Automatic and Adaptable Registration of Live RGBD Video Streams Sharing Partially Overlapping Views

by

© *Afsaneh Rafighi*

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of *Sciences*

Department of *Computer Science*

Memorial University of Newfoundland

October 2015

St. John's Newfoundland

Abstract

In this thesis, we introduce DeReEs-4v, an algorithm for unsupervised and automatic registration of two video frames captured depth-sensing cameras. DeReEs-4V receives two RGBD video streams from two depth-sensing cameras arbitrary located in an indoor space that share a minimum amount of 25% overlap between their captured scenes. The motivation of this research is to employ multiple depth-sensing cameras to enlarge the field of view and acquire a more complete and accurate 3D information of the environment. A typical way to combine multiple views from different cameras is through manual calibration. However, this process is time-consuming and may require some technical knowledge. Moreover, calibration has to be repeated when the location or position of the cameras change.

In this research, we demonstrate how DeReEs-4V registration can be used to find the transformation of the view of one camera with respect to the other at interactive rates. Our algorithm automatically finds the 3D transformation to match the views from two cameras, requires no human interference, and is robust to camera movements while capturing. To validate this approach, a thorough examination of the system performance under different scenarios is presented. The system presented here supports any application that might benefit from the wider field-of-view provided by the combined scene from both cameras, including applications in 3D telepresence, gaming, people tracking, videoconferencing and computer vision.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Oscar Meruvia-Pastor, for his constant support and guidance, dedication of time and knowledge and also his patience and motivation during my studies here at Memorial University. This thesis would not have been possible without his encouragement and supervision.

Second, I would like to thank my friend and co-researcher Sahand Seifi for his support and thoughtful comments and Dr. Lourdes Peña-Castillo for her generous time in assisting me with my evaluations. Also, my gratitude extended to Computer Science General office including Ms. Darlene Oliver, Ms. Brenda Hiller and Ms. Sharon Deir, for their support during my masters study. I would like to take this opportunity to express my deep happiness for having great friends at MUN, without whom this past two years would have not been as adventurous and enjoyable as it was.

Last but not least, I would like to thank my family, especially my parents Ebrahim and Pary who never stopped loving and supporting me. I owe my achievements, rather small, to them for they believed in me ever since I was a child. My heartfelt appreciation to my friend and husband Mojtaba for his constant support and for knowing how to cheer me up in tough times and loving me all the way through.

Contents

List of Tables

List of Figures

Acronyms

- **DeReEs:** Detection, Rejection and Estimation
- **DeReEs-4V:** DeReEs for Video
- **PCL:** Point Cloud Library
- **OpenCV library:** Open Source Computer Vision library
- **RGBD:** RGB + Depth
- **PCD:** Point Cloud Data
- **ToF:** Time of Flight
- **CW:** Continuous Wave
- **ICP:** Iterative Closest Point
- **RANSAC:** Random Sample Consensus
- **GPU:** Graphics Processing Unit
- **NAN:** Not A Number
- **IR:** Infra Red
- **SLAM:** Simultaneous Localization And Mapping
- **SURF:** Speeded Up Robust Features
- **ORB:** Oriented FAST and Rotated BRIEF

Chapter 1

Introduction

The introduction of depth-sensing cameras has transformed computer vision research. These cameras provide depth maps where each pixel shows the distance of the point in the scene to the camera. Initial 3D cameras such as laser scanners and sonars are highly accurate, however, they are expensive and slow in providing 3D information [35]. The arrival of RGB-D sensors that provide color and depth images, such as Microsoft Kinect in November 2010, has led to a rapid change in research and industry, since they come at an affordable price and are appropriate for real time applications. Although its initial purpose was for gesture recognition and human detection in gaming, soon researchers realized its role in many different areas.

Applications include 3D scanning (such as [46] or [28]), in which a camera is moved around a scene or object to acquire a 3D model of it, indoor 3D mapping and localization, which is simultaneously calculating the position and orientation of the depth-sensing cameras and providing 3D maps in real-time(e.g [72]), object/human detection and tracking in images and videos, and gesture recognition. Another im-

portant application, which is also the main focus of this research, is telepresence. Telepresence in this case refers to connecting two remote parties and their environments through a window or screen in a way that users experience the feeling of being present in the remote environment.

The predominant technologies supporting these cameras are "Structured Light" and "Time of Flight". Time of Flight(ToF) can be measured either by using pulse or continuous wave modulation(CW) [24]. Sensors using pulse modulation find depth by measuring the time a light pulse trip takes. Whereas in CW modulations, an IR is emitted to the object and a sensor detects the reflected light. In this case, depth is measured by calculating the phase difference between the emitted and the reflected wave. In Structured Light cameras, a structured light with a known pattern is casted on the environment and the reflection of the pattern is captured by a sensing device. Depth is obtained by analyzing the distortion of the pattern. The technology behind the first generation of the Kinect(2010) is Structured Light, while the second generation Kinects are based on ToF.

While off-the-shelf RGB-D sensors assist us in different areas of Computer Vision, these cameras suffer from artifacts such as missing depth data. Regardless of the technology they are based on, holes are a common artifact in depth maps. In Microsoft Kinect cameras, objects and surfaces that are shiny and bright cause missing information in the depth values. The depth maps suffer from quantization noise, when the resolution decreases and distance of the object increases [36]. Moreover, holes are found due to the shadows objects cast on other objects as a result of occlusion. This situation arises because of the misplacement of the IR projector and sensor. An IR light is emitted on the scene and the area behind an object will not be cast by the

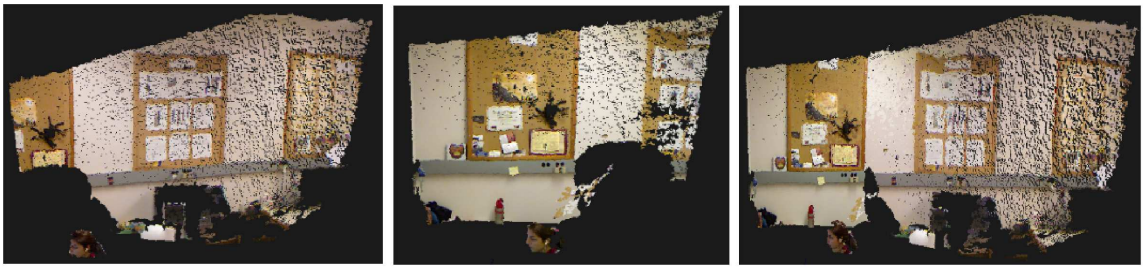


Figure 1.1: Effect of having a multi-depth sensor system. Areas occluded by each of the cameras,(a) and (b), are complete in the registered point cloud(c).

light, however, the IR projector and sensor are located separately by a small baseline. Thus, the IR sensor is observing the occluded area that IR projector was unable to cast the light on. This leads to missing values in depth. As shown in Figure 1.1, these holes are the missing depth information which are shown in black.

RGB-D cameras also suffer from limited field of view. Kinect cameras,for instance, have a range of 57 degree horizontal and 43 degree vertical field of view. This limited field of view led researchers to explore the use of multiple depth-sensing cameras simultaneously. Applying multiple depth-sensing cameras in scenarios such as telepresence helps capture a larger portion of the environment.(e.g Figure 1.2 shows the field of view of one camera compared with two calibrated cameras). With the help of another camera, areas which are occluded by one camera can be captured by the other(e.g. the left corner of Figure 1.1(a) is not captured due to the shadow of the user in front of camera one. The second camera, Figure 1.1(b), has captured the space and the registered point cloud, Figure 1.1(c), is complete). It also has applications in gaming, where users can add cameras to involve more users in the game. Multiple

depth sensing camera scenarios are truly beneficial when the system is automatic and robust, particularly when the cameras are moved while the system is running. In chapter 2, a list of the applications of multiple depth-sensing cameras is provided.

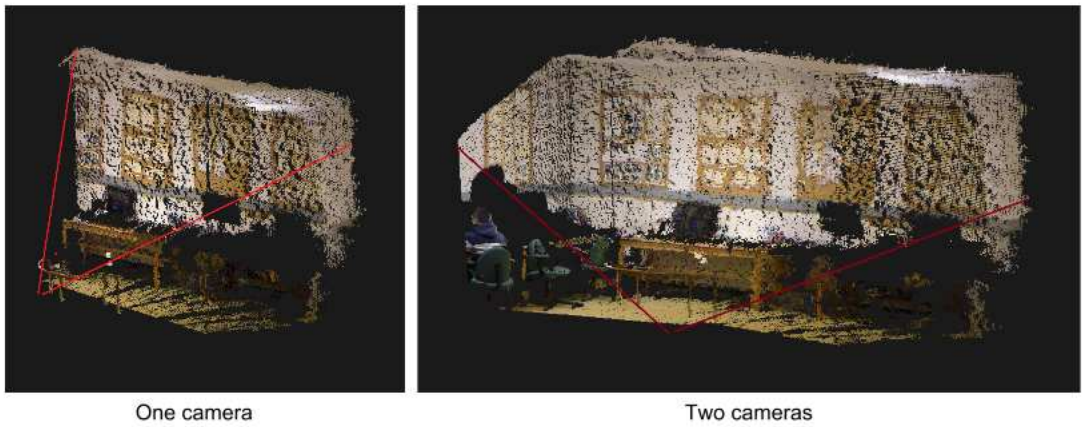


Figure 1.2: Comparison between field of view of one camera and two calibrated cameras

1.1 Motivation

This thesis presents a solution for using multiple depth-sensing cameras for telepresence and applications that might benefit from the wider operational field-of-view provided by registering the views of two cameras sharing some part of a common scene. Careful calibration of the depth-sensing cameras in any scenario that includes a multi-camera system is a necessary and initial step [70]. Once point clouds are roughly aligned by finding a coarse transformation through calibration, registration methods are then used to perform a fine alignment on them [25]. Calibration is time consuming and has to be repeated in case the cameras are moved. It is a cumbersome

task for users whether they are in a video conferencing, or part of a game; even for regular researchers, this initial stage can be frustrating and time-consuming [50].

In this research we present a client-server system that uses two Kinect sensors at server side which is constantly capturing video streams and streaming to the client side on a local network. This system should be robust to camera movements and no manual calibration stage will be required. Instead, we are introducing DeReEs-4V, a registration method that uses RGBD image pairs. DeReEs-4V is short for DeReEs for Video and is an extension to DeReEs (Detection, Rejection, Estimation) [55] which was proven to work with viewpoints(RGB+Depth imagges only) that share small overlapping sections. This is in contrast to ICP(Iterative Closest Point) where point clouds have to be relatively close and roughly aligned in advance. Further, a rigorous evaluation of its performance and robustness in maintaining point clouds' alignment under various conditions is provided. Additional contributions, described in Chapter 3, include applying ICP in the last step of the algorithm for having a more stable registration, extending the Octree Point Cloud compression method for two depth sensing cameras, and a technique to improve the quality of the registration over multiple iterations during the first seconds of use.

In this work, we are trying to find the answers to the following fundamental questions:

> "***How can we make the multiple Kinect system hassle free for users without employing calibration targets and markers?***"

Extensive research has been conducted on using multiple depth-sensing cameras, however, most of them require an initial stage of calibration to align viewpoints with respect to each other [8]. As discussed earlier, calibration methods are time con-

suming, and may require knowledge of computer vision algorithms and programming skills to be performed. Therefore, it will not benefit non-expert users who, for instance, want to employ two cameras in a gaming environment. Even for researchers, the necessity to calibrate cameras initially, and repeating it in case of camera movement, is cumbersome. For these reasons, we will be introducing DeReEs-4V to align viewpoints.

> "***Knowing that DeReEs is successful in registering two RGBD images, will it be successful in registering RGBD video streams constantly being captured by two Kinect cameras?***"

DeReEs is a feature based RGBD registration technique and its initial stage is finding the corresponding features in both images to obtain a transformation based on them. In DeReEs-4V, we will apply feature matching on each two RGB images captured from two cameras. DeReEs-4V is a multi-threaded system designed to reduce delay in every part of the system that includes capturing, registration and visualization.

