0.02	2000	0.02	10.54
0.02	5000	0.02	9.2

It is clear that the number of particles significantly changes the tracking speed. High tracking speed can be obtained with a low maximum particle number. However, poor accuracy is obtained with low values. Large number of particles indicates large number of predictions considered. Hence, improved accuracy can be obtained with an expense of high computational power.

The values of parameters obtained are specifically related to the configuration of the PC used. Different results will be obtained with different configurations. The values slightly change even when the program run repeatedly in the same PC. This could be due to the effect of other programs running in the background and the speed of transmission of data from Kinect sensor to PC. Hence, an average value obtained from different trails are taken into consideration for the analysis. The result obtained applying the optimized parameters values are shown in figure 3.19.



```
particles:     526

downsampling:    142.994136 fps
tracking:        10.537076 fps
number of Measured PointClouds:  3286
number of Reference PointClouds: 246
111.1 FPS
```

Figure 3.19: Output of tracking after optimization

The parameter optimization improved the real-time tracking performance and accuracy. However, since the input streaming point cloud from the Kinect sensor changes in every frame, stability is still at a level of unacceptability. Therefore, a moving average method is applied on the tracking data. The idea is to average the position and orientation values obtained from tracking in order to reduce its variations. Taking a 20 values of position and orientation and averaging it over time makes the model steady. However, this causes large errors in the output especially to the orientation. The results obtained are shown in figure 3.20.

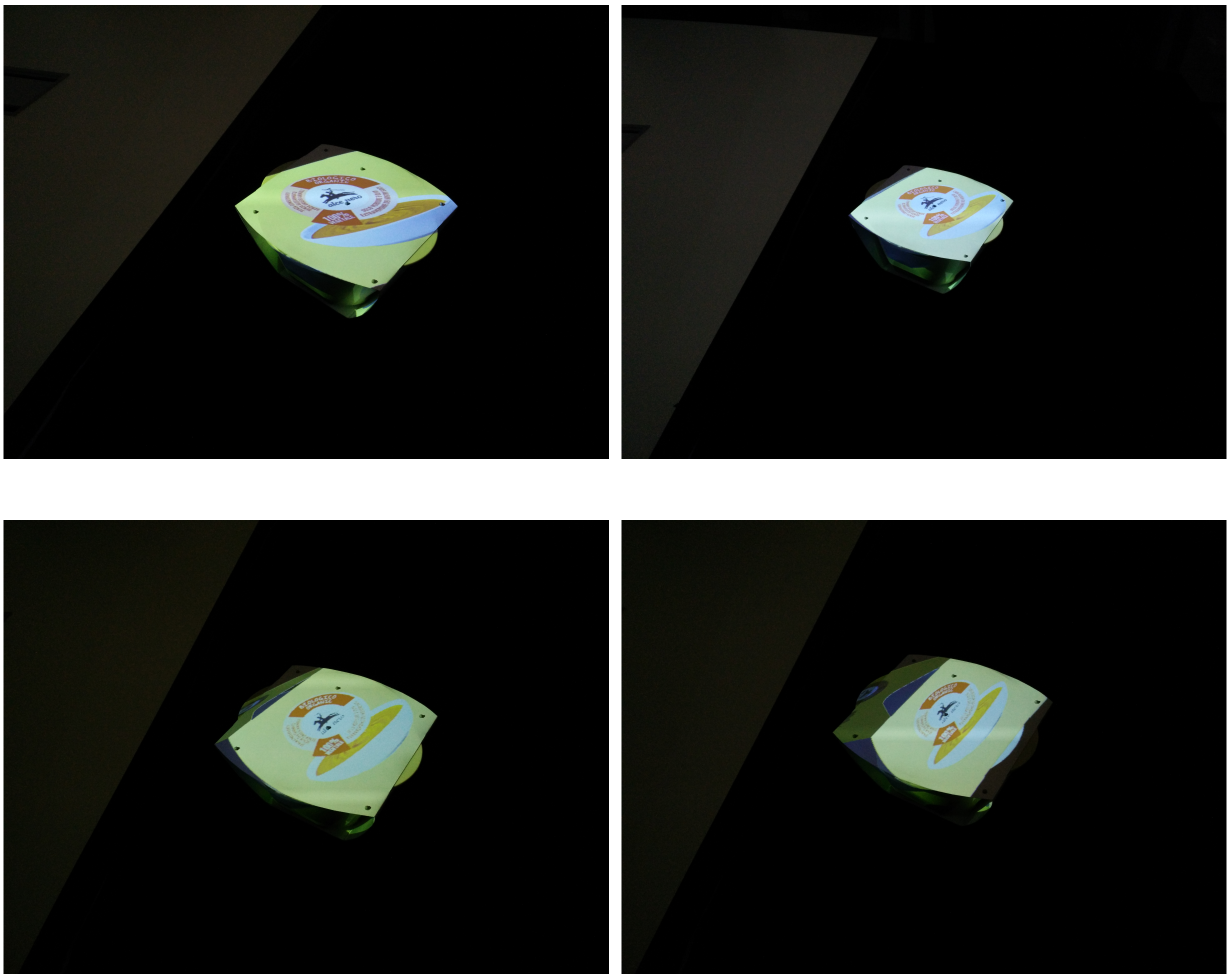


Figure 3.20: Results after averaging

3.4 Discussion and Comparison

From the study performed, it can be understood that both the tracking approaches proposed have several advantages over one another. The approach based on the marker tray provides stable tracking results whereas, the particle filter tracking fails to provide a stable output. The accuracy of the position is also better in the marker-based tracking. It is identified as the cheapest tracking solution due to the availability of low cost web-cameras. Web-cameras are available from 15 € to 90 € whereas, Kinect v2 costs more than 100 €. Since the manipulation of the object is being done by manipulating the tray, it causes no interference with the projection by hands. However, the bottom portion of the object is completely masked due to the tray. When the tray is horizontal with respect to the camera, the markers are poorly detected and causes loss of tracking. Hence manipulation of the tray to certain angles are restricted in this method. The calibration procedure for this method is time consuming than the Kinect sensor based approach. The particle filter tracking with Kinect v2 has an easy calibration procedure due to the availability of an automatic calibration program implemented by Microsoft. This calibration method allows the device to be placed far from the projector that gives more degrees of freedom. Since the approximate initial position of the object is given to the tracker, easy and fast initial detection is achieved unlike the marker-based approach. Poor initial detection due to camera position is found to be the main disadvantage of the marker tray approach. The use depth data allows the tracking system to function in any light conditions. Whereas the marker-based approach requires good lighting conditions to properly detect the markers. Fast motion of the object causes loss of tracking for both the proposed systems. However, the point cloud based approach can detect fast motions with high processing power. Absence of tray makes it suitable for SAR systems with objects that are manipulated by hands. Nevertheless, the hands interferes with the projection and causes poor tracking output due to change in the point cloud of the object of interest.

The main advantage of the proposed systems over the Optical tracking approach is the cost. The OptiTrack cameras alone costs over 3000 € whereas, the proposed systems have a total cost less than 500 €. The proposed systems also avoid the use of markers on the object. The markers attached to track interferes with the projection even though, they are small in size. Since the calibration of the prototype is not required and the simple calibration methods are available, the proposed method saves significant amount of time and human effort. Despite of the advantages, both the proposed system failed to provide robust and accurate tracking. A detailed study is required to solve the issues with the proposed tracking system and future hardware improvements could avoid problems due to technological limitations.

Chapter 4

Calibration

Similar to tracking, calibration is also challenging in SAR. Accurate calibration of the devices is necessary for SAR systems. A small imperfection could cause significant misalignment in the visual output while the object is being tracked. Hence, efforts must be put in order to develop a robust, accurate and simple calibration method for SAR systems. The difficulty and methodologies vary with the tracking technologies used. However, projectors calibrated with a camera using structured light is a widely used technique. For SAR systems with camera based tracking, this procedure is sufficient since it provides the relative position of projector coordinate system and the tracking coordinate system. From the literature review and from the attempts made in this study, it has been inferred that the optical tracking with IR markers is the best tracking method available for SAR systems with small non-static objects. In order to make the system work, an additional procedure must be carried out to obtain relationship between the coordinates of tracking and projections units. This process is time consuming and cumbersome. This is one of the major drawbacks of the optical tracking system. This part of the study deals with the evaluation of the currently used method (e.g calibration method employed for SPARK) and the implementation and evaluation of a proposed method for the same. The main goal is to solve this major drawback of optical tracking system by simplifying the calibration procedure.

4.1 Evaluation of Calibration in SPARK platform

The calibration procedure employed for the SPARK platform is cumbersome and difficult to understand. Coordinates of the tracking system and projector is connected using a camera. First calibration of the projector-camera system is performed and obtained results are used to compute the relation with the tracking coordinates system. The open-source software scan3d-capture (pro,) is used to calibrate the projector-camera setup. A printed chess board is required for the process. A 7x10 checker board is used for the purpose. Number of corners (6x9) and the dimension (in mm) along the x and y axis are required to be inserted. A Projector with a resolution of 1920x1200 and a web camera with a resolution of 2592x1944 are calibrated as a stereo pair. The camera is attached to the projector as shown in figure 4.1 in order to get better calibration results. By placing the checkerboard on the projection area, calibration process can be initiated. As the calibration begins, gray code patters are being displayed and captured by the camera. Once the capture ends the checker board needs to be positioned in a different position and orientation to capture the second set of images. At least 3 different sets of capturing process should be done in order to compute the calibration parameters. This process can be exhausting when more number of sets are taken to increase the accuracy of the results. Number of patterns and exposure time parameters can be controlled accordingly with the resolution of the projector and the speed response of the camera respectively.