> "***Can users move the cameras while viewpoints are being registered? and will the system maintain its stability in aligning viewpoints even after camera movement?***"

The main purpose of this research is to allow users to move the cameras and still maintain a registered visualization from two viewpoints. Imagine users in a video conferencing session would like to show other portions of the room to the viewer. Or in gaming, more users want to be engaged at certain positions in the room. We will show that, by employing DeReEs-4V, online camera movements are permissible, since scenes can be re-aligned in less than $63ms$.

*"**Assuming its success in registering video streams, will it perform in real-time?**"*

In terms of speed of the system, breaking several stages of the system into threads is a good alternative to achieve real-time performance. One thread is responsible for capturing RGBD streams and buffering them. Another thread applies registration on them. A separate thread is responsible for detecting camera movement and one more thread to perform visualization. The Open Souce Computer Vision library (OpenCV) [12] implementation of SURF feature matching [6] is employed using the GPU to further increase the speed. Applying these methods, we prove that our system is capable to perform in real-time.

One question that arises while in the process of research is the following:

*"**If a user decides to move the cameras, how can we make the system perceptive enough to recognize camera movement?**"*

One of the contributions of this work is in modifying DeReEs to adapt it to video streams coming from two Kinects, as opposed to individually captured images. The point clouds continue to maintain their alignment even with changes in the number of users, user movements from either cameras. Other contributions, such as detecting camera movement, are proposed and will be further described in Section 3.

The system is fast and removes the hassle of the initial calibration of the cameras. In this work, we show that users can easily move cameras around to show other portions of the environment because it is robust to camera movements and it is successful in finding the point clouds' alignment as long as a minimum overlap of approximately 25% exists between two viewpoints. The need for this system arises

when non-technical users decide to show other portions of the environment to remote users while communicating, to enhance the field of view to add people in a gaming environment, or in robotics or virtual reality applications, where users or robots might bump into an obstacle and cause cameras to move.

1.2 Methodology

The system consists of a client and a server, in which the server is the machine that registers the video input and streams the registered point clouds to the client, which displays the output at the receiving end. Two Kinect cameras are placed arbitrarily at the server side in an indoor environment in a way that an overlap of at least %25 exists between two viewpoints. RGBD streams are constantly being captured and later buffered using Point Cloud Library (PCL) [2]. RGB images are used as input for SURF video matching. OpenCV implementation of SURF feature matching is used. PCL is used to visualize point clouds both at sender and receiver. The Octree Point Cloud compression implementation in PCL is also used in streaming point clouds to the client.

The execution of DeReEs-4V is separated into different threads. No calibration method is required here, instead, DeReEs [55] is chosen which is proved to work with viewpoints with at least 25% overlap. The robustness of our system is fairly dependent on the performance of DeReEs;therefore, two scenarios are considered in the system:

- Performing registration while cameras are stable. A memory mechanism is developed to record the best transformation found so far.

- Detecting camera movements and performing registration after detecting camera movements. DeReEs-4V is again used to detect camera movements.

In order to evaluate our system, Zhang's standard calibration method [70] is used. For this purpose, implementation of RGBDemo library [13] is used to find a ground truth transformation to compare our system to. Kinect cameras are used because they are fairly compatible with all the mentioned libraries here.

1.3 Outline

The rest of this thesis is structured as follows: In Chapter 2, a review of the related multi-camera systems along with an overview on camera calibration and point cloud registration is provided. In Chapter 3, our system is presented in detail. In Chapter 4, we present an extensive evaluation of the proposed system, and Chapter 5 presents our conclusions and suggestions for extending this work.

1.4 Associated Publications

The research described in this thesis has been published in the following conferences:

- Automatic and Adaptable Registration of Live RGBD Video Streams.
  Afsaneh Rafighi, Sahand Seifi and Oscar Meruvia-Pastor.
  Paper Proceedings of the 8th International Conference on Motion in Games - ACM SIGGRAPH. Accepted for publication. November 16-18, 2015, Paris, France.

- Continuous and automatic registration of live RGBD video streams with partial overlapping views.

  Afsaneh Rafighi, Sahand Seifi and Oscar Meruvia-Pastor.

  Poster Proceedings of the 42nd International Conference and Exhibition on Computer Graphics and Interactive Techniques. ACM SIGGRAPH. August 9-13, 2015, Los Angeles, USA.

Previous work that was used as a starting point for this thesis was published in the following venues:

- DeReEs: Real-Time Registration of RGBD Images Using Image-Based Feature Detection And Robust 3D Correspondence Estimation and Refinement.

  Sahand Seifi, Afsaneh Rafighi, Oscar Meruvia Pastor.

  Paper proceedings of the International Conference on Image and Vision Computing, IVCNZ 2014. November 17-19, 2014, Hamilton, New Zealand.

- Real-Time Registration of Highly Variant Colour+Depth Image Pairs.

  Sahand Seifi, Afsaneh Rafighi, Oscar Meruvia Pastor.

  Poster Proceedings of the 40th International Graphics Interface Conference. May 6-9, 2014, Montreal, Canada.

Chapter 2

Background and Related Work

In this chapter we review the background topics of the thesis in four sections. Each section provides an introduction to the knowledge areas related to this thesis. In Section 2.1, pre-processing steps of working with RGBD cameras is provided. Next, Section 2.2 introduces calibration. Section 2.3 provides an overview of point cloud registration and two of the registration algorithms used in this thesis. Lastly, in Section 2.4, we present an overview of related multiple camera setups.

2.1 Pre-processing steps in RGB-D acquisition

In this section, we will describe common issues with regards to depth maps generated by depth-sensing cameras either in single or multiple camera set scenarios.

Depth-sensing cameras such as Microsoft Kinect can provide large volumes of data when used for video processing. This large volume of point cloud data decreases the processing time. Moreover, not all of the points are required for processing in most of the applications such as object tracking or registration. Above all, depth maps

mostly suffer from common artifact of noise or NaN (Not a Number) values. This makes the estimation of the local point clouds' characteristics, such as surface normal or changes in the curvature, difficult. In this thesis, a pre-processing step is performed to remove any NaN value from the depth image.

Another common step is Down-Sampling. Down-sampling is to reduce the amount of points in point clouds. The simplest way is to randomly remove some points. However, since the amount of point clouds which are close to the depth sensors are usually greater than further areas, this might lead to removing the already limited point clouds that are further from the camera. For the purpose of this thesis, down-sampling was not required and performance speed remains near interactive rates.

2.1.1 Holes and Edge distortions

Regardless of the technique with which depth maps are generated (Time of Flight or Structured Light), holes are a universal artifact [34]. They refer to the missing depth values. As seen in Figure 2.1, missing depth values are shown as black areas.

Many aspects contribute to the generation of these artifacts. [36] lists "Imaging Geometry" and "lighting conditions" as two major factors affecting the depth quality. In intense light situations, pattern of the infra-red becomes invisible to the IR camera, thus resulting in holes in the image. Regarding the geometry, objects further from the camera are more prone to depth inconsistency. Another situation which could be categorized as imaging geometry and is known as "occlusions"is where the IR camera casts a pattern on an object which is occluding another object. In this case, the IR sensor could see the occluded object due to the distance at which these two

Figure 2.1: Missing information in depth maps known as holes is shown as black areas.

cameras are from each other, but since no pattern has reached it, it is shown as a black area. [34] refers to this issue as"matching ambiguity". Moreover, in areas with many objects close to each other, the chances of a shadowing effect increase. Surface properties, for instance of shiny objects, also affect the depth estimation accuracy.

Extensive research has been conducted to resolve the holes artifacts. [34] categorized two main methods to overcome this issue :"Filtering" and "classification".

1. **Filtering methods**: The most naive approach to use is Median filter. It assumes that missing pixels are probably having the same depth value as their neighbour;therefore, the resulting missing value will be the median of its neighbourhood. However, it is only efficient if noise occupies less than one half of the neighbor area. Another disadvantage is that it smooths corners and edges. The ***Bilateral Filter (BF)*** and its extensions [48] are a non-linear Gaussian based group of filters that are not only used as denoising step, but which preserve cor-

ners and edges. Each pixel is the result of weighted average on the neighbouring pixels by considering the differences of neighbours so that edges are maintained. In other words, pixels affect each other that are not only nearly located, but also are similar in value. [62, 20]. An important extension of the Bilateral Filter is the ***Joint Bilateral Filter (JBF)*** which uses additional information such as the color image to fill depth holes. Essmaeel et al.[22] produced a comparative analysis between several depth denoising filters. They studied the following main filters: *the Median Filter*, *Bilateral Filter*, *Joint Bilateral Filter*, *Non-local Means Filter*(a weighted average of neighbours but with a bigger size window) and *the Adaptive Threshold Filter*(which is a temporal filter that changes the depth value only if its difference with the last filtered data is larger than a threshold) and *Temporal Denoising Filter* [22](where the temporal changes in depth maps are used.)

2. **Segmentation/Classification methods**: In classification methods, objects are classified into different clusters and therefore, missing depth values could be interpolated from the pixels that belong to the same object. Applying segmentation methods, such as in [43], rely on the color information to interpolate the missing depth values. [71] uses a "texture combined inpainting algorithm" which uses RGB images to fill the missing depth values. Their method is based on segmenting the objects to the foreground and background and further using a region growing technique to match the texture with the relative hole region in the depth field.

2.2 Calibration

In this section, we introduce calibration and state-of-the-art techniques in finding calibration parameters. Calibration is performed in either single or multiple camera scenarios. The purpose of the calibration is to determine the projective transformation of a 3D point in camera coordinates and then the projective transformation of a point in camera coordinates to the image plane coordinates with the latter known as Extrinsic Calibration and the former known as Intrinsic Calibration.

2.2.1 Single Camera Calibration

The Microsoft Kinect camera consists of two cameras internally: an RGB camera and an IR camera. Depending on the application of the Kinect camera and whether both of the RGB and IR sensors are used, it is necessary to find the intrinsic and extrinsic parameters of each sensor.

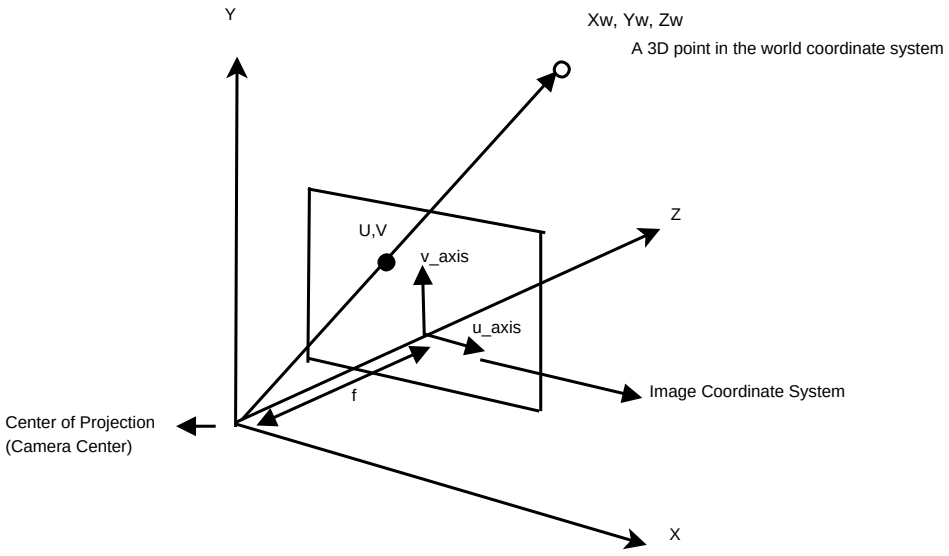


Figure 2.2: Real world to image plane projection of a 3D point

2.2.1.1 Intrinsic Calibration

Intrinsic calibration defines the projective transformation that maps points from 3D (camera coordinates) to 2D (image plane coordinates). A common camera model consists of intrinsic parameters which are the focal length (f_x,f_y), the principal point (c_x,c_y) and radial and tangential distortions of the lenses (k_1, k_2, p_1,p_2,k_3). The calibration process is based on the pinhole camera model. According to Figure 2.2, the point where all the rays intersect is denoted as the center of projection. The center of projection is the center of our camera model. The camera coordinate system is a 3D based system that describes the points in the real world, denoted as X_w,Y_w,Z_w, in the camera coordinate system. The camera frame (X,Y) is parallel with the image plane. The line that passes through the image plane is the optical axis and is equal to the Z axis. The intersection of the image plane with the Z axis is denoted as principal point. The focal length(f) is the distance from the center of projection to the image plane. Using equation 2.1, the projection of the 3D world point in the image plane, U and V, can be obtained. Considering Figure 2.3, using the triangles in the perspective imaging, U and V can be obtained as follows:

$$\frac{V}{Y} = \frac{f}{Z} \quad \text{and} \frac{U}{X} = \frac{f}{Z} \tag{2.1}$$

Where f denotes the focal length. If the center of the 2D image plane does not coincide with the intersection of the Z axis and image plane, the point on the image plane(U,V) will be shifted by c_x and c_y. The center of the image plane is known as the principal point. Thus, we can define k as the intrinsic parameters of the camera

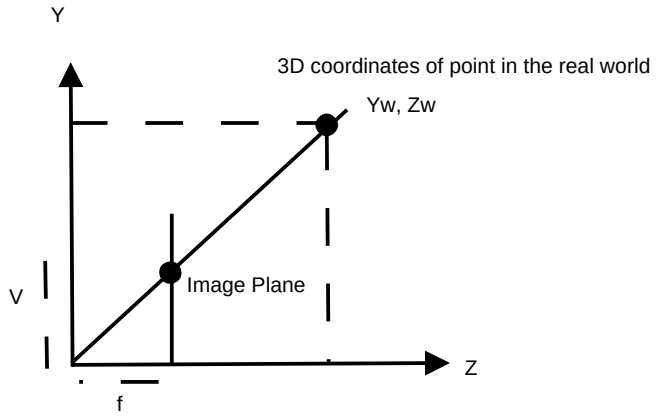


Figure 2.3: Real world coordinate to image plane coordinate projection

and can be denoted by 2.2.

$$k = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Using matrix notation, the observed point in image plane coordinate will be the following:

$$\begin{bmatrix} U \\ V \\ 1 \end{bmatrix} = k \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \tag{2.3}$$

Using Equation 2.3, it is possible to see that point (U,V)=($\frac{f_x X}{Z}$+c$_x$,$\frac{f_y Y}{Z}$+c$_y$). Intrinsic parameters do not change with camera movement; however, they change in situations such as optical zooming or changes in the lenses.

2.2.1.2 Extrinsic Calibration:

In a single camera, extrinsic calibration defines how to convert known world coordinates to camera coordinates. In other words, if the center of the camera model is not at the center of projection (0,0,0) and the axis of both coordinate systems are not aligned, a translation(T) and rotation(R) matrix is required to make the center of camera coordinate coincide with the center of projection, as shown in Figure 2.2. This can be expressed using the following equation:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + T \tag{2.4}$$

This translation and rotation can be denoted as $[R|T]$ which is known as the transformation matrix. Combining Equations 2.2, 2.3 and 2.6, the 3D world coordinate will be transferred to camera coordinates and then to image plane coordinates:

$$\begin{bmatrix} U \\ V \\ 1 \end{bmatrix} = k[R|T] \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \tag{2.5}$$

Substituting k with 2.2, it can be written as 2.6.

$$\begin{bmatrix} \mathrm{U} \\ \mathrm{V} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.6}$$

Equation 2.6 shows that by applying the intrinsic matrix and the transformation

matrix on any point in the real world, its 2D position in the image plane can be obtained.

The Kinect camera needs another type of extrinsic calibration. Since the depth sensor and the RGB camera are separated by a small baseline, RGB and depth map images are not completely aligned. With extrinsic calibration, the location and orientation of the RGB depth sensor with regards to the depth sensor is calculated. When calibrating a single Kinect camera, these values are usually obtained: The RGB sensor calibration matrix, the depth sensor calibration matrix, and the rigid-body transformation between the RGB and the depth sensor [59].

Extensive amount of research exists regarding regular single camera calibration methods, which predates depth-sensing cameras [50]. However, less work has been conducted on the calibration of the depth sensors. Staranowicz et.al [59] studied and evaluated three main calibration techniques: Zhang's method [68], Herrera's method [29] and DCCT method [59]. For the full description of the mentioned methods, the reader is encouraged to review reference [59].

In [70], calibration methods are classified in two general ways: "*photogrammetric calibration*" and "*self-calibration*". In photogrammetric calibration, an object for which its 3D geometry in the real world is known, is used as reference. In the second method, no reference object is required and by using methods such as moving the camera in a scene, internal and external parameters are recovered by finding the correspondence between three images [50].

The calibration method we used in this thesis to evaluate our results is introduced by Burrus, as the freely available software RGBDemo [13]. The intrinsic and extrin-

sic parameters could be found by taking several snapshots of a chessboard pattern based on the OpenCV calibration. OpenCV calibration itself is based on the work of Zhang's [70] method, which is implemented both as a toolbox in Matlab [11] and in the OpenCV library [12]. In this method, a planar pattern, usually a chessboard, is observed from at least two different orientations. Then, feature points are detected in the images and the intrinsic and extrinsic camera parameters are obtained. For calibrating the depth and RGB cameras together, inside a Kinect for instance, [68] proposes the same chessboard calibration target. A maximum likelihood method for depth and RGB camera calibration is provided. After taking several snapshots of the object, the 3D coordinates of the feature points are extracted from the RGB camera. Depth values are obtained from the depth sensor's coordinate system. Feature matching between the RGB and depth image results in features getting their true depth values based on the RGB camera coordinate system.

Through extrinsic calibration, RGB and Depth images have a one to one correspondence; that is, the index of X and Y in RGB refer to the same coordinates in the depth image. Kinects are initially calibrated in the factory and their intrinsic and extrinsic parameters are saved in the device's internal memory. These parameters are suitable for robotics, however, in areas such as mapping and visualization in which accuracy is of high importance, a more accurate way of finding calibration parameters in often required.

2.2.2 Multiple Camera Calibration

As mentioned earlier, having a multi-camera system helps capture a larger portion of the environment. Furthermore, a comprehensive 3D model of the scene could be obtained, since areas occluded by one camera can be captured by the others. Izadi et al. presented KinectFusion [46], in which only one camera is held around a scene and a 3D model is obtained by moving the camera around the object or scene being captured in small movements. However, the 3D model obtained is an offline map. Another motivation in having multiple cameras is for online applications such as human computer interaction, gaming, virtual reality and gait analysis.

The result of extrinsic calibration stage is a rotation and translation matrix that shows the relative position and orientation of one camera with regards to the other. The goal is to bring the viewpoints of all cameras in to one coordinate framework. Figure 2.4(left) demonstrates the viewpoints of two uncalibrated cameras and Figure 2.4(right) shows the calibrated cameras. Notice that overlapping sections are perfectly aligned.

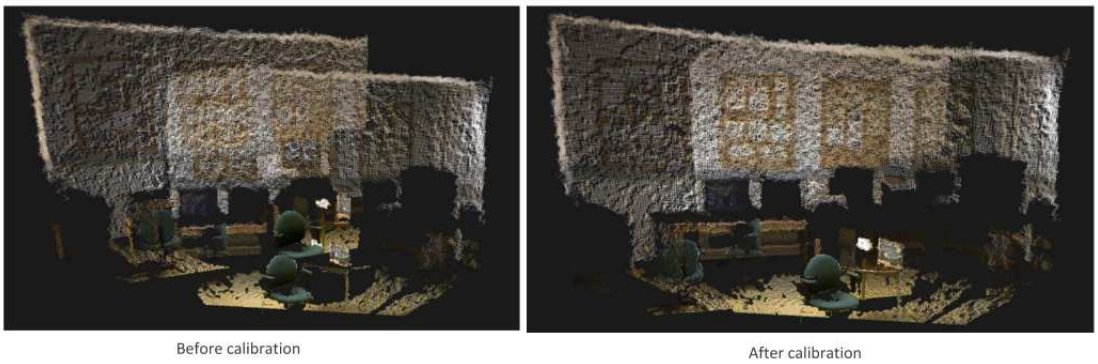


Figure 2.4: Two camera viewpoints before and after calibration

It is important to notice that for a single camera calibration, the extrinsic and intrinsic parameters would not change; because the relative position of the cameras is usually fixed for the built-in components of a device; however, in multiple cameras, once cameras are registered, they have to remain static, since the parameters are only valid for the current position of the cameras. Any small movement in either of the cameras will result in misalignment of the point clouds and hence would require the calibration process to be performed again. Calibration of multiple sensors could be divided in to two main categories:

1. **Standard Calibration:** This technique requires the use of a calibration pattern, normally a chessboard, texture planes [19], spheres [56] or even tracking LED or users motion [15, 61, 44], that is visible by all the cameras. The goal is to find the correspondences between multiple viewpoints and obtain a transformation matrix using these correspondences. The same technique is used in calibrating single Kinect cameras to align the internal Depth and RGB cameras. According to [21], these methods are suitable when the calibration is performed on an RGB camera, assuming that either single Kinects are already calibrated or the factory calibrated parameters of the Kinects are used. Others, such as [9], use a calibration target with holes or depth variations so that they could be visible to depth sensors as well. Figure 2.5 shows a sample of the calibration procedure that requires several snapshots of the target, which is a chessboard in this case.

Figure 2.5: Several snapshots of the chessboard target used in the standard calibration method

2. **Automatic Calibration:** Also known as target-free or self-calibration methods. These methods do not depend on users or patterns. Cameras can be calibrated only based on the environment and images they are viewing [58]. For example in [18], instead of an object, they take orthogonal walls inside the room to calibrate cameras and therefore, require no user interventions. The algorithm only requires an image with three pairs of points which are located in different walls being orthogonal to each other. In addition, [30] uses geometric features such as vanishing points to determine both intrinsic and extrinsic parameters of an array of surveillance cameras by examining the pedestrians.

2.3 Point Cloud Registration

Point cloud registration refers to aligning point clouds captured either using one depth-sensing camera (that is capturing a scene from different viewpoints) or using multiple depth-sensing cameras. Each of these point clouds have their own coordinate system. In order to bring these point clouds to a single framework, so that overlapping parts of point clouds correctly intersect, registration methods are used. Current registration techniques, such as ICP, are mostly successful when they are used for fine alignment and are usually not suitable for point clouds sharing small overlapping sections. In those scenarios, cameras are first calibrated and then ICP is performed.

2.3.1 Iterative Closest Point (ICP)

The ICP (Iterative Closest Point)[10] is one of the major methods used in aligning 3D data when registering the 3D scans from a model taken at different times. It uses an initial guess of a rigid body transformation between two point clouds and then iteratively refines the transformation with the goal of finding the best transformation to minimize a distance metric.

Assuming there are two point clouds where one of them is regarded as the source and the other as the target. The goal is to find the relative transformation (including rotation and translation) to transform one to best match the other. For every point in the source point cloud, the corresponding point in the target is chosen based on the smallest Euclidean distance, which is called the matching stage. Then, in a rejection stage, all points that have a distance higher than threshold will be removed from the sets. After finding the corresponding points, a transformation is obtained with the

goal to minimize the error metric, which is the sum of squared distances[10]. The procedure of finding the correspondence and applying a registration based on that is iteratively performed until a certain criteria is met. These criteria can be used continuously until reaching the desired number of iterations or finding a transformation that no longer affects the distance of point clouds from each other.

The necessity of an initial pose arises from the issue that if point clouds are not located closely in terms of distance, a wrong point correspondence can affect the transformation considerably. ICP is successful in scenarios where two point clouds have large overlapping areas. Otherwise, a false alignment of point clouds will be performed.