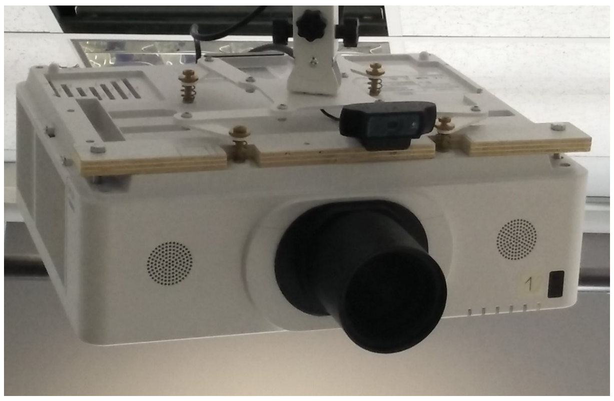


Figure 4.1: Camera-Projector arrangement

Both the projector and the camera are described using pinhole model extended with radial and tangential distortion in order to compensate for distortions in the projected patterns. From the images of the checkerboard of known dimensions at several orientations, the algorithm calculates the camera calibration parameters using the relation between the checkerboard corners in a camera coordinate system and a world coordinate system attached to the checkerboard plane. It uses OpenCV's findChessboardCorners() function to automatically find checkerboard corner locations and, then, it refines them to reach sub-pixel precision. Finally, a call to the function calibrateCamera() returns the calibrated camera parameters. The projector is also described with the same mathematical model of the camera with an idea that if the projector were a camera, it would be possible to capture images from its viewpoint, to search checkerboard corners in them. In order to compute checkerboard corner coordinates in the projector coordinate system,first the structured-light sequence is decoded and every camera pixel is associated with a projector row and column, or set to "uncertain". Then a local homography is estimated for each checkerboard corner in the camera image. Each of the corners is converted (Figure 4.2) from camera coordinates to projector coordinates applying the local homography just found. OpenCV's stereoCalibrate() function with the previously found checkerboard corner coordinates and their projections are used to find the relative rotation and translation between projector and camera. The output is a rotation matrix R and a translation vector T relating the projector-camera pair. Detailed information about this calibration method and the algorithm behind can be found in (Moreno and Taubin, 2012).

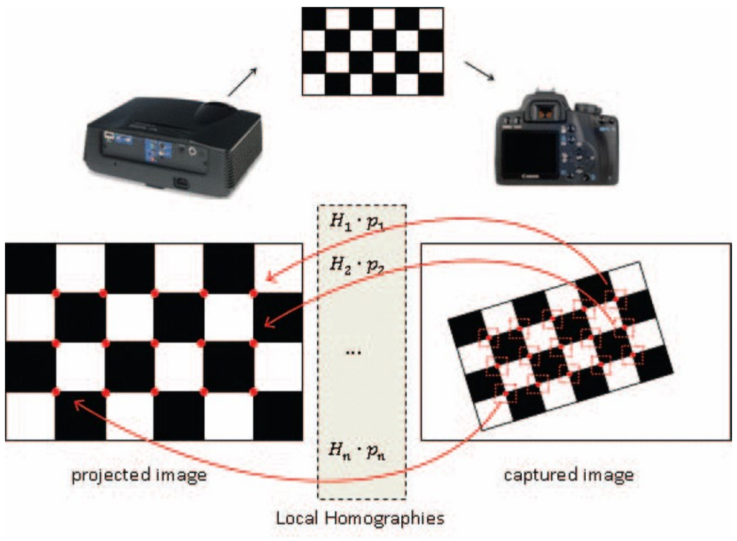


Figure 4.2: Projector corner locations are estimated with sub-pixel precision using local homographies to each corner in the camera image

Verification of the previously performed calibration and calculation of the relation between projector-camera system and OptiTrack system is performed using a Python script. Although this process is time consuming, it enables the correction and adjustments to the calibration results so that an accurate calibration can be obtained without repeating the previous steps. Since two projectors are used for the SPARK system, each projector is to be calibrated separately. Hence the previous steps needs to be done twice which makes the whole process long and exhausting.

A physical chess board is required to be prepared for the extrinsic calibration of the projector. Four retro reflective markers are placed on the board at random positions such that it doesn't cover corners or edges to avoid poor detection of the chessboard by the camera. A fifth marker is places at a known position (known offset from the corner). This preparation process is challenging and should be done carefully to avoid calibration errors. Using the previous calibration data, the program calculates the extrinsic parameters of the projector with the OptiTrack motive running in background. The previously generated coordinate system of the projector-camera (at the corner of the chessboard)is transformed to the OptiTrack coordinate system (at the marker placed at known offset) as in figure 4.3. Finally the calibration files are imported to Unity 3D and manual creation and transformation of the cameras with the obtained coordinates are performed.

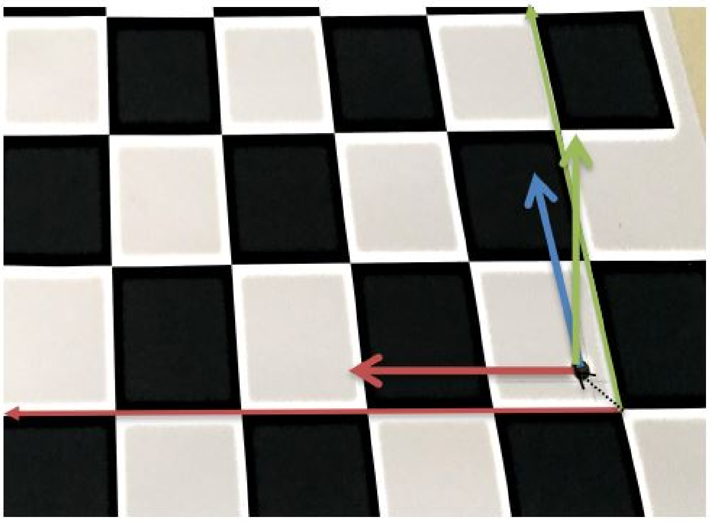


Figure 4.3: Transformation of the coordinate system of projector to tracking system

The object used is also needs to be calibrated. The initial(neutral) orientation and the position of pivot of the physical prototype with respect to the virtual object is set manually during this phase. Final adjustments are done in Unity 3D to correct the errors.

4.2 RGB-D data based calibration

Structured light calibration of projector-camera system and the calculation of extrinsic parameters relating the projector-tracking system is found to be complex and time consuming. Repetition of the process is required to calibrate multiple projector systems which makes the process even longer to complete. Hence an approach based on RGB-D data that could simplify and speed-up the process is considered.

4.2.1 Concept

The proposed calibration method consists of three main procedures. First the calibration of the projector is carried out using Kinect v2. This results in finding the position of the projector with respect to the Kinect sensor. RoomAlive pro-cam calibration tool based on the method introduced by (Jones et al., 2014), is used for the purpose. This tool enables the easy and automatic calibration of multiple projectors. Secondly, the real world coordinates of a point in space is calculated from the color and depth data from Kinect sensor with the help of PCL library functions. A color based segmentation using PCL is used for extracting the coordinates of the point. Finally, the origin of the OptiTrack system is set at this point with known coordinates. The resulting calibration data is used in Unity 3D environment to setup a SAR scene with world origin at the Kinect sensor and the origin of the tracking system at a known point from the world origin.

4.2.2 Methodology

The methodology of the proposed calibration is divided into four sections. The first Pro-Cam calibration section deals with the calibration of projectors with Kinect sensor. The procedure and the principle of operation is explained in detail. In the second section, the color based region segmentation procedure is explained. Results from the previous procedure is being used for the third step that creates a SAR scenario in Unity 3D to align the real and virtual objects and provides the possibility to set the origin of tracking system. Finally, the origin is set at the already known position.

4.2.2.1 Pro-Cam Calibration

The RoomAlive Toolkit from Microsoft is a tool that enables the creation of dynamic projection mapping experiences. This toolkit has been in internal use at Microsoft Research for several years and has been used in a variety of interactive projection mapping projects such as RoomAlive, IllumiRoom, ManoAMano, Beamatron and Room2Room. The toolkit consists of two separate projects. ProCamCalibration and RoomAlive Toolkit for Unity. ProCamCalibration is a C# project that can be used to calibrate multiple projectors and Kinect cameras in a room to enable immersive, dynamic projection mapping experiences.The calibration process is completely automatic, requiring no user intervention. In the proposed method,Position and orientation of the projectors from the Kinect is calculated using ProCamCalibration project of the RoomAlive toolkit. The processes performed during this calibration phase is summarised in the figure 4.4.

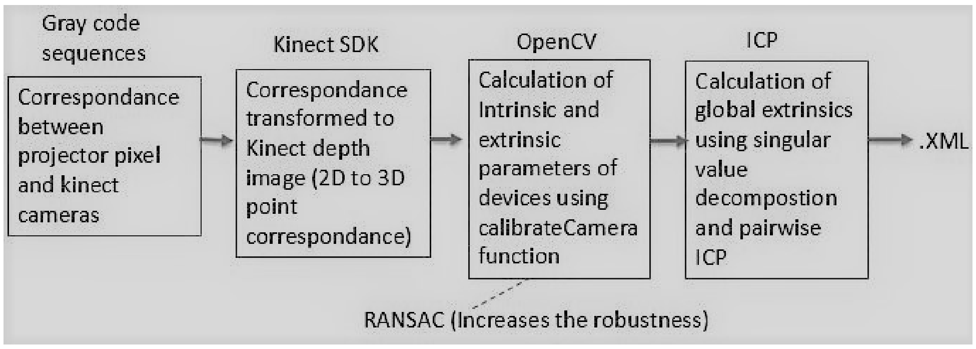

Gray code sequences
Correspondance between projector pixel and kinect cameras
Kinect SDK
Correspondance transformed to Kinect depth image (2D to 3D point correspondance)
OpenCV
Calculation of Intrinsic and extrinsic parameters of devices using calibrateCamera function
ICP
Calculation of global extrinsics using singular value decompostion and pairwise ICP
.XML
RANSAC (Increases the robustness)

Figure 4.4: Processes in Pro-Cam calibration

The project uses SharpDX and Math.NET Numerics packages. The calibration application is built using Visual Studio 2017 Community Edition and the Kinect for Windows v2 SDK is required to run the built application. The Calibration of multiple projectors can be performed using the application. In this work calibration of one projector and two projectors has performed.

The Placement of Kinect is an important aspect of the procedure. The Kinect v2 sensor should be placed so that it views most of the projected image, and so that the projected image occupies most of the Kinect's viewable area. Precise alignment is not critical, however both the Kinect color camera and Kinect depth camera must observe a good portion of the projected image in order for calibration to succeed. The alignment can be done by running the Kinect SDK's Color Basics sample. For a system of two projectors, a single kinect camera can be used for calibrating both projectors, if the above mentioned requirement is satisfied. Hence the placement and initial adjustments of the projectors are critical for this method of calibration. It is necessary that the Kinect is placed parallel to the ground (or table) as shown in figure 4.5 in order to ease the second section of the calibration methodology. The reason for this will be explained in the section 4.2.2.2. The parallel alignment can be done using a spirit level.