Since the introduction of ICP by Besl and McKay[10], many variants of it have been introduced [52] which are classified based on different stages of the ICP algorithm: point selection, point matching, weighing the corresponding pairs, rejecting pairs, an error metric description, and minimizing the error metric. For instance, [57] proposed using a k-d tree and/or closest point matching for finding the correspondence in the matching stage. [64] proposed rejecting pairs on the mesh boundaries in the rejection stage.

2.3.2 DeReEs

DeReEs [55] is an algorithm for real time registration of RGBD image pairs using image based feature detection and robust 3D correspondence estimation and refinement. It consists of three main stages: Feature detection, pair rejection and fine transformation estimation. The algorithm takes two RGBD images taken from either

two depth sensing cameras or from one camera but from different viewpoints. The two captured scenes should have an overlap of at least 25%. The output is a $4x4$ matrix that shows the transformation and rotation of one viewpoint with respect to the other. DeReEs performs in $36ms$ on average.

2.3.2.1 Feature Detection

Image-based feature matching methods, such as SURF [6] and ORB [51], are used in this stage to find the corresponding pairs between two RGB image pairs. The output of this stage is two sets of feature pairs where each pair contains the 2D positions of features in the RGB images. Considering RGB+depth images, the RGB and depth images are aligned, therefore, the 2D locations of the features correspond to the exact 2D locations in the depth image, giving access to the third value which is the depth(Z) information. Using point clouds, however, one can easily access the 3D positions by knowing the 2D position of the features.

2.3.2.2 Pair Rejection

Some of the feature pairs found in the previous step could be false due to the fact that matching algorithms are prone to matching two unrelated features. The aim of this step is to find three true feature pairs using RANSAC [23] which are used to obtain a coarse transformation. For a number of iterations, three random feature pairs in the source cloud and their corresponding pairs in the target are used to find a transformation. This transformation is applied to all feature pairs. Assuming a transformation is successful, it is expected that true features will be close to each other, whereas wrong features will be placed relatively further from each other. The

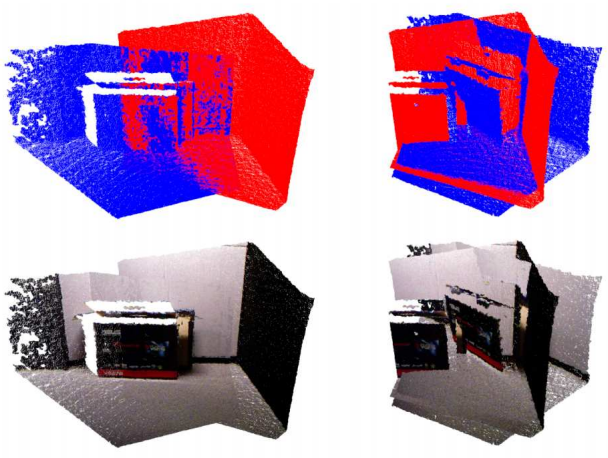
metric used to find best transformation is the number of features for which their distance to their corresponding transformed features is less than a distance threshold. This process of selecting three random pairs will be iterated for a certain amount. Within each iteration, the number of features that are placed in a distance of smaller than certain threshold will be counted. Based on the proposed metric, the best transformation found is the one that has highest score, in this case, the highest number of matching features. After finding the best transformation, it is then applied to all feature pairs and those with a distance more than threshold are removed from the feature sets of both point clouds. This removes any false feature pair correspondence and the remaining feature pairs will be considered true matches.

2.3.2.3 Fine Transformation Estimation

In this step, all remaining feature pairs are used to estimate a fine transformation that transforms the features of one cloud to the other, minimizing the least square error between corresponding feature pairs. The transformation is performed using singular value decomposition (SVD) [5] implemented in PCL. Since true feature pairs are used in this stage, the transformation is more reliable and ensures that false features will not affect its accuracy.

2.3.2.4 Evaluation of DeReEs

DeReEs has been shown to be successful with viewpoints that are sharing a minimum amount of 25% overlap (Figure 2.7). Figure 2.6 shows how DeReEs outperforms ICP being performed on the same pair of point clouds. For a thorough review on the method's robustness, the reader is directed to [55].



(a) DeReEs (b) ICP

Figure 2.6: Comparison between the registration results of ICP and DeReEs being performed on the same pair of point clouds(This figure is reprinted from [55] with permission from the owner).

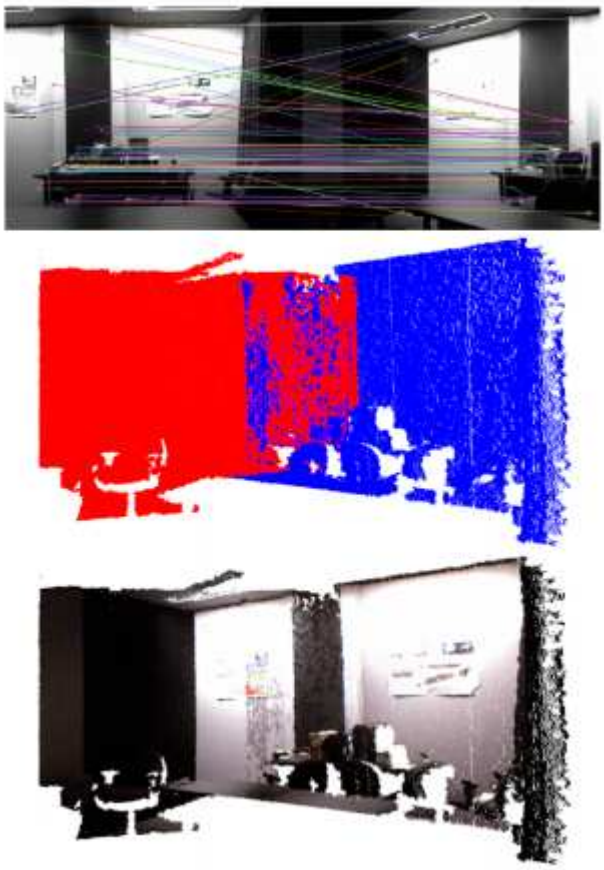


Figure 2.7: A successful registration of a 25% of overlap. The top image shows the feature matching section of the algorithm(This figure is reprinted from [55] with permission from the owner).

2.4 Related Work on Multiple Depth-Sensing Cameras

The use of multiple depth-sensing cameras has been recently investigated in research and industry [46, 72, 39]. As mentioned earlier, with multiple cameras, a larger

field of view and a more accurate and complete 3D model is obtained. Since we are interested in the hassle-free setup of multiple cameras and providing a system easy for non-expert users, we will investigate the state-of-the-art work based on three categories: Unsupervised calibration, allowing camera movement, and the execution speed. A recent survey on the applications of Kinect sensors lists 9 different areas of applications and over 80 different documented cases [39], some of which we will describe here in brief. It is important to note that many papers have been published regarding the following categories using single RGBD cameras, however, the goal is to provide a survey of the same topics when using multiple RGBD setups.

2.4.1 3D mapping and reconstruction

Multiple depth sensing cameras can be used to build 3D models of the environment or even people mostly in indoor environments. For instance, one of the applications is in Simultaneous Localization and Mapping(SLAM) where a 3D map is constantly being updated while the location of a scanning agent is constantly tracked [63]. Rafibakhsh et al. [49] introduces a setup of two Kinect cameras that are calibrated using a chessboard pattern. They evaluated their work by measuring accuracy, resolution and interference of Kinect. They state that two Kinect cameras should not face each other and that a minimum angle of at least 35 degrees should be maintained between two cameras. In [47], three calibrated Kinect cameras are used to construct texlets. Textlets are surface patches that consist of a center point and surface normal. In their work, a real time algorithm on GPU is presented for the texlets' extraction. They also investigated how adding Kinect cameras affects the texlets' extraction ac-

curacy. Macknojia et al. [40] calibrated five Kinect sensors to build a textured 3D map that covers a 180 degree field of view of a vehicle. The calibration method uses a chessboard calibration target being fixed at one place between the overlapping areas of Kinect cameras. In [1], six Kinect cameras are used to provide a 360 degree view reconstruction of a dynamic scene. Their configuration consists of simultaneous capture and reconstruction that each Kinect is connected to a separate PC. Interference between cameras is neglected here since the goal was to acquire a 360 degree 3D model. The cameras are calibrated using chessboard pattern and the speed is not mentioned. In their work, the main concern is to synchronize each machine to a web server, so that the internal clocks of each independent machine are synchronized. In [45], four kinects are connected to different computers and their clocks are synchronized by a NTP server. The focus is more on the fact that depth data is captured asynchronously, however, it is later synchronized applying a temporal calibration.

2.4.2 Human Recognition/Tracking and Activity Analysis

Caon et al. [14] propose a real time system for tracking users from two Kinect cameras. Cameras are carefully calibrated using the coordination of the users' skeleton joints that are visible to both cameras. They also include gesture recognition for home applications. For gesture recognition purposes, the coordinates of the objects to which the user gestures towards are initially saved in the 3D model of the scene and therefore, moving any of the cameras results in false gesture recognition since the locations of the objects in the 3D model have changed. In [60], four cameras are used to generate point clouds from the calibrated depth information. The cameras are first calibrated

by manually rotating the point clouds and the point clouds are then refined using ICP. They provide improvement in object classification and recognition and give successful measures in object recognition using both depth and RGB information rather than only RGB. In [66], LightSpace, a room with calibrated Prime-sense depth cameras and projectors is introduced which provides interactions between people, space and objects such as surfaces and walls. The room is highly interactive where any surface such as walls, tables, human bodies and even the air between these surfaces can be used as display areas. Virtual objects can be displayed in any of the surfaces, allowing the user to pick up the virtual object by hands and transfer it to any other display area. Their maximum number of users is reported to be six for functional interactions and reported frame rates of 30Hz. The overall latency of the system is reported to be 100ms in user interactions. Authors of [26] describe a system using three Kinect cameras to map performers' movements while dancing to music parameters to synthesize music. Multiple cameras are used to improve tracking of the user's joints, in case of occlusion. They do not process the whole point clouds, but only try to merge the skeleton data. In order to register the skeletons, a user moves slowly between the overlapping sections. The process takes about one minute to gather enough data and an exact time for registration is not provided, instead, it is only mentioned the registration takes"few seconds". In [31], three Kinect sensors were mounted on an equilateral triangular rig. The application is in facial recognition. An automatic calibration procedure is presented which is based on Kinects sending messages to each other to rotate in order to find the distance between each other. To increase facial detection quality using depth maps, geometric features are applied.

In the health care sector, active and immersive games are being used together

with RGBD sensors to help rehabilitate patients with motor disabilities and for other treatments as well [37, 65, 4]. As part of a medical exam, multiple calibrated cameras are being used to record the gait patterns of patients walking within large rooms [27, 17, 16].

2.4.3 Telepresence and Teleconferencing

Telepresence and teleconferencing is the ability that allows two remote environments to connect through a virtual window in a way that the virtual environment seems like an extension to the other room. Extensive research has been conducted on remote 3D collaboration and telepresence systems. Zhang et al. [69] mentioned one of the first telepresence systems using a depth sensing camera: a Kinect based 3D video conferencing system that consists of two laptops, one Kinect at each end sending and receiving audio and video. The frame rate reported is 15 f/s, however, only one Kinect is used at each end. Later, [67] proposed an immersive teleconferencing system that assumes one user at each end during the conference. Users are rendered in a virtual room as if they are seated around a table. Eye tracking methods are used so that the virtual room is rendered according to the users' gaze. Each site is equipped with 3 IR cameras, 3 color cameras and two IR laser projectors, carefully calibrated. Maimone et al. [41], proposed a telepresence system by merging overlapping views from multiple calibrated Kinect cameras. A fully dynamic 3D scene capture is presented that preserves gaze, performing at 30Hz. Later in [42], this work is improved by using bigger and higher resolution 3D displays to convey the feeling of a window through another room, and also improvements made on camera calibration, rendering and

sensor data processing. In [7], an immersive telepresence system is presented that allows meeting of a group of people in a shared virtual 3D world. In their setup, two projection-based multi-user display systems, two arrays of Kinects, and a distributed virtual reality framework named AVANGO is used. Alexiadis et al. [3] proposed to reconstruct 3D models of moving humans in a setup of 4 kinect cameras connected to a single PC. Their results are provided with a case of mixing 3D humans in a virtual reality space. The reported speed is close to $10Hz$ and cameras are fully calibrated using an external calibration target.

In all multi-camera scenarios above, the cameras must be carefully calibrated and need to be fixed and placed in a certain position during the whole session, while the work presented in this thesis allows multiple camera setup with the flexibility to allow cameras to be moved during operation.

2.5 Related Work On Non-Calibrated Multiple RGBD Cameras

While many research contributions exist regarding using multiple depth sensing cameras, almost all of them require careful calibration between the cameras. In this section, we give a brief overview of the systems that do not require calibration which are similar to our method.

2.5.1 Unsupervised extrinsic calibration of depth sensors in dynamic scenes

Miller et al. [44] proposed a camera registration solution without human supervision or fully textured environments. With a similar purpose to ours, they try to provide a multiple camera setup that will not require human effort or a specific target. Their goal is to calibrate two cameras placed in a cross-eyed view setup (where the cameras do not share a background), based on the analysis of the motion of a user or an object moving within the field of view of both cameras. Instead of using feature matching, they interpret the optical flow across multiple frames and find their correspondence to obtain the extrinsic calibration parameters. Foreground objects (their centroids) are used to obtain an initial transformation using RANSAC. This step gives a rough transformation which is then refined by an occlusion-aware energy minimization step.

To evaluate their results, they compared their method to the standard chessboard calibration and reported an estimated translation error of 5 cm. and a rotation error of 1 degree. Although they are able to handle situations that cameras do not share any overlapping sections but the motion of the target, it has an average execution time of about 10 minutes, which makes the solution unsuitable for live operation or online adjustments of camera arrangements. Later in this thesis, part of our evaluations will include a comparison between our method and the work of Miller et al.

2.5.2 Human Performance Capture Using Multiple Hand-held Kinects

Liu et al. [38] presented a markerless full human motion tracking and skeleton capture system using three hand-held Kinect cameras where the cameras can be moved. Capturing motions and skeleton usually requires static calibrated cameras or the use of markers on human bodies to track motions and skeletons. In this work, segmentation methods are used to classify the captured data into *human*, *background* and *ground plane*. The ground plane is segmented and modelled for more robust human and camera tracking results, however, it has limited radius of $3m$, thus limiting the performers' space. In their algorithm, they simultaneously track the motion of the Kinect cameras to align the RGBD data as well as aligning the tracked skeleton on the RGBD frames. For a more robust result, the background is also used to provide alignment measures through SIFT feature matching.

To align data frame by frame, the cameras' extrinsic parameters of the previous time steps are used. This requires an initialization of camera extrinsic for the first captured frames. For the human tracking part, the human template model is defined using scanned models of humans captured with a laser scanner. For the new frames being captured after the first time step, a new algorithm is defined that finds the correspondence of point clouds to the fitted human models. Simultaneously, correspondences are found through the ground plane and also the background.

Their work does not suffer from interference of multiple Kinect cameras as the viewing angle between them is 120°. Moreover, missing depth information will not produce 3D point clouds and therefore, it will not affect the tracking accuracy. Failure,

however, occurs when too many occlusions or fast motions take place while capturing.

Although the results are fast and suitable for real-time human tracking systems using moving Kinect cameras, their work requires a step of initialization in the first frame and requires persons to move within the scene (although the motivation is to track humans, the human tracking results are also used in improving the registration in time), and finding correspondence in both background and ground plane for more robust results. A comparison could not be made between our algorithm and the method of Lui et. al since we did not have access to their data and the results published in their paper evaluates the system based on the accuracy of tracked humans rather than the accuracy of the registration.

2.6 Conclusion

In this chapter, we presented the background topics related to this research. We first presented calibration, which is necessary in every multiple camera system. Then, we introduced registration techniques, which are an important processing step in aligning multiple viewpoints of a scene. Finally, we provided an overview of the state-of-the-art research on having multiple depth-sensing cameras, which is split in two general categories of calibrated and non-calibrated cameras. In the next chapter, we will present the design of a system to support live RGBD registration of video streams.

Chapter 3

Design of a system to support live RGBD registration of video streams

In this chapter, we will present the design of a system to support live RGBD registration of video streams and introduce the modifications on DeReEs [55]. We observed that DeReEs by itself was not appropriate for continuous video frame registration. First, applying DeReEs to every two point clouds being captured by cameras produces delay. The reason is that DeReEs performs in $36ms$, however, this measure is only for finding the registration matrix and excludes the time required for capturing and visualization. Secondly, updating the registration information for every two frames produces an unstable alignment and results in a flicker effect. Finally, since we are motivated to have a system robust to camera displacements, camera movement detection was required. In this section, we introduce DeReEs-4V which stands for

DeReEs for Video and satisfies all the requirements mentioned above.

3.1 Pipeline

In our setup, two Kinect cameras are connected to a computer, the server. The cameras can be arbitrary located anywhere inside an indoor space and must share a small overlapping section of at least 25%. The sender side captures point cloud video frames from two Kinects, performs DeReEs-4V on it and streams the registered viewpoint to the other computer, the client. In order to reduce delay, every step of the algorithm is implemented on a separate thread. Figure 3.1 shows the system's pipeline.

Capturing RGB+Point Cloud data from Kinect is performed using the Point Cloud Library (PCL) frame-grabbing interface. Instead of capturing point clouds, one can use RGB frames and their respective depth images. This information is sufficient for finding a transformation, however, for visualizing the data, point clouds should be produced continuously. The captured frames are stored in separate buffers at $30fps$ where one buffer stores point cloud data(PCD) and one stores its respective RGB image. These are the inputs for different steps of the algorithm. Using buffers reduces delay and allows threads to separately function once data is available in them. The thread responsible for registering the point clouds is the implementation of DeReEs with modifications we proposed which make it suitable for video registration.

The first stage of the registration is to find corresponding features in both point clouds. In our system, data is streaming constantly. The input of the registration

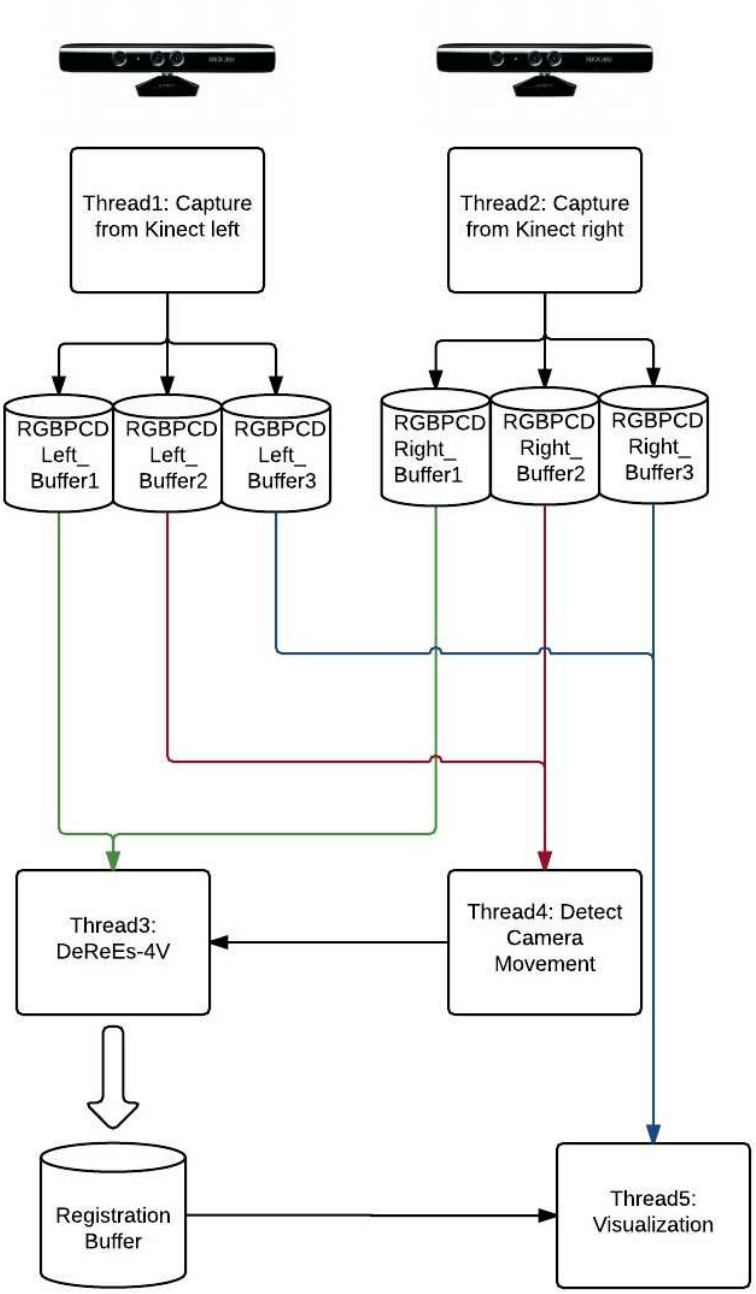


Figure 3.1: General work flow of the system designed with different threads to reduce delay.

algorithm, which is being executed on a separate thread, is a pair of RGB images plus the corresponding PCDs. PCD is the point cloud data structure which is obtained at each frame using PCL library. The point cloud contains X,Y,Z and RGBA information of the scene the cameras capture. RGB images are obtained from each of the camera's RGB buffer and are constantly being uploaded on the GPU to apply SURF feature matching on them.

The first step of DeReEs is finding corresponding features. The RGB images are used to detect features using SURF feature matching algorithm. Once corresponding features are found, they are used to find the coarse transformation estimation using RANSAC. The procedure is that, having 2D locations of features in the RGB image, their 3D information can be accessed in the point clouds using the same 2D index. Finding the coarse transformation is an iterative process. Three random feature pairs and their correspondence in the second point clouds are used to find a transformation. The goal is to align the target point cloud to the reference one. The transformation is applied on all feature pairs of the target and features that are located under a distance threshold will be counted. This stage is repeated for a fixed number of times (in our algorithm, 100 times) and the transformation with the highest number of threshold is considered as the coarse transformation. This transformation is then applied on all feature pairs of the target and features that are further than distance threshold are removed from both image feature sets, resulting in a set of "true" corresponding feature sets. Finally, using the true feature pair set, a transformation is obtained and is the final output of DeReEs. A detailed description of DeReEs is found in Seifi et al. [55, 54].

One modification to DeReEs which is introduced in DeReEs-4V is applying ICP

on the subset of feature pairs remaining after the feature rejection phase, known as true feature pairs. This added step of ICP gives a more stable alignment over several iterations of DeReEs-4V. ICP is known as a computationally complex registration method, however, this will not introduce significant delay on DeReEs-4V since it is performed on a very small subset of 3D feature pairs, with execution times of less than $1ms$. Since the point clouds are already transformed with a coarse transformation which is close to the final transformation, it will tend to fall within the local minimum that corresponds to the best possible transformation and will not produce wrong alignments, which is one of the disadvantages of ICP when point clouds are not close enough to the actual solution.

With the transformation information, one point cloud, the target, will be translated and rotated to the coordinates of the other point cloud, the reference. The whole procedure of finding a transformation registration, applying it on the point clouds and visualizing them executes in $65ms$ or 15 fps on average. At the same rate of $65ms$, point clouds are registered continuously from the start. The registration information produces a well aligned background, though the foreground objects require more time to be finely tuned. Therefore, for updating each of the two coming point clouds, we are using the best transformation, that is the one with the highest number of matching features, found over several iterations of DeReEs-4V. In this way, the transformation keeps improving over time while the cameras are stable. Once a registered point cloud is obtained, it is sent over the network to the receiver using Octree Compression method implemented in Point Cloud Library(PCL)[2]. At the receiver side, the point cloud is decompressed and is visualized. This is outlined in Algorithm 1.