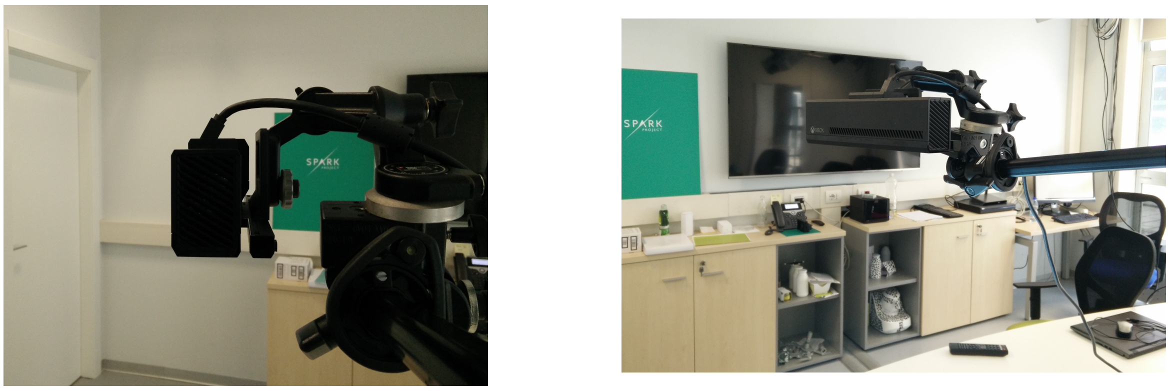


Figure 4.5: Kinect Positioning

The calibration procedure will attempt to calculate projector focal length and principal point. This will succeed only if the projection surface is not flat. Hence, the first calibration should be done with a non-flat surface (figure 4.6) to recover projector focal length and principal point. To create a non-flat projection surface, sizable objects like light (white/yellow)colored boxes can be placed in the projection area. It is advised to avoid black colored objects since it absorbs the infrared light which can cause large errors in the calibration. In subsequent calibrations, if the projection surface is determined to be flat, the previously calculated focal length and principal point will be used.

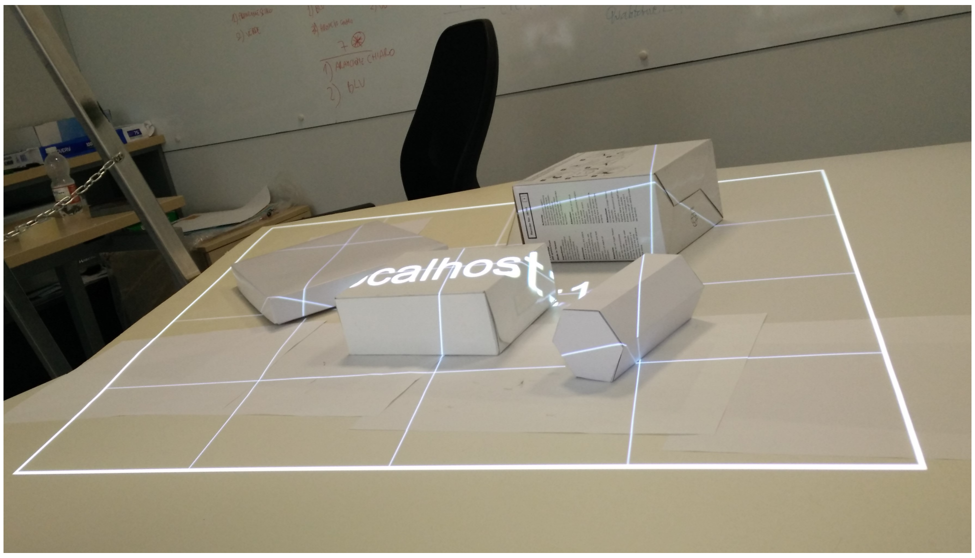


Figure 4.6: Non-flat projection surface

Kinect server and projector server programs needs to be run with administrative privileges on the background. CalibrateEnsemble.exe is the user interface that allows the acquisition of calibration images, performs the actual calibration calculation and inspects the results. The number of cameras and projectors are inserted at the beginning while creating a fresh calibration file. This allows the interface to make provisions to assign parameters of each projector and camera unit for multiprojector calibration. The display indexes of the projectors are to be inserted manually as the default values are set while creating the calibration file. 'Show Projector Server Connected Displays' option allows to verify the display indexes of the projectors. An important aspect to be noted is that the ProjectorServer and the calibration process is not High-DPI aware. If the display showing the display index does not fill the entire projected display area, turn off DPI scaling (set to 100%), and try again.

After reloading the modified calibration file the capturing process can be initiated by the 'Acquire' option. This should trigger a serie of Gray code patterns in turn (figure 4.7). These are captured and saved by the Kinect color camera. Gray code patterns are used to map from a given pixel coordinate in the Kinect color image to a pixel coordinate in the projector. For multiple projector camera units, all cameras observe the Gray code patterns in order to establish which cameras belong to a given 'projector group'. Additionally, the depth image from each Kinect depth camera is saved. CalibrateEnsemble recovers Kinect camera calibration information. This is used to compute the precise 3D coordinate of a given point in the depth image, and to map this 3D point to color camera coordinates.

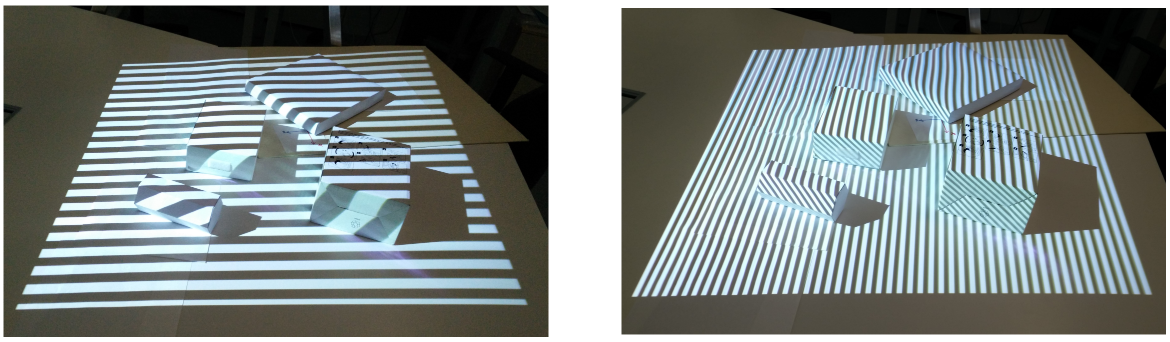
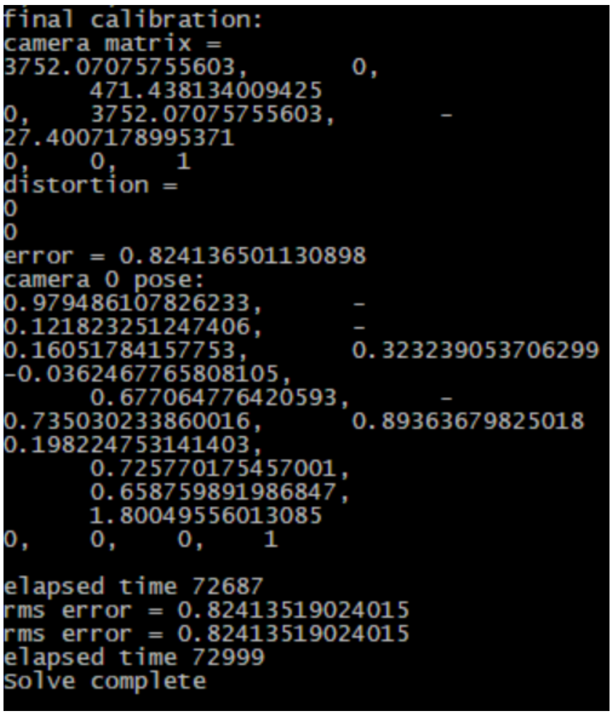


Figure 4.7: Gray code patterns

Once the acquisition is completed, 'Solve' option is used to initiate the calibration process. During the solve phase of CalibrateEnsemble, for each camera in the projector group, points in the saved depth camera image are transformed to 3D points. The 2D color camera coordinate is computed via the saved Kinect calibration information. These color camera coordinates are then associated with projector coordinates by Gray code mapping. The end result is a set of 3D points for each depth camera in the group, and their associated 2D projector coordinates.

Projector intrinsics (focal length, principle point) and camera extrinsics (depth camera pose, in the projector coordinate frame) are computed by minimizing the error in the projection of 3D points to projector points. Because this projection is nonlinear, a standard Levenberg-Marquardt optimization procedure is used. This is performed for each projector in turn. The 'ensemble'(set of projector groups that share cameras) necessarily includes cameras that belong to multiple projector groups. These cameras can be used to put all camera and projector poses in the coordinate frame of the depth camera of the first Kinect listed in the .xml file. This is done via successive matrix compositions. The 'ensemble' can be thought of as a graph of projectors, where each projector is a node, and cameras that belong in more than one group establish an edge between projectors. The previous step of unifying the coordinate systems is done in a greedy fashion, with the consequence that there may be multiple possible estimates of a camera or projector's pose if this graph includes a cycle (consider a ring of projectors, or a 2x2 grid of projectors). To address this possible source of error, a final optimization is performed over all projectors, solving for a single pose for each projector and camera. RANSAC(Random Sample Consensus) algorithm is used for this purpose which increases the robustness. The final RMS error reported must be some value less than 1.0 for a successful calibration. The results of a successful calibration run is shown in figure 4.8. In order to improve the accuracy of calibration, the RMS error should be reduced which can be done by repeating the process adjusting the position of the camera close to the projector keeping the conditions of placing the devices satisfied. However, an error value less than to 0.85 is fine for general applications.



```
final calibration:
camera matrix =
3752.07075755603,       0,
       471.438134009425
0,     3752.07075755603,        -
27.4007178995371
0,     0,     1
distortion =
0
0
error = 0.824136501130898
camera 0 pose:
0.979486107826233,      -
0.121823251247406,      -
0.16051784157753,       0.323239053706299
-0.0362467765808105,
       0.677064776420593,       -
0.735030233860016,      0.89363679825018
0.198224753141403,
       0.725770175457001,
       0.658759891986847,
       1.80049556013085
0,     0,     0,     1

elapsed time 72687
rms error = 0.82413519024015
rms error = 0.82413519024015
elapsed time 72999
Solve complete
```

Figure 4.8: Pro-Cam Calibration results example

The first calibration step with non-flat surface was just to extract the intrinsics of the projector. If the projection surface is required to be flat, further calibration steps can be done by keeping the projection surface as required. The projector section in the .xml file features a "lockIntrinsics" field which when true, ensures that calibration leaves focal length and principal point unchanged during further calibrations.