Input : $RGBPCD_l$: RGB+PCD frames from left Kinect.

$RGBPCD_r$: RGB+PCD frames from right Kinect.

Definitions: $RGBPCD_{lcurr}$: The current RGB+PCD from left Kinect.

$RGBPCD_{lprev}$: The previous RGB+PCD from left Kinect.

$RGBPCD_{rcurr}$: The current RGB+PCD from right Kinect.

$RGBPCD_{rprev}$: The previous RGB+PCD from right Kinect.

T: Transformation information produced by DeReEs.

while *cameras are capturing* **do**

$Cam1IsMoved \leftarrow DetectCamMove(RGBPCD_{lprev}, RGBPCD_{lcurr})$

$Cam2IsMoved \leftarrow DetectCamMove(RGBPCD_{rprev}, RGBPCD_{rcurr})$

$T, countFeatures \leftarrow DeReEs\text{-}4V(RGBPCD_l, RGBPCD_r)$

if $Cam1IsMoved$ *or* $Cam2IsMoved$ **then**

$score \leftarrow countFeatures$

$memory \leftarrow T$

else

if $countFeatures > score$ **then**

$memory \leftarrow T$

$score \leftarrow countFeatures$

else

$T \leftarrow memory$

end

end

end

Algorithm 1: Registration transformation produced by DeReEs-4V.

As we discussed earlier in Chapter 2, moving a camera will result in misalignment of the point clouds. In order for the system to be perceptive enough regarding the camera movement, a camera movement detection module is proposed which will be discussed in detail in Section 3.1.2. This module allows the system to maintain registration when any of the cameras move.

3.1.1 Scenario one: Cameras are stable

An important observation while capturing video streams and registering them is that the registration transformation obtained for each two video frames is rarely exactly the same as the previous one. This is visually more evident for the users observing a movement in the point clouds due to the fact that the point clouds are updating themselves with different values in the registration transformation. In this research, we propose a registration system that strives to produce registration transformation that are stable over time unless any of the cameras are moved or a better transformation is found, which results in updating the registration information (i.e., the transformation matrix). In our experiments, we noticed that the algorithm takes about 15 seconds or less to find a stable registration information that can be used until the cameras are moved.

To implement this, we devised a memory mechanism to store the best transformation obtained from the registration procedure. As discussed in Chapter 2, DeReEs randomly selects 3 feature pairs from both point clouds. Using these features, a transformation is obtained that is performed on the target point cloud. Corresponding features which are located in a shorter distance than a certain threshold ($5cm$) are

counted. This procedure is repeated for a certain number of iterations and the transformation with higher number of features is selected as the best transformation. The number of features is the metric used to update the memory. Each time a registration is found, the metric is compared to the one saved in memory and if the new score is higher than the score in the memory, then the new transformation information and the new score(the number of features) will be updated in the memory. Otherwise, the point clouds keep getting updated with the former transformation registration. It is important to note again that the registration will be performed for each two coming point clouds; however, it will not be visualized and updated in memory unless it is regarded as a better one.

Our evaluations, which are described in Chapter 4, show how the system reaches a more stable transformation in time. However, this transformation is only acceptable as long as cameras are not displaced. The system has to be perceptive enough to detect camera movement, otherwise, the new camera placement might result in a lower score; in which case the memory will not get refreshed and the point clouds will instead be registered with the transformation stored in the memory, as long as it does not find a higher score. This results in producing an incorrect alignment for some time.

3.1.2 Scenario two: Cameras are moved

The memory information is valid until cameras are moved. Once cameras are moved, the memory has to be updated with the registration information corresponding to the new camera arrangement. In order to detect camera movement, we again use

DeReEs-4V running on a separate thread. Each camera has its own thread responsible to detect any displacements. Assume the thread responsible for camera movement detection on the left camera. Each RGB+PCD information from its corresponding buffer at time t_{curr} is given to I_{lcurr}. Getting new RGB+PCD from the buffer will result in transferring I_{lcurr} to I_{lprev}, while updating the I_{lcurr} with the new RGB+PCD frames. These two frames(I_{lcurr} and I_{lprev}) can now be given as input to DeReEs-4V to find a transformation matrix. This transformation is temporarily saved (A), the current frame(I_{lcurr}) is saved as the previous frame while the new frame is obtained. These two frames will now be given to DeReEs-4V, and the resulting registration matrix (B) will be compared to the stored one (A). Equation 3.1 is used to compare these two matrices by subtracting them. If the absolute difference of any indexes in the resulted subtraction matrix is higher than the threshold (0.05), the system is then notified that the cameras have moved and the memory's scoring function and transformation information has to be refreshed using the main DeReEs-4V thread for finding the new registration values. The threshold is selected with trial and error method in a way that the small movements of people in the viewpoints are not accounted as camera movements. The whole procedure is outlined in Algorithm 2.

$$abs((A - B)_{i,j}) > 0.05 \tag{3.1}$$

3.1.2.1 Alternative to DeReEs for detecting Camera movement

In the first attempt to detect camera movement, we proposed using RANSAC to find a homography matrix with the same procedure as described in Section 3.1.2. Although only RGB information of each camera is used to detect camera movement,

results showed a slower camera movement detection of $800ms$ on average. Therefore, we proposed using DeReEs-4V, which is more stable and faster in detecting camera movement detection, and takes only $63ms$ on average.

3.2 Conclusion

In this chapter we presented DeReEs-4V which is an extension and real time implementation to a previous work, DeReEs, which registers images taken by RGBD camera. DeReEs finds the 3D transformation of two viewpoints in about $60ms$ per frame. The input of DeReEs is two RGB images plus the associated PCD files and the output is the 3D transformation of the viewpoints. We introduced DeReEs-4V, which is used for RGBD video streaming. DeReEs-4V takes the stream of RGB+PCD frames coming from a pair of RGBD cameras. DeReEs-4V performs DeReEs for each frame pair and applies the obtained transformation on the point clouds. In DeReEs-4V, a memory mechanism is implemented to register point clouds based on the best transformation found so far, and for every better transformation that is obtained, the memory is refreshed. The memory is also refreshed in case of camera movement detection. Therefore, DeReEs-4V also supports camera movement detection. DeReEs-4V is a system which includies different threads for grabbing frames, applying DeReEs, detecting camera movement and visualizing the registered streams. DeReEs is the one part of DeReEs-4V that is solely responsible for obtaining the 3D transformation that produces the alignment of the video streams coming from both RGBD cameras.

Input: $RGBPCD_l$: live video streams from Kinect on the left.

$RGBPCD_r$: live video streams from Kinect on the right.

while *cameras are capturing* **do**

  if *first capture is taking place* **then**

    $I_{lcurr} \leftarrow BufferRGBPCD_l$

    $I_{rcurr} \leftarrow BufferRGBPCD_r$

  else

    $I_{lprev} \leftarrow I_{lcurr}$

    $I_{lcurr} \leftarrow BufferRGBPCD_l$

    $I_{rprev} \leftarrow I_{rcurr}$

    $I_{rcurr} \leftarrow BufferRGBPCD_r$

  end

  if *first measuring transformation is taking place* **then**

    $H_{lcurr} \leftarrow DeReEs\text{-}4V(I_{lprev}, I_{lcurr})$

    $H_{rcurr} \leftarrow DeReEs\text{-}4V(I_{rprev}, I_{rcurr})$

  else

    $H_{lprevs} \leftarrow H_{lcurr}$

    $H_{lcurr} \leftarrow DeReEs\text{-}4V(I_{lprev}, I_{lcurr})$

    $H_{rprev} \leftarrow H_{rcurr}$

    $H_{rcurr} \leftarrow DeReEs\text{-}4V(I_{rprev}, I_{rcurr})$

  end

  if $||H_{lprev}| - |H_{lcurr}|| > threshold$ *or* $||H_{rprev}| - |H_{rcurr}|| > threshold$ **then**

    $CameraIsMoved \leftarrow true$

  end

end

Algorithm 2: Camera Movement Detection

Chapter 4

Evaluation

In this chapter, we mainly focus on evaluating our algorithm with respect to the related research (such as Miller et al.[44]) and ground truth (standard calibration). First, we will describe our system settings. We will then introduce our data set and different scenarios and experimental hypothesis used to evaluate our system.

4.1 System Configuration and Experimental Environment

In our experiments, two Microsoft^{TM}XboX 360 Kinect cameras are used at one end, the sender, to capture a portion of an indoor office space. The other PC will receive the registered point clouds through the network. The Kinects are connected to a PC equipped with an Intel(R) Core(TM) i7-960 CPU with a NVIDIA GeForce GTX 570 GPU, 1280MB memory, 480 CUDA cores, running Ubuntu 14.04 64-bit. The Kinects are arbitrarily located and only require a small overlap in the part of the

scene they capture. Since image-based feature matching algorithms are used in the registration pipeline, the indoor space needs to have enough illumination to produce a useful result.

We have compared our system to two methods. One is the work of Miller et al. which is described in Chapter 2, Section 2.5. The second one is the standard calibration method proposed by Nicolas Burrus RGBDemo [13]. Miller et al. also evaluated their results based on the same method. In order to compare our method to the ground truth, for each scene captured we first calibrated the Kinects and then compared the calibration parameters with our results being applied on the same camera pose.

The algorithm and system is implemented with the use of Open Source Computer Vision library(OpenCV) and Point Cloud Library (PCL) using C++ programming language. The GPU implementation of SURF feature matching in OpenCV is used. PCL is used for point cloud stream capture from both Kinects simultaneously. It is also used for visualizing the aligned point clouds using PCL Visualizer, and in Octree point cloud compression for streaming the point clouds over the network.

4.2 Experiments

Our experiments are based upon different video capturing scenarios where one, two or three users are captured under varying conditions. Users are moving from overlapping regions captured by the Kinect to non overlapping and past the boundaries of the regions captured by both cameras during the registration, and it is still successful.

Evaluations are categorized in four aspects: varying overlapping conditions, cam-

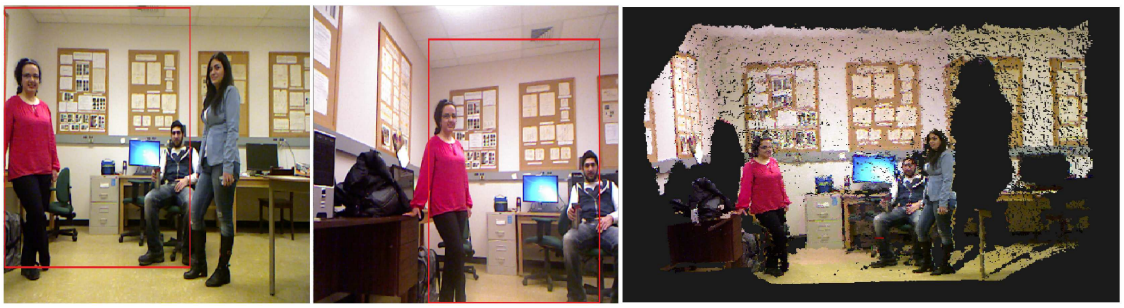


Figure 4.1: An example of successful registration of a multi-person scene(right). Two persons are captured by both Kinects(center), while the third is captured by only one camera(left).

Figure 4.2: An example of successful registration of a multi-person scene where each person is captured by only one Kinect, each.

era movement, registration speed and accuracy.

In order to evaluate the algorithm under different conditions of overlap, we measured the RGB window size on the screen(S), and for a certain amount of overlap, for instance 45%(B), we can measure the amount of B in centimetres using equation 4.1. A and C are the two RGB images and B is the overlap between them. The result is the amount in centimetres that shows how much the two RGB images should overlap.

Figure 4.3: Example result of two persons where half of one person is captured by one camera and the full body is captured by the other. The image on the right shows a successful match of both scenes.

Figure 4.4: An example of successful registration of a multi-person scene(on the right) where one camera(on the center) is located higher than the other one (on the left).

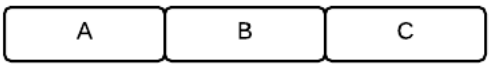


Figure 4.5: Measuring the overlap: A and C are the non overlapping regions of two RGB images and B is the region of overlap between them.

$$\%overlap = \frac{B * S}{100 - (A + C)} \tag{4.1}$$

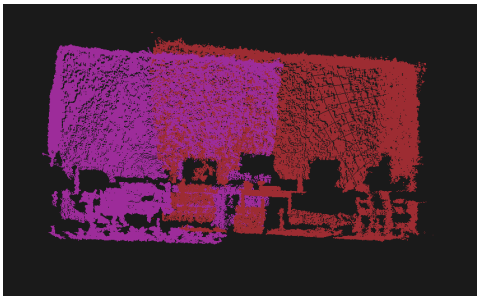
4.2.1 Performance under varying amounts of overlap

In order to evaluate the alignment improvement over time under varying overlapping conditions of 25%, 45%, 65% and 85% where one camera is stable and the other is being moved to vary the degree of overlap(Equation 4.1). The captured space is an office where one user is standing in front of the camera (Figure 4.6).

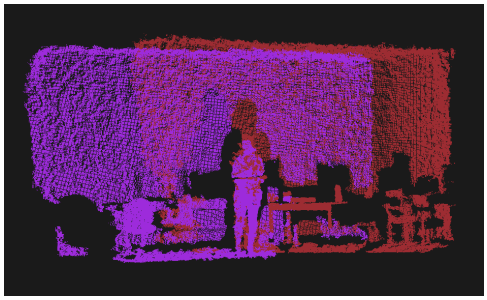
Figure 4.7 shows the improvement of the algorithm over time with increasing amounts of overlap. The number of features in this figure shows the metric described in Section 3.1.1, which is the number of feature pairs used to update the alignment to a better one over time. Each time a higher value is found, the transformation is updated.

As it can be observed from Figure 4.7, plots do not reach the same point on the number of features. This is due to the fact that each scene has a different number of matching features to start with. For example, scenes with high amount of overlap have high number of matching pairs, as well as scenes with many distinctive features. It also shows the number of features for different overlaps. More features are found for larger overlaps (85%) with an overall trend toward higher number of features over time. Note that the graphs contain the data for 10 iterations of algorithm1 per condition, therefore, each data point is the average over 10 iterations.

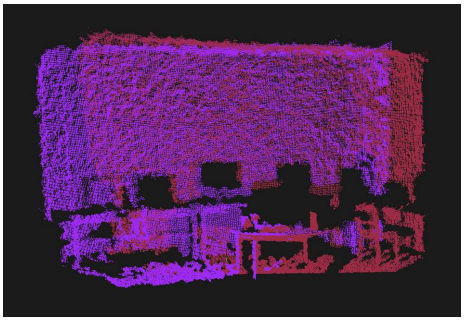
Figure 4.8 demonstrates a fitted model of the data for each overlap condition separately. As can be seen, the figure shows the tendency of the algorithm towards finding an increasing number of features over time in the 25% overlap. Other conditions follow the same rule, however, being more stable the entire time. In terms of rotation error, Figure 4.9 shows the fitted line over data for four overlapping conditions, where



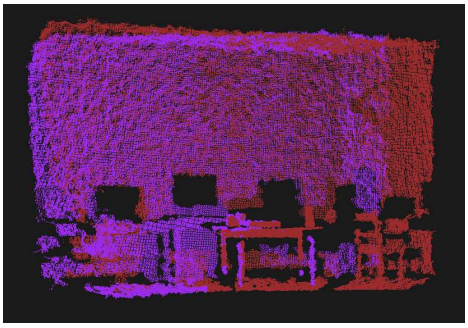
(a) 25% overlap



(b) 45%overlap



(c) 65%overlap



(d) 85%overlap

Figure 4.6: Different overlapping conditions evaluated in the test cases

the rotation error stays less than 1° in all scenarios.

In these Figures, we observe that a larger number of feature pairs produce more stable results in terms of translation error. For example, with an 85% of overlap, the algorithm seems to converge to a stable solution after 40 matching pairs are found.

As can be understandable, in the 25% of overlap condition, it is more challenging for the algorithm to reach a stable solution as a function of the number of matching feature pairs, however, it still can be observed that in general the graphs suggests that a higher number of features leads to a smaller translation error.

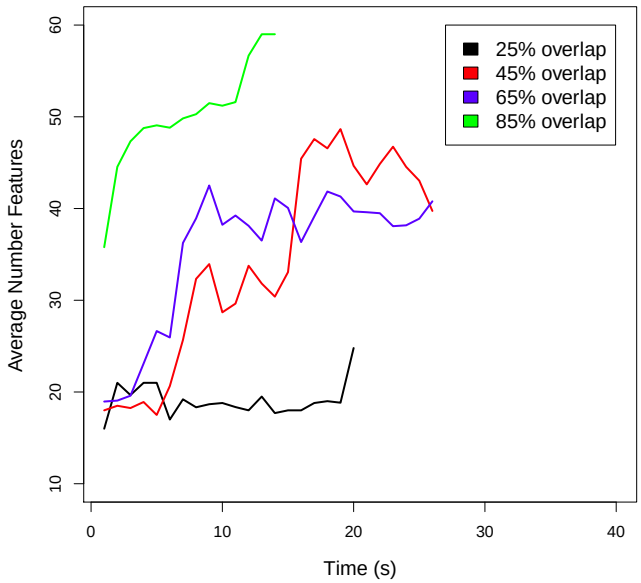


Figure 4.7: Behaviour of the algorithm in terms of the average number of successful feature pairs matched over extended periods of time.

4.2.2 Performance with regards to Registration Accuracy

In this section, we evaluate the algorithm by comparing the translation and rotation error over time. Errors are obtained by comparing our method to the standard calibration method using Nicolas Burrus RGBDemo software [13]. For this purpose, several snapshots of a calibration target(a chessboard) at different orientations should be taken. Then, the images are given as input to the software and extrinsic calibration parameters can be saved in to a file. The outputs of the calibration are the intrinsic parameters of the camera and the extrinsic parameters which are a 3×3 rotation matrix and a 3×1 translation matrix.

For this purpose, four different overlaps of 25%, 45%, 65% and 85% are used in

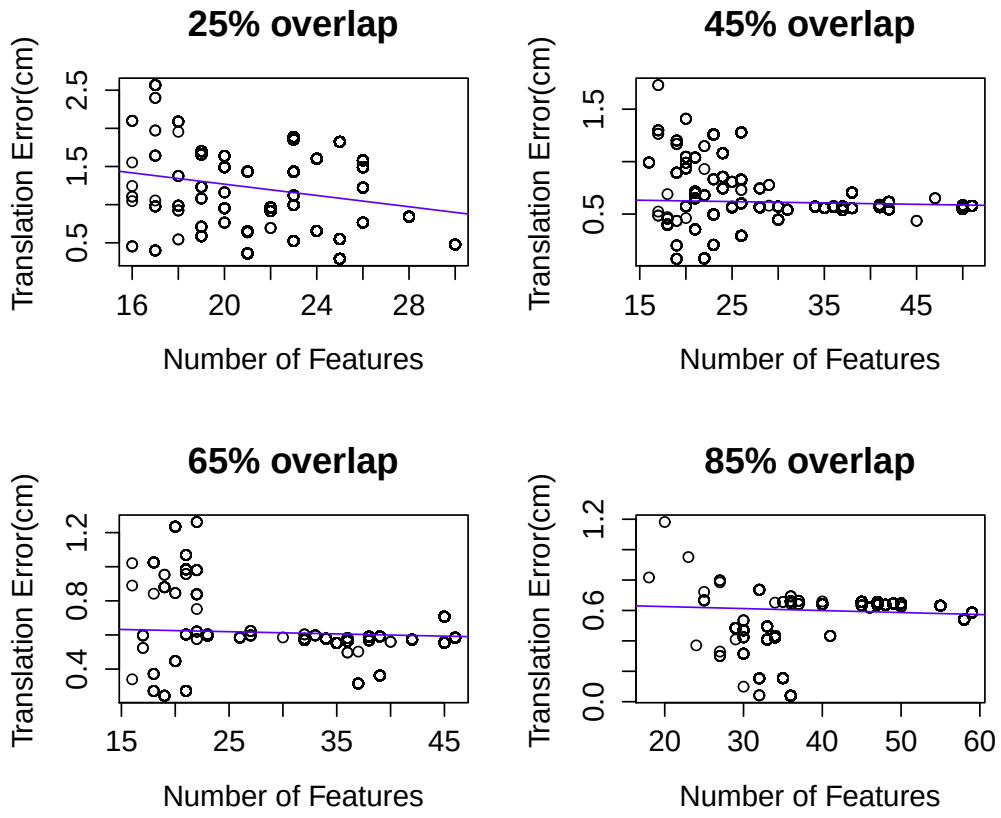


Figure 4.8: Behaviour of the algorithm in terms of translation error as a function of the number of features under varying overlapping conditions.

our measurements. For each of the overlapping conditions, we first calibrated the cameras and recorded the translation and rotation information over an average of 40 seconds. Then, the same camera positions are used for measuring the translation and rotation information for the same amount of time using our algorithm. Figure 4.6 shows the captured environment with different amounts of overlap.

Figure 4.10 shows translation error (in cm) over time (in seconds). Figure 4.11 shows the rotation error for the same overlapping conditions over time.

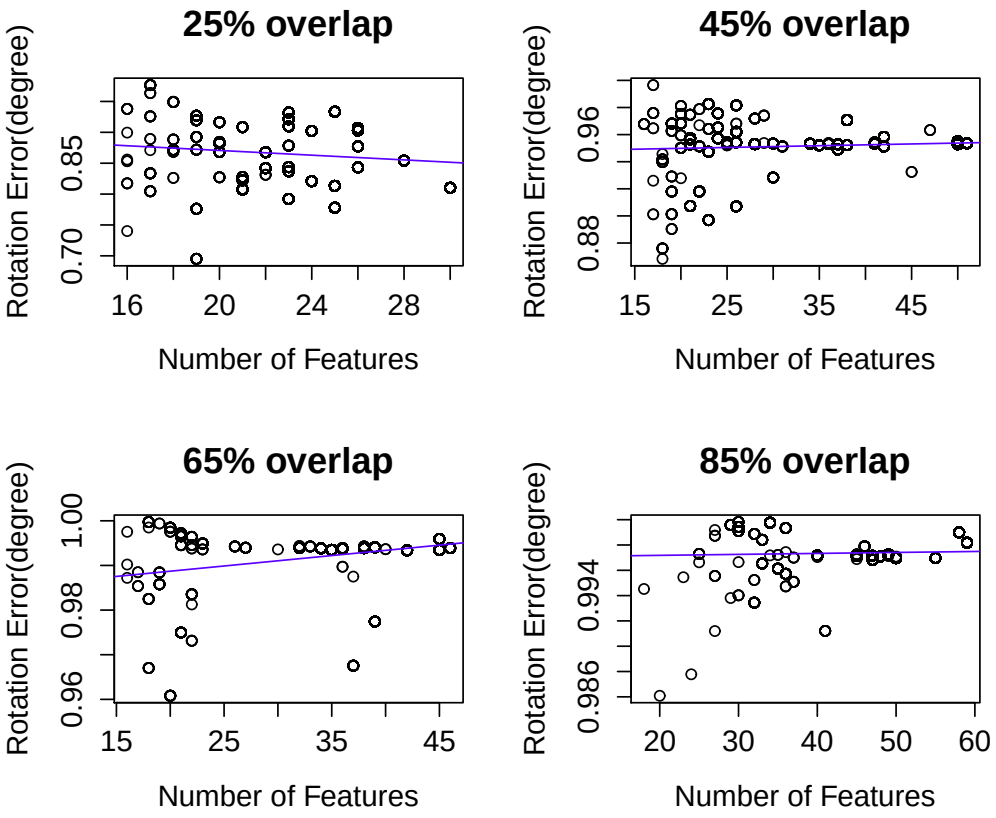


Figure 4.9: Improvement of the algorithm in terms of rotation error based on the number of features under varying overlapping conditions.