Inspection of the calibration results can be done in the interface. The verification of the calibration results is performed by changing the projection matrix of the graphics camera. It can be done with the View menu. It defaults to an ordinary perspective projection. Selecting the second option in the View menu ('Projector name', where name is the name found in the .xml file) sets the projection and view matrices to match that of the projector, as calculated in the calibration procedure. If the calibration was successful, the white projected frame should fall just inside the border of the rendered frame(figure 4.9). A projector lens is being considered to have a focal length and principal point (or optical center) just like a camera, and setting the graphics projection and view matrix to that of the projector essentially renders the scene from the projector's precise point of view, as if it were a camera. It should therefore 'see' precisely what it projected, as it was projected.

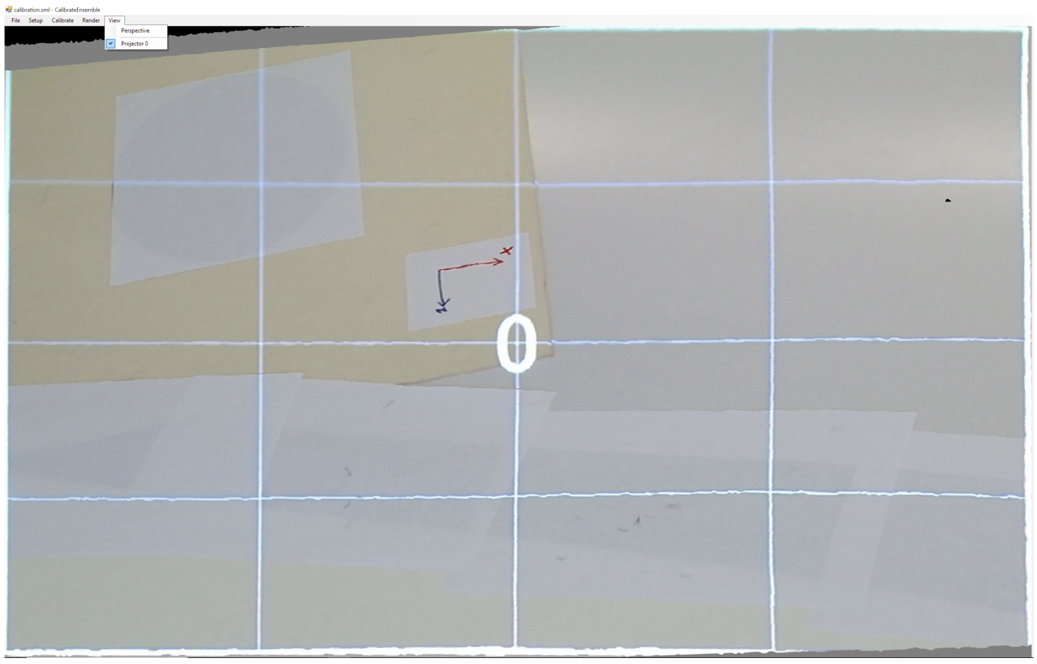


Figure 4.9: Projector Camera View

The RoomAlive Toolkit projector/camera calibration tool has been designed from the ground up to handle multiple simultaneous projector and camera combinations. In order to calibrate multiple projectors, the frustums of the projectors and cameras must overlap enough so that the calibration procedure can infer the overall arrangement of all cameras and projectors. Since two projectors are used for the SPARK platform, a single Kinect camera should be sufficient enough to perform the calibration as it can view the projected images from the two projectors from the fixed position.

4.2.2.2 PCL

In order to correlate the coordinate system of the optical tracking system and the coordinate system of calibrated Kinect-Projector unit, a physical point is being set. Since the rendering of the SAR scene deals with recreation of the actual physical space and its elements in Unity

3D, the physical origin of the optical tracking system can also be included. As explained in the previous section, the position and orientation of the object tracked by the Opti-track is with respect to an origin at a manually positioned point in the physical space. Depth and RGB data from the Kinect can be used to identify this point with the help of powerful algorithms of PCL. A color based region growing segmentation and cluster extraction is performed on the point cloud obtained from the Kinect sensor.

The color based segmentation is based on colorimetrical similarity and spatial proximity of the points in the point cloud. The segmentation method has mainly three steps: region growing, region merging and refinement(Zhan et al., 2009). The region growing is a process in which a particular region is segmented starting from the first point of the input point cloud and grows till an unlabeled point is reached. When the process meets an unlabeled point, a new region is initiated, the current unlabeled point is added to the region and pushed into a points-to-grow stack Points. Then each time Points pops one point and finds its k nearest neighbors within a given distance TD and grows the current region on the basis of colorimetrical similarity. The colorimetrically similar points are added to the current region and added to Points and the current growing process terminates when Points is empty. The procedure iterates reading in an unlabeled point, grows a new region and terminates until all the points are labeled. The output of this process is roughly segmented regions to be used in region merging and refinement. Euclidean distance metric is used to measure colorimetrical difference in RGB space since it gives the positional relationship of two objects in 3D space. The colorimetrical difference function is defined as the equation below:

$$CD(C_1, C_2) = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2}$$

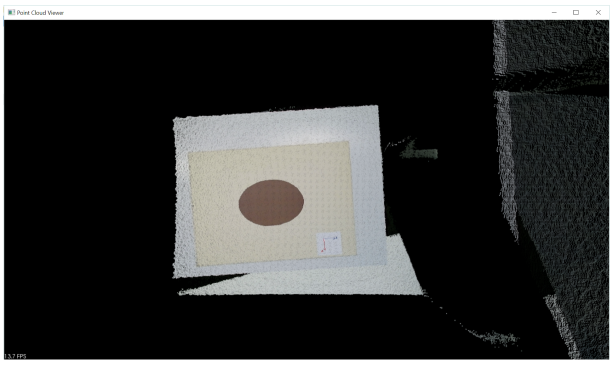
Where C1, C2 represent color vectors of two objects which can be points and regions. RGB values of points can be read directly from raw data.

The Region merging and refinement process merges and refines the roughly segmented regions output from the region growing process. A region-region threshold TRR is used for colorimetrical similarity if two neighboring regions are colorimetrically similar, then the two regions are considered as homogeneous and they are recorded in a homogeneous region lists set H which records homogeneous regions in the same list. The region merging process is analogous to the region growing process. For each region R_i, the process firstly searches it in H and ensures that the region has been in some list in H. Then search the neighbors of R_i. The searching can be realized by using KNN (k nearest neighbour) on the points in R_i. To ensure the correctness of neighbor searching, a distance constraint TD2 and number constraint TNN2 are used. The searched neighboring regions are compared with R_i on the basis of colorimetrical similarity. The similar neighbors are added to the list which contains R_i. After the merging process, all the regions in R have been classified into different lists in H, then merge the regions in the same lists in H and get merged regions R'. Finally, refinement of R' is done using a threshold Min for the minimal size of an acceptable region. The regions which has less than Min points are merged to their neighboring regions. For each point P_i in the region, its nearest neighboring point is found which belongs to another region $R_j$and add P_i to R_j.

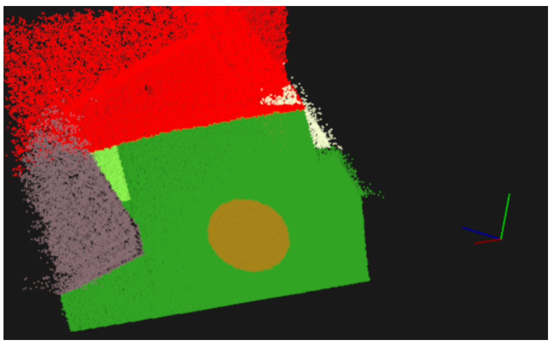
The default parameters set for the segmentation process are given below. This parameters can be changed however the default values gives good results for the proposed application.

- Distance Threshold - 10
- Point Color Threshold - 6
- Region Color Threshold - 5
- Minimum Cluster Size - 600

Since the results from this step is used to verify the previously performed pro-cam calibration, care must be taken to avoid errors. The following aspects could improve the accuracy of results. An elliptical shape with a strong color (Figure 4.10a) is used as the target. Any shapes can be used, however the color is an important aspect. The color of the shape must be stronger than the area around it. The color over the area of the shape should be homogeneous. The surface (paper used) of the shape must be non-reflective to avoid errors while capturing the scene. Any reflective material on the scene can cause errors. It is suggested to use sufficiently larger shape dimensions as we set the minimum cluster size to 600. As discussed in the section 3.3.1.1, pre-heating of the device should be done in order to avoid errors in depth measurements. It is also advised to capture the image on low light as some lights change the appearance of the colors. The result of the segmentation process is shown in figure 4.10b.



(a) Before



(b) After

Figure 4.10: Point cloud of the scene before and after the color based region growing segmentation

The cluster representing the shape is extracted (Figure 4.11) and the centroid of the shape is calculated using compute3DCentroid function of Common library of PCL. This gives the coordinates of the center of the shape with respect the Kinect sensor which is then used as the origin of the OptiTrack system. Since the Kinect sensor is fixed parallel to the ground, the orientation of the obtained shape will be parallel to the Kinect sensor. Only the orientation around the blue normal axis (z) will be different. This difference can be compensated while aligning the markers in the next step. Hence placing the Kinect sensor parallel to the ground is critical for this method.