In order to illustrate a general trend, Figure 4.12 shows the translation error over time for different overlapping conditions separately. The red line shows the mean value of the translation error in each condition. The blue line is the fitted model of the data. As can be seen, the 25% condition suffers from the highest translation error among all with the mean value of $1.09cm$; however, there is a slight downward trend towards improvement over time. It is important to mention that the results under this amount of overlap, can still be considered a successful registration compared to other

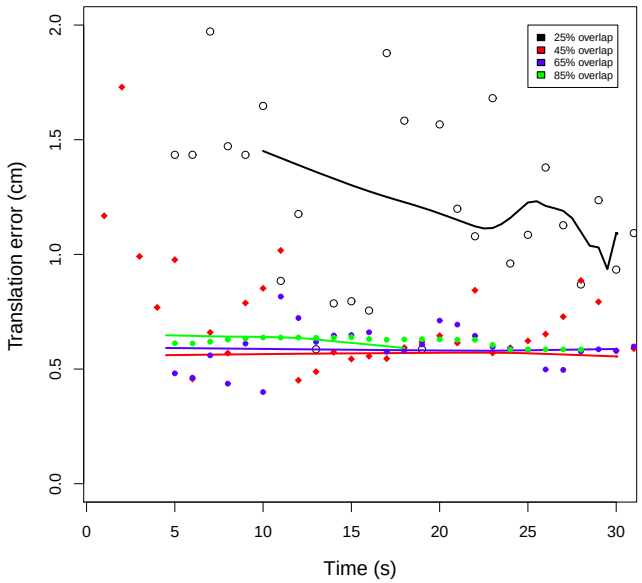


Figure 4.10: Translation error over time under varying overlapping conditions

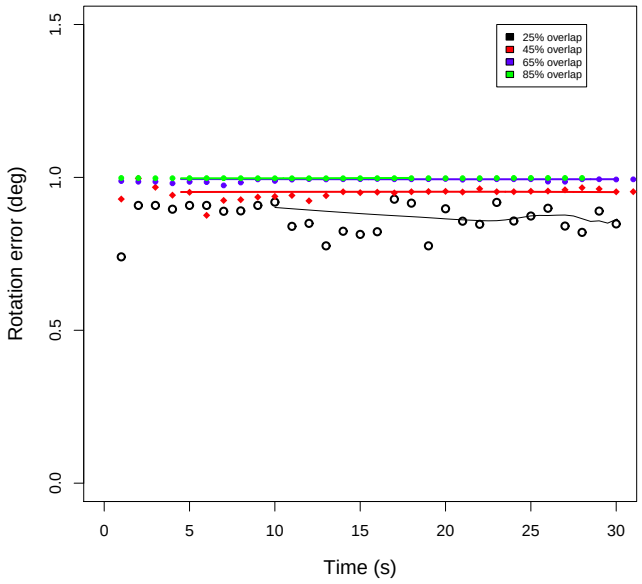


Figure 4.11: Rotation error over time under varying overlapping conditions

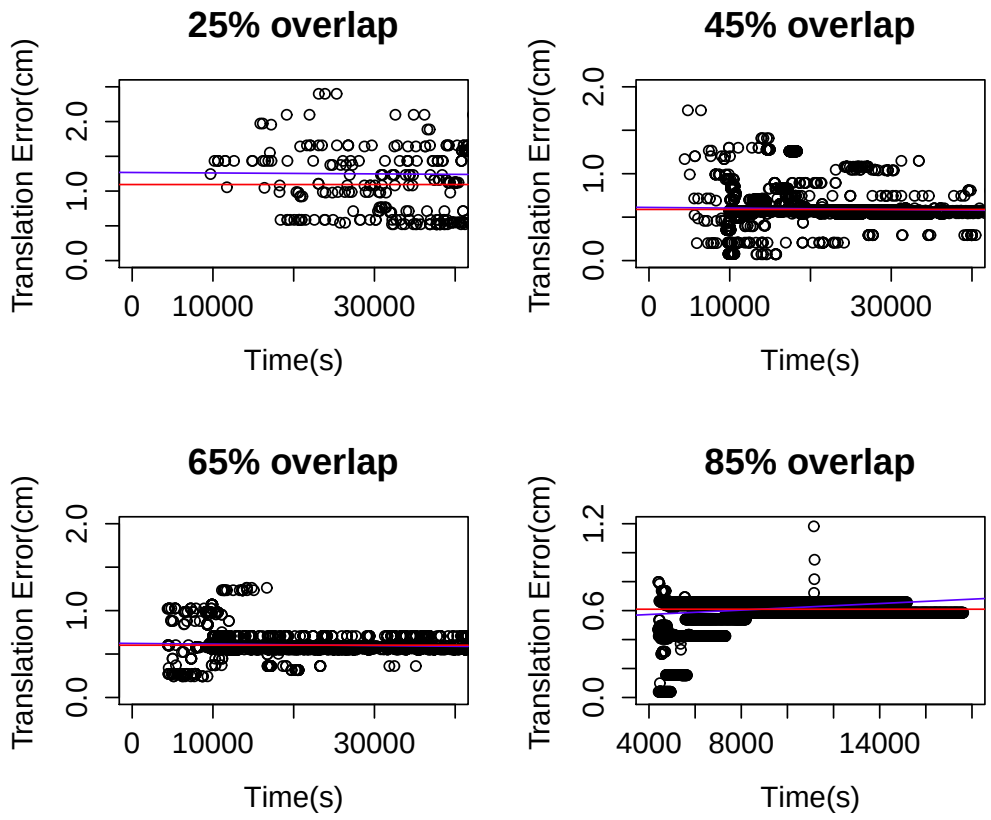


Figure 4.12: Translation Error over time under varying overlapping conditions shown separately.

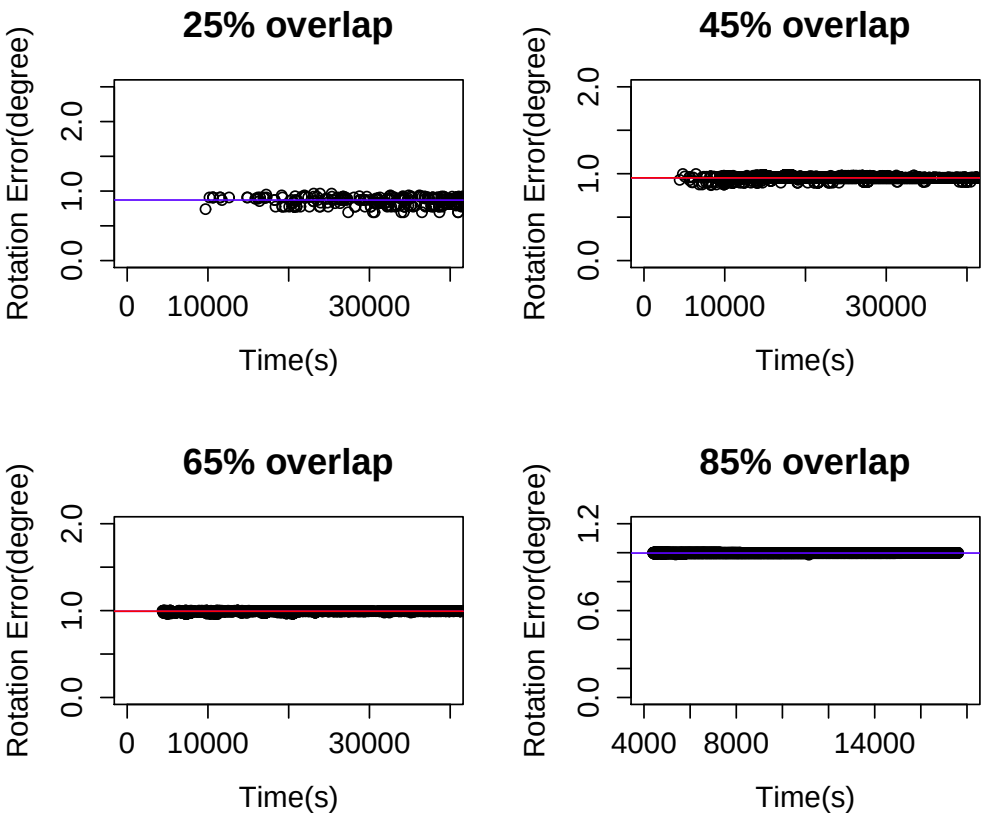


Figure 4.13: Rotation Error over time under varying overlapping conditions shown separately.

techniques. For example, according to table 2.5, Miller et al.'s lowest measurement in translation error is $2.75cm$. Figure 4.13 shows the rotation error over time for different overlap conditions. As can be observed, results show that rotation and translation error are kept to very low levels of less than $1cm$ in translation and 1° in rotation error.

We also compared our results to the work of Miller et al. As mentioned in Section 2, the work of [44] is the closest to what we achieved here in terms of alignment accuracy. A direct comparison to their results is not possible since we did not have access to their configuration and video sequences andno response was received from the authors of the paper when this data was requested; however, we used the same methodology to obtain the ground truth. We compared our results to the classic calibration method. Miller et al. also evaluated their results based on the same method. The results and sequence pictures mentioned in Table 4.1 of Miller et al.'s method are obtained from their paper [44].

As shown in Table 4.1, the results are very close to those obtained from the manual calibration method([13]) with an average of $0.63cm$ translation and 0.67° rotation errors. The results suggest that our algorithm outperforms the work of Miller et al., in terms of translation and rotation error, using the same method to obtain the ground truth. However, it is important to note that the video sequences and configuration parameters are not the same and thus the differences could be explained by other factors outside of our control. In addition, Miller et al. uses a crossed eye view setup where the scene background is different.

Our algorithm	Errort(cm)	Errorr(°)	Miller et al.	Errort(cm)	Errorr(°)
Figure 4.6(25% overlap)	1.094	0.86	Sequence 1	3.43	0.70
Figure 4.6(65% overlap)	0.601	0.992	Sequence 2	2.75	0.49
Figure 4.6(85% overlap)	0.609	0.997	Sequence 3	4.67	1.23

Table 4.1: The comparison between our algorithm and Miller et al. [44]. Results of Sequence 1 to 3 are retrieved from their article.

4.2.3 Performance with regards to Registration Speed

As mentioned in Chapter 3, the speed of the algorithm plays an important role for continuous video registration. Considering applications such as virtual reality or gaming, any delay will be a nuisance to users. In this section, we are evaluating the speed of the algorithm in different ways.

As Table 4.2 shows, we measured the execution times of different stages of the algorithm in a scenario of point clouds sharing 45% overlap and the execution times for camera movement detection. In all cases, 100 RANSAC iterations are used to obtain a coarse transformation estimation in step 2 of the algorithm. Since camera movement detection is implemented in a separate thread, its execution time does not influence the execution time of the dual camera registration. The numbers are an average of running the algorithm for 10 iterations. It is important to mention that no noticeable change in the speed in smaller or larger overlapping sections was observed. It takes $60.79ms$ on average for DeReEs-4V to find a registration for each two coming RGB plus point clouds and $63.62ms$ on average to perform camera movement detection, for an approximate frame rate of 16 frames per second.

Step	Camera Registration (ms)	Camera Movement Detection (ms)
Feature Matching	24.15	21.33
Coarse Transformation Estimation	35.59	40.32
Bad Features Removal	0.19	0.95
ICP Refinement	0.86	1.02
Total DeReEs-4V(ms)	60.79	63.62

Table 4.2: The speed of the different stages of DeReEs-4V measured on a case where the cameras are sharing a viewpoints of 45% overlap (Figure 1).

Table 4.3 shows the speed of the algorithm in different test cases, both for camera motion detection and dual camera registration, and the execution time across several iterations