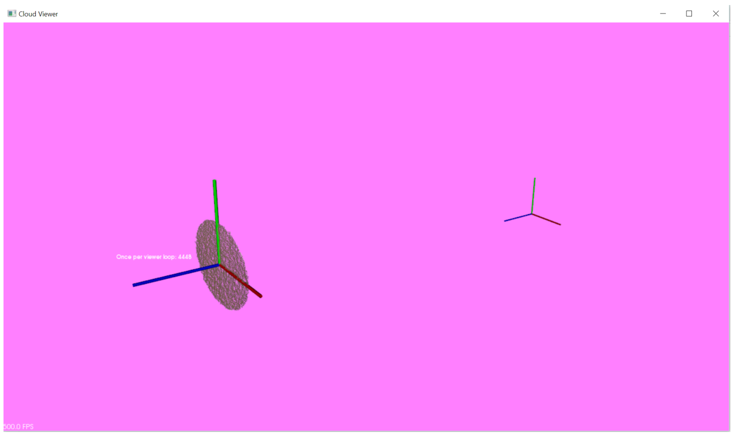


Figure 4.11: Extracted point cloud of the ellipse

4.2.3 Unity 3D

Unity 3D application is an important tool for this method of calibration. It has the function to connect the two coordinate systems as well as to render the SAR scene. RoomAlive Toolkit for Unity is the tool that is used to import the ProCamCalibration file and create projector and Kinect cameras from it. This tool also contains a set of Unity scripts and tools that enable immersive, dynamic projection mapping experiences.

RATCalibrationData script of the toolkit enables the import of the calibration files and RATSceneSetup generates cameras using the calibration data. Default 3D models of both projector and Kinect cameras are available with the tool which makes it easily identifiable. The process of setting up the scene can be found in the Git repository of RoomAlive Toolkit (Microsoft, 2017). Once the cameras are generated, an empty game object with a name 'reference' is to be created in order to set the origin of OptiTrack system (Figure 4.12). This reference object is transformed to the previously calculated centroid of the ellipse shape. The tracking data from OptiTrack is with respect to its origin that we set during before launching. If this origin is set at a known location from the Kinect sensor (The centroid of the ellipse shape), the two coordinate systems can be connected without manual coordinate transformations. In Unity environment, children object transforms with respect to its parent. Therefore, if a reference object is placed at the centroid (Origin of tracking system), the tracking data can be directly applied to the model of the object which is added as a child to the reference object.

PCL uses right hand coordinate system,on the other hand unity uses left hand coordinate system. Hence, the x axis needs to be reversed while using the data from PCL. A cross hair at the centroid perpendicular to the Kinect camera and the prefab of the soup packet (object to render on the SAR scene) at offset position from the centroid are added to the reference game object as shown in figure 4.13.

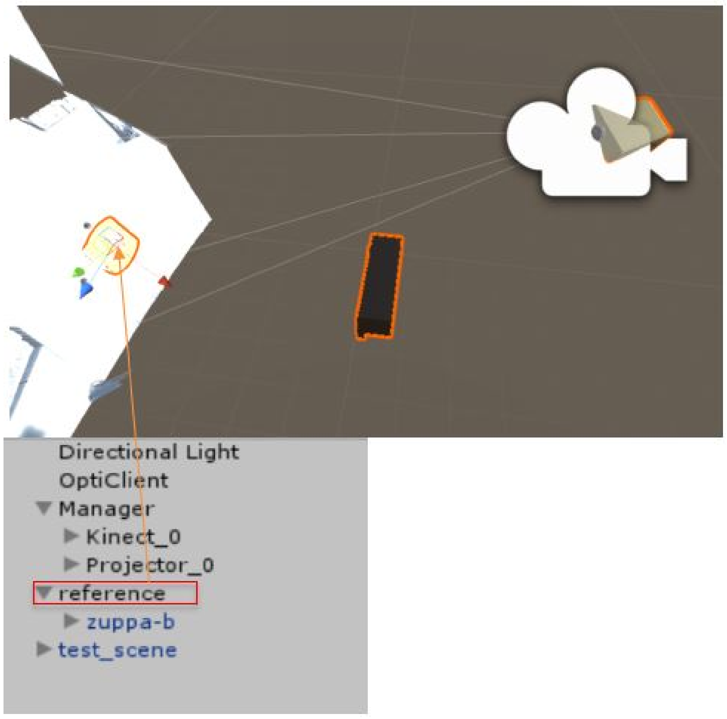


Figure 4.12: Setting the origin of tracking system in Unity3D

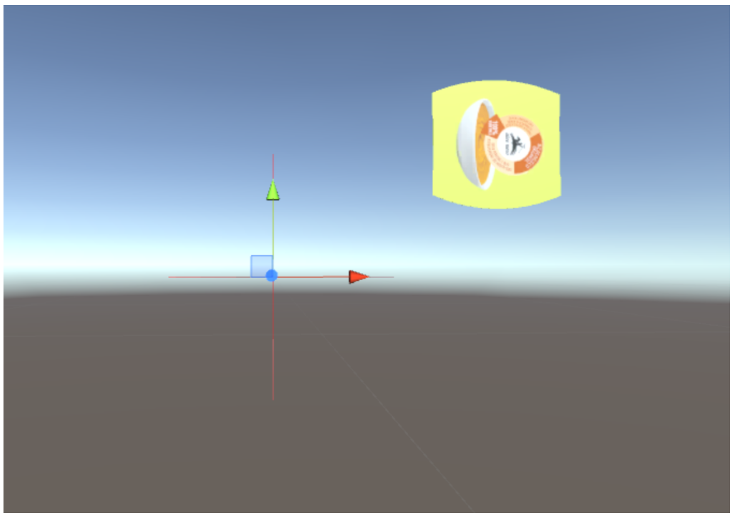
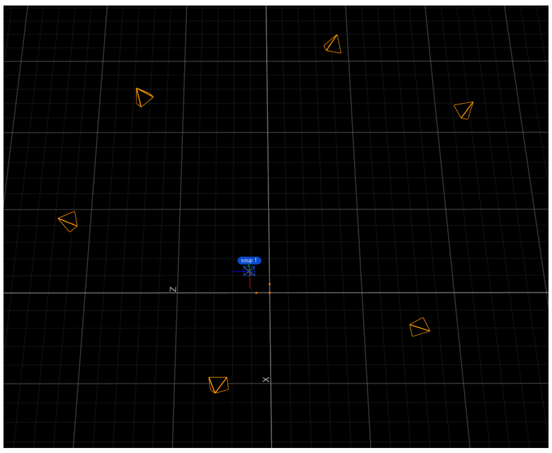


Figure 4.13: Reference game object components

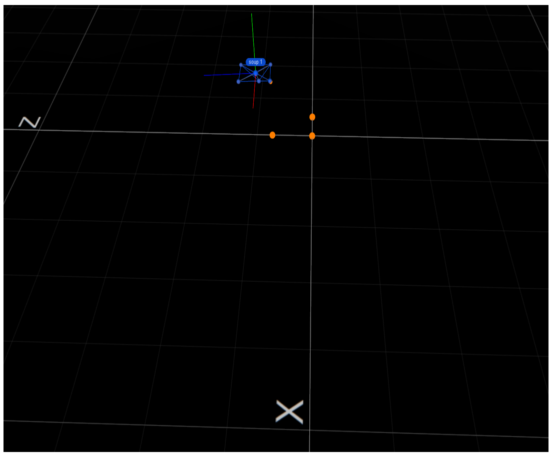
This scene is then built in order to setup a coordinate system for the OptiTrack system and to align the object of interest. The option to display the game in multiple displays are enabled and the scene is built. It is then run to setup the OptiTrack coordinate system. This step also enables the verification of the calibration results. If the center of the cross hair perfectly aligns with the center of the ellipse shape, it can be confirmed that calibration is accurate. If the conditions while performing the rgb segmentation are correctly followed, the extracted centroid coordinates should be accurate. Hence, the verification by this method can be considered as valid. In case of misalignment, repeat the pro-cam calibration to get less RMS error.

4.2.3.1 Opti-Track Motive

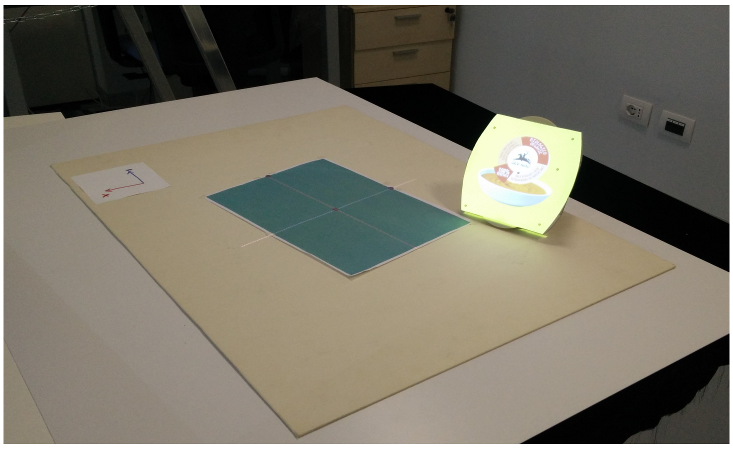
Motive is the user interface for the Opti-track system. Once the previously built scene is run successfully, the origin and the rigid body to be tracked can be set. In this study 6 Opti-track cameras (Figure 4.14a) being used. The cameras must be calibrated following the standard procedures developed by the manufacturer while installing. The object prototype with infra-red markers attached is placed inside the tracking volume in such a way that the projected image perfectly matches with the object. A plane must be prepared placing three circular markers at the vertices of a right angle triangle. This plane is then placed in the projection area such that it aligns with the cross hair projected. The marker at the 90 degree angle must align with the center of the cross hair. The origin is then set at the marker at the 90 degree angle by selecting the three markers and setting the ground plane in Motive. The placement of markers and the object is shown in figure 4.14c. A "rigid body" (4.14b) is then created from the individual markers with Motive. The object is placed aligning the projected model to match the initial orientation of the real object and the virtual object. Perfect alignment is required since small misalignment can cause larger variations while tracking initiated. Once the alignment of object and setting of the origin and rigid body is finished, the markers can be removed and the system is ready to function.



(a)



(b)



(c)

Figure 4.14: OptiTrack Motive setup

4.3 Results and Discussion

After the calibration is completed successfully, the SAR scenario can be developed and run. While developing the scene, some aspects of the calibration needs to be considered. The coordinate system of the motive and unity is not same. Hence transformations are required in order to match both coordinate systems. In the Motive, the horizontal plane (table surface) is the XZ plane with as shown in figure 4.13b. Whereas in unity the plane is XY. Since the reference object is transformed on run-time with respect to the Origin in Unity, initial transformation of this object is not performed. Hence a child object is added to the reference and transformed it matching with the OptiTrack coordinate system. When an object is transformed its children also transforms. Hence, a transformation of the model to be displayed to its initial orientation must be performed. Once the scripts importing tracking data from Motive is added to the model object, the scene can be run. The result obtained is shown in figure 4.15

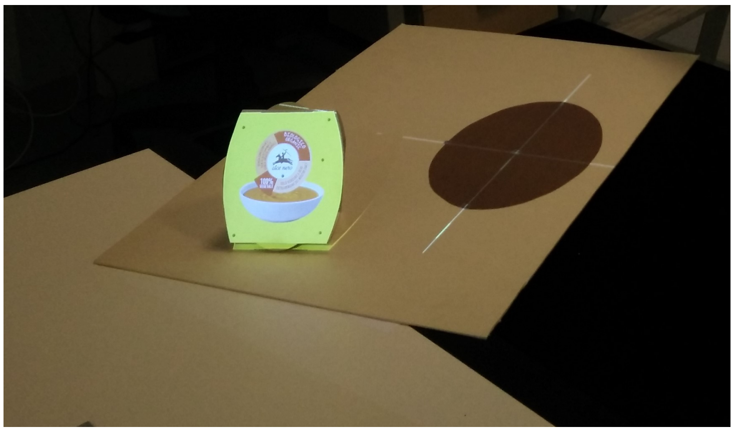


Figure 4.15: SAR scene after calibration

The pipeline of the procedure visualized as a flow chart is shown in figure 4.16. As it can be understood from the flow chart, the proposed procedure simplifies the calibration. In the currently used method for SPARK, a checker board needs to be prepared and positioned in different locations and orientations for each set of calibration. This is completely avoided by the proposed method. It doesn't require any checker board preparation and positioning. Once the devices are fixed, the calibration is completely automatic and fast. A non planar projection surface is required once to extract the intrinsics of the projector which can be done easily by placing cardboard boxes. One of the key advantage is that multiple projectors can be calibrated simultaneously. For the SPARK platform, the two projectors can be calibrated using just one Kinect sensor. Where as in the current method each projector requires one camera and the whole procedure needs to repeat for each projector. Hence the proposed method reduces time significantly.

The calibration data can be easily imported and the projector camera system is automatically created with just one click. The RoomAlive toolkit for unity enables this easy procedure. This is an important advantage when more number projectors are involved. In order to use the tracking data, the current method needs to prepare the chessboard with markers and perform another set of complex calibration procedure. Transformations of the coordinate system is required before using the tracking data. On the other hand, the proposed method is simple to execute by preparing a shape and running a program. The program generates a text file that can be easily imported to Unity 3D. Since the calibration procedure connects the tracking and projection systems inside unity, the development of the SAR scene will be easier as it can be developed inside the same calibration project.

Despite the fact that the proposed system simplifies the whole calibration procedure, it suffers from some disadvantages. The method restricts its use in SAR systems with OptiTrack system as it is developed for SPARK platform. The placement of the devices satisfying the requirements and the parallel alignment of Kinects sensor can be exhausting. In some cases, it would be difficult to satisfy the requirements because of the size difference of projected image and view area of Kinect and space limitations. However, the initial preparation is necessary and required for all the calibration methods. As it can be seen from figure 4.15, the results of the calibration is not so accurate. Many factors cause this issue. Although the accuracy of the Kinect sensor improved significantly in the version 2, the depth data from the sensor still deviates from the actual value (Yang et al., 2015). Pre-heating of the device and making the work area at the center of the view plane reduces the errors however, highly accurate results are impossible with the current hardware technology. There is no standard method developed to verify the calibration results as the RoomAlive toolkit is designed for large rooms where accurate results are not required. The verification of the pro-cam calibration results are done by changing the projectors matrix of graphics camera. It is also difficult to obtain a RMS error less than 0.5. In the proposed method, the verification is done by considering that the results from RGB segmentation is accurate. As mentioned before regarding the accuracy of the Kinect sensor, the values of centroid obtained may not be accurate. However, results with acceptable error are obtained from PCL program. Errors can also be generated while placing the markers and aligning the object. This type of errors can only be avoided by proper placement of markers and alignment of the object with manual adjustment and corrections. A solution for correcting and adjusting the calibration results without repeating the previous steps still needs to be developed. Since this calibration method significantly utilizes Unity3D application, only people with at least basic knowledge of this application can perform this calibration procedure. A calibration with an acceptable final error is not achieved with this method. Therefore, it cannot be used for SAR systems with small non-static objects. Nonetheless, it could be used in SAR systems with larger sized objects where these errors can be ignored.

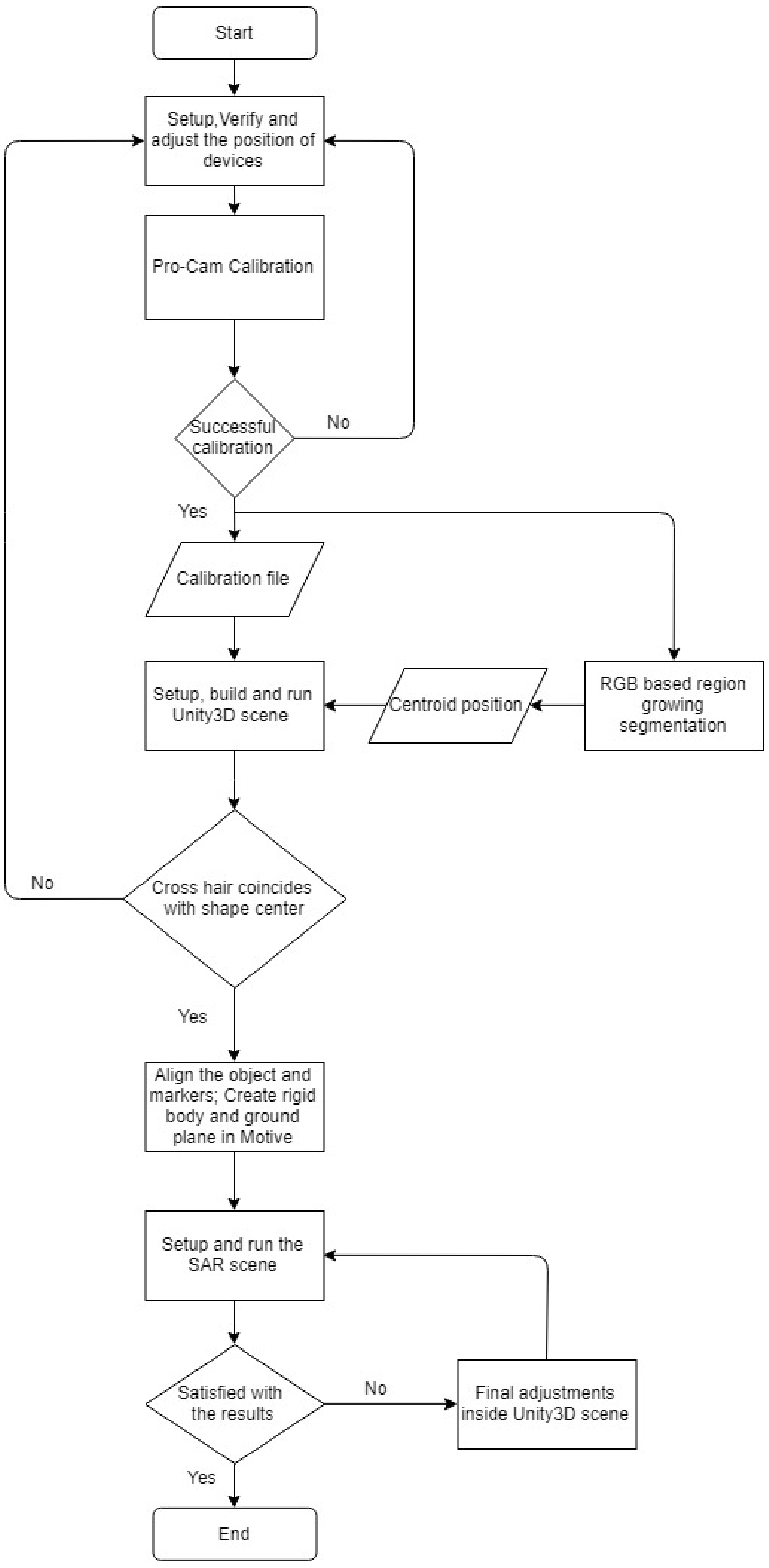

Start
Setup,Verify and adjust the position of devices
Pro-Cam Calibration
Successful calibration
No
Yes
Calibration file
Setup, build and run Unity3D scene
Centroid position
RGB based region growing segmentation
Cross hair coincides with shape center
No
Yes
Align the object and markers; Create rigid body and ground plane in Motive
Setup and run the SAR scene
Satisfied with the results
No
Final adjustments inside Unity3D scene
Yes
End

Figure 4.16: Flow chart of the proposed calibration procedure

Chapter 5

Conclusion and future work

This thesis has analyzed the optical tracking system used for the SPARK platform and proposed two tracking solutions to solve the issues found. The implementation and analysis of the two proposed methods are performed and inferences are drawn from the results obtained. An attempt to develop an easy calibration procedure for the optical tracking system is performed with the aim of solving one of its major drawback. The conclusions drawn from the study and the suggestions for future works are explained below.

5.1 Conclusions

SAR systems with manipulable objects require robust and accurate tracking. Compared to other methods available, the optical tracking system with small infra-red markers provides good results. However, the cost, use of markers and the complex calibration procedure makes the need for the development of an alternative tracking solutions necessary. Although, the use of invisible infra-red ink and small sized markers reduces the issue due to the presence of markers, these solutions failed to avoid the problems completely.

An approach using flat base tray with markers is proposed as a the first solution. Due to the interference of the markers and the projected images, marker-based approaches are not suggested for SAR systems. However, this solution enables low cost and effective marker-based tracking for SAR systems without affecting the quality of projected images. This solution significantly reduces the cost of the system and complexity of calibration procedure. However, the proposed system failed to provide robust and accurate tracking. Desired results are obtained after manual adjustments and approximated calculations. The presence of tray makes the manipulation of the object difficult and position of the camera causes poor initial detection of the markers. An attempt to solve the initial detection issue by placing the camera in different position, is being obstructed by the calibration method used. The system also suffers from the common drawback of the marker-based approach. Cameras fail to track fast movements of the markers. Hence, this method of tracking cannot be applied without solving the problems mentioned.

The second proposed tracking solution based on point cloud and particle filter provides a promising solution for the future. The system uses the easiest calibration method and low cost devices for tracking. The Point Cloud Library (PCL) used to implement the tracking system, allows the color and depth data from the Kinect sensor to be manipulated for a a large variety of applications. The system doesn't require any marker to be attached on the object. Calibration of the prototype can also be avoided using this method. Although, the system is able to track a texture-less white object, the accuracy and stability of tracking is poor due to hardware limitations and absence of optimization programs. Optimization of tracking parameters improved the performance. However, solutions must be developed to stabilize the results and improve the accuracy before utilizing it for SAR systems.

Since a new and complete tracking solution is not found, attempts to solve the drawbacks of existing optical tracking method are performed. From the attempt of implementing particle filter tracking with PCL using Kinect sensor, the calibration procedure based on color and depth data is identified as the easiest. Hence, a procedure is developed to find relate the tracking unit with the calibrated projection unit. The proposed calibration procedure significantly simplifies the calibration of the SAR system by avoiding human interventions and coordinate transformations. Nevertheless, the procedure doesn't result in accurate calibration of the system. A valid verification of calibration results and tools to correct the errors without repeating the procedure is missing. By manual adjustments of the values obtained, approximate results can be obtained. The method proposed is not able to avoid the calibration of prototype as well. Hence, valid verification methods must be developed to check the accuracy of calibration result at each step and provisions must be made to correct the errors without repeating the whole procedure.

5.2 Future works

In order to avoid the issue of poor initial detection of markers and loss of tracking, the camera should be positioned at a position where it perfectly views the marker at every position of the tray. Since the current method doesn't allow the placement of the camera far from projector, a new calibration method should be adopted. A calibration method that gives accurate results even when the camera is far from the projector could solve the issue. Another possible solution is to utilize RGB camera of Kinect sensor to track the markers. With the calibration method available, the device can be placed anywhere satisfying the requirements. However, current plugins allow the Kinect to work only as a web-camera in Windows. The EasyAR package of unity doesn't support Kinect as a render camera. Hence, adding plugins to EasyAR package to allow the use of Kinect as a render camera could also solve the issue of poor initial detection. Instead of using a flat base tray, a properly designed tray with inclined surfaces in such a way that, at least one marker is perfectly visible at every position of the tray is also a possible solution. Future releases of open source AR platforms specially developed for SAR could avoid the manual adjustments and calculations performed to obtain real world coordinates.

The major issue persisted during the implementation of the particle filter tracking using PCL is the poor stability of tracking results. This problem is due to the hardware limitations and the absence of optimization programs. As Microsoft stopped the production of Kinect sensors, advanced depth sensors like Intel RealSense are becoming popular. The PCL grabber program that extracts the streaming data from the RealSense device is still in development. With the release of the RealSense grabber, use of these advanced depth sensors for the tracking will be possible. This could improve the results. However, an easy and robust calibration method needs to be developed for the same. Optimization of particle filter tracking output could also improve the stability. The attempt of taking simple moving average of tracking output resulted in large errors. An optimized approach of stabilizing the output can be developed with the help of statistical algorithms. Another possible solution is to utilize the object recognition program of PCL. The program is able search and retrieve the position and orientation of an object point cloud inside the point cloud of the scene taken. Hence, when the object changes its position kept stationary, the new position and orientation of the object can be obtained and the virtual object can be updated. However, this program is not optimized for real-time performance like the particle filter tracking. Therefore, optimization of the PCL object recognition program for real-time calculations could be performed to adopt a possible solution to the stability issues of particle filter tracking.

Since the calibration procedure developed is based on the open source Pro-Cam calibration tool from Microsoft, further developments in the tool can be performed to add verification methods and provisions for correcting the errors without repeating the whole procedure.

Bibliography

Ababsa, F. and Mallem, M. (2011). Robust camera pose tracking for augmented reality using particle filtering framework. *Machine Vision and applications*, 22(1):181–195.

Agarwal, C. and Thakur, N. (2014). The evolution and future scope of augmented reality. *International Journal of Computer Science Issues (IJCSI)*, 11(6):59.

Ahad, A. R. and Hossain, S. (2004). Augmented reality and its challenges. In *SICE 2004 Annual Conference*, volume 2, pages 1041–1043. IEEE.

Akkaladevi, S., Ankerl, M., Heindl, C., and Pichler, A. (2016). Tracking multiple rigid symmetric and non-symmetric objects in real-time using depth data. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 5644–5649. IEEE.

Azad, P., Münch, D., Asfour, T., and Dillmann, R. (2011). 6-dof model-based tracking of arbitrarily shaped 3d objects. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5204–5209. IEEE.

Azuma, R. (1993). Tracking requirements for augmented reality. *Communications of the ACM*, 36(7):50–51.

Azuma, R. T. (1997). A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385.

Baillot, Y., Davis, L., and Rolland, J. (2001). A survey of tracking technology for virtual environments. *Fundamentals of wearable computers and augumented reality*, page 67.

Bajura, M. and Neumann, U. (1995). Dynamic registration correction in video-based augmented reality systems. *IEEE Computer Graphics and Applications*, 15(5):52–60.

Benko, H., Wilson, A. D., and Zannier, F. (2014). Dyadic projected spatial augmented reality. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 645–655. ACM.

Bimber, O. and Raskar, R. (2005). *Spatial augmented reality: merging real and virtual worlds*. CRC press.

Bryson, S. T. (1992). Measurement and calibration of static distortion of position data from 3d trackers. In *Stereoscopic Displays and Applications III*, volume 1669, pages 244–256. International Society for Optics and Photonics.

Bukey, C. M., Kulkarni, S. V., and Chavan, R. A. (2017). Multi-object tracking using kalman filter and particle filter. In *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, pages 1688–1692. IEEE.

Calife, D., Tomoyose, A., Spinola, D., Bernardes, J., Tori, R., et al. (2007). Robot arena: Infrastructure for applications involving spatial augmented reality and robots. In *In Proceedings of the IX Symposium on Virtual and Augmented Reality*. Citeseer.

Cebulla, A. (2013). Projection-based augmented reality. *ETH Zurich*.

Chan, L. K. and Lau, H. Y. (2010). A tangible user interface using spatial augmented reality. In *3D User Interfaces (3DUI), 2010 IEEE Symposium on*, pages 137–138. IEEE.

Chen, P., Dang, Y., Liang, R., Zhu, W., and He, X. (2018). Real-time object tracking on a drone with multi-inertial sensing data. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):131–139.

Choi, C. and Christensen, H. I. (2010). Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4048–4055. IEEE.

Choi, C. and Christensen, H. I. (2013). Rgb-d object tracking: A particle filter approach on gpu. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1084–1091. IEEE.

de Figueiredo, R. P., Moreno, P., Bernardino, A., and Santos-Victor, J. (2013). Multi-object detection and pose estimation in 3d point clouds: A fast grid-based bayesian filter. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4250–4255. IEEE.

Dünser, A., Grasset, R., and Billinghurst, M. (2008). *A survey of evaluation techniques used in augmented reality studies*. Human Interface Technology Laboratory New Zealand.

Dunston, P. S. et al. (2008). Identification of application areas for augmented reality in industrial construction based on technology suitability. *Automation in Construction*, 17(7):882–894.

Foxlin, E. (1996). Inertial head-tracker sensor fusion by a complementary separate-bias kalman filter. In *Virtual Reality Annual International Symposium, 1996., Proceedings of the IEEE 1996*, pages 185–194. IEEE.

Franz, A. M., Haidegger, T., Birkfellner, W., Cleary, K., Peters, T. M., and Maier-Hein, L. (2014). Electromagnetic tracking in medicine—a review of technology, validation, and applications. *IEEE transactions on medical imaging*, 33(8):1702–1725.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., and Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292.

Ghazisaedy, M., Adamczyk, D., Sandin, D. J., Kenyon, R. V., and DeFanti, T. A. (1995). Ultrasonic calibration of a magnetic tracker in a virtual reality space. In *Virtual Reality Annual International Symposium, 1995. Proceedings.*, pages 179–188. IEEE.

Harris, C. and Stennett, C. (1990). Rapid-a video rate object tracker. In *BMVC*, pages 1–6.

Hartmann, J. and Vogel, D. (2018). An evaluation of mobile phone pointing in spatial augmented reality. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, page LBW122. ACM.

Held, D., Levinson, J., and Thrun, S. (2013). Precision tracking with sparse 3d and dense color 2d data. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1138–1145. IEEE.

Held, R., Gupta, A., Curless, B., and Agrawala, M. (2012). 3d puppetry: a kinect-based interface for 3d animation. In *UIST*, pages 423–434. Citeseer.

Hoff, W. A., Nguyen, K., and Lyon, T. (1996). Computer-vision-based registration techniques for augmented reality. In *Intelligent Robots and Computer Vision XV: Algorithms, Techniques, Active Vision, and Materials Handling*, volume 2904, pages 538–549. International Society for Optics and Photonics.

Hu, X., Tang, Y., and Zhang, Z. (2008). Video object matching based on sift algorithm. In *Neural Networks and Signal Processing, 2008 International Conference on*, pages 412–415. IEEE.

Irlitti, A. and Von Itzstein, S. (2013). Validating constraint driven design techniques in spatial augmented reality. In *Proceedings of the Fourteenth Australasian User Interface Conference-Volume 139*, pages 63–72. Australian Computer Society, Inc.

Johnson, A. S. and Sun, Y. (2013). Exploration of spatial augmented reality on person. In *Virtual Reality (VR), 2013 IEEE*, pages 59–60. IEEE.

Jones, B., Sodhi, R., Murdock, M., Mehra, R., Benko, H., Wilson, A., Ofek, E., MacIntyre, B., Raghuvanshi, N., and Shapira, L. (2014). Roomalive: magical experiences enabled by scalable, adaptive projector-camera units. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 637–644. ACM.

Kato, H., Billinghurst, M., Poupyrev, I., Imamoto, K., and Tachibana, K. (2000). Virtual object manipulation on a table-top ar environment. In *Augmented Reality, 2000.(ISAR 2000). Proceedings. IEEE and ACM International Symposium on*, pages 111–119. Ieee.

Kim, K., Lepetit, V., and Woo, W. (2010). Keyframe-based modeling and tracking of multiple 3d objects. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, pages 193–198. IEEE.

Kyriazis, N. and Argyros, A. (2013). Physically plausible 3d scene tracking: The single actor hypothesis. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 9–16. IEEE.

Lang, P., Kusej, A., Pinz, A., and Brasseur, G. (2002). Inertial tracking for mobile augmented reality. In *Instrumentation and Measurement Technology Conference, 2002. IMTC/2002. Proceedings of the 19th IEEE*, volume 2, pages 1583–1587. IEEE.

Laviole, J., Thévin, L., Albouys-Perrois, J., and Brock, A. (2018). Nectar: Multi-user spatial augmented reality for everyone. In *VRIC 2018, ACM Virtual Reality International Conference.* ACM.

Lee, J. C., Dietz, P. H., Maynes-Aminzade, D., Raskar, R., and Hudson, S. E. (2004). Automatic projector calibration with embedded light sensors. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 123–126. ACM.

Li, Q., Li, Y., Tian, J., and Ren, D. (2017). Augmented reality registration method based on natural feature points. In *Communication Technology (ICCT), 2017 IEEE 17th International Conference on*, pages 1937–1941. IEEE.

Lindlbauer, D., Grønbæk, J. E., Birk, M., Halskov, K., Alexa, M., and Müller, J. (2016). Combining shape-changing interfaces and spatial augmented reality enables extended object appearance. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 791–802. ACM.

Livingston, M. A., Garrett, W. F., Hirota, G., Whitton, M. C., Pisano, E. D., Fuchs, H., et al. (1996). Technologies for augmented reality systems: Realizing ultrasound-guided needle biopsies. In *Proceedings of the 23rd annual conference on computer graphics and interactive techniques*, pages 439–446. ACM.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Marner, M. R., Smith, R. T., Porter, S. R., Broecker, M. M., Close, B., and Thomas, B. H. (2011). Large scale spatial augmented reality for design and prototyping. In *Handbook of Augmented Reality*, pages 231–254. Springer.

Marner, M. R. and Thomas, B. H. (2014). Spatial augmented reality user interface techniques for room size modelling tasks. In *Proceedings of the Fifteenth Australasian User Interface Conference-Volume 150*, pages 39–45. Australian Computer Society, Inc.

Marner, M. R., Thomas, B. H., and Sandor, C. (2009). Physical-virtual tools for spatial augmented reality user interfaces. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 205–206. IEEE.

Mast, D., Bosman, M., Schipper, S., Diederiks, S., and de Vries, S. (2017). Adding interactivity to balansar: a spatial augmented reality game for balancing in physical education. In *Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play*, pages 383–389. ACM.

Mellor, J. P. (1995). Enhanced reality visualization in a surgical environment.

Menk, C. and Koch, R. (2013). Truthful color reproduction in spatial augmented reality applications. *IEEE transactions on visualization and computer graphics*, 19(2):236–248.

Microsoft (2017). Microsoft/roomalivetoolkit.

Miyazaki, D. and Hashimoto, N. (2018). Dynamic projection mapping onto non-rigid objects with dot markers. In *Advanced Image Technology (IWAIT), 2018 International Workshop on*, pages 1–4. IEEE.

Moreno, D. and Taubin, G. (2012). Simple, accurate, and robust projector-camera calibration. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 464–471. IEEE.

Morikubo, Y. and Hashimoto, N. (2017). Marker-less real-time tracking of texture-less 3d objects from a monocular image. In *SIGGRAPH Asia 2017 Posters*, page 50. ACM.

Naimark, L. and Foxlin, E. (2002). Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 27. IEEE Computer Society.

Nakamura, T., De Sorbier, F., Martedi, S., and Saito, H. (2012). Calibration-free projector-camera system for spatial augmented reality on planar surfaces. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 85–88. IEEE.

Narzt, W., Pomberger, G., Ferscha, A., Kolb, D., Müller, R., Wieghardt, J., Hörtner, H., and Lindinger, C. (2006). Augmented reality navigation systems. *Universal Access in the Information Society*, 4(3):177–187.

Neumann, U. and Cho, Y. (1996). A self-tracking augmented reality system. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 109–115.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE.

Park, Y., Lepetit, V., and Woo, W. (2011). Texture-less object tracking with online training using an rgb-d camera. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 121–126. IEEE.

Pausch, R., Crea, T., and Conway, M. (1992). A literature survey for virtual environments: Military flight simulator visual systems and simulator sickness. *Presence: Teleoperators & Virtual Environments*, 1(3):344–363.

Porter, S. R., Marner, M. R., Smith, R. T., Zucco, J. E., and Thomas, B. H. (2010). Validating spatial augmented reality for interactive rapid prototyping. In *Mixed and augmented reality (ISMAR), 2010 9th IEEE international symposium on*, pages 265–266. IEEE.

Prince, S. J., Xu, K., and Cheok, A. D. (2002). Augmented reality camera tracking with homographies. *IEEE Computer graphics and Applications*, 22(6):39–45.

Raab, F. H., Blood, E. B., Steiner, T. O., and Jones, H. R. (1979). Magnetic position and orientation tracking system. *IEEE Transactions on Aerospace and Electronic systems*, (5):709–718.

Rabbi, I. and Ullah, S. (2013). A survey on augmented reality challenges and tracking. *Acta graphica: znanstveni časopis za tiskarstvo i grafičke komunikacije*, 24(1-2):29–46.

Rabbi, I., Ullah, S., and Khan, S. U. (2012). Augmented reality tracking techniques—a systematic literature. *IOSR Journal of Computer Engineering (IOSRJCE)*, 2(2):23–29.

Raskar, R., Welch, G., Low, K.-L., and Bandyopadhyay, D. (2001). Shader lamps: Animating real objects with image-based illumination. In *Rendering Techniques 2001*, pages 89–102. Springer.

Ren, C., Prisacariu, V., Kähler, O., Reid, I., and Murray, D. (2017). Real-time tracking of single and multiple objects from depth-colour imagery using 3d signed distance functions. *International Journal of Computer Vision*, 124(1):80–95.

Ridel, B., Reuter, P., Laviole, J., Mellado, N., Couture, N., and Granier, X. (2014). The revealing flashlight: Interactive spatial augmented reality for detail exploration of cultural heritage artifacts. *Journal on Computing and Cultural Heritage (JOCCH)*, 7(2):6.

Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *Robotics and automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE.

Sánchez, J. R., Álvarez, H., and Borro, D. (2010). Towards real time 3d tracking and reconstruction on a gpu using monte carlo simulations. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, pages 185–192. IEEE.

Siltanen, S. (2012). *Theory and applications of marker-based augmented reality.* VTT.

Simon, G. (2011). Tracking-by-synthesis using point features and pyramidal blurring. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 85–92. IEEE.

Smith, R. T., Webber, G., Sugimoto, M., Marner, M., and Thomas, B. H. (2013). Automatic sub-pixel projector calibration. *ITE Transactions on Media Technology and Applications*, 1(3):204–213.

Song, S., Qiu, X., and Meng, M. Q.-H. (2017). An improved 6d pose detection method based on multiple magnets tracking. In *SENSORS, 2017 IEEE*, pages 1–3. IEEE.

Sutherland, I. E. (1968). A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 757–764. ACM.

Ullah, S. (2011). *Multi-modal Interaction in Collaborative Virtual Environments: Study and analysis of performance in collaborative work.* PhD thesis, Université d'Evry-Val d'Essonne.

Van Krevelen, D. and Poelman, R. (2010). A survey of augmented reality technologies, applications and limitations. *International journal of virtual reality*, 9(2):1.

Viyanon, W., Songsuittipong, T., Piyapaisarn, P., and Sudchid, S. (2017). Ar furniture: Integrating augmented reality technology to enhance interior design using marker and markerless tracking. In *Proceedings of the 2nd International Conference on Intelligent Information Processing*, page 32. ACM.

Voida, S., Podlaseck, M., Kjeldsen, R., and Pinhanez, C. (2005). A study on the manipulation of 2d objects in a projector/camera-based augmented reality environment. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 611–620. ACM.

Wasenmüller, O. and Stricker, D. (2016). Comparison of kinect v1 and v2 depth images in terms of accuracy and precision. In *Asian Conference on Computer Vision*, pages 34–45. Springer.

Willis, K. D., Poupyrev, I., Hudson, S. E., and Mahler, M. (2011). Sidebyside: ad-hoc multiuser interaction with handheld projectors. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 431–440. ACM.

Yang, L., Zhang, L., Dong, H., Alelaiwi, A., and El Saddik, A. (2015). Evaluating and improving the depth accuracy of kinect for windows v2. *IEEE Sensors Journal*, 15(8):4275–4285.

Yuheng Ren, C., Prisacariu, V., Murray, D., and Reid, I. (2013). Star3d: Simultaneous tracking and reconstruction of 3d objects using rgb-d data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1561–1568.

Zhan, Q., Liang, Y., and Xiao, Y. (2009). Color-based segmentation of point clouds. *Laser scanning*, 38(3):155–161.

Zhou, F., Duh, H. B.-L., and Billinghurst, M. (2008). Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 193–202. IEEE Computer Society.

Zhou, H., Yuan, Y., and Shi, C. (2009). Object tracking using sift features and mean shift. *Computer vision and image understanding*, 113(3):345–352.

Zhou, J., Lee, I., Thomas, B., Menassa, R., Farrant, A., and Sansome, A. (2011). Applying spatial augmented reality to facilitate in-situ support for automotive spot welding inspection. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, pages 195–200. ACM.

Acknowledgements

The work reported in this thesis has been supported by the SPARK project, funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 688417. The author would like to thank all the partners of the SPARK Consortium for the extensive support and valuable contribution provided in the development of this activity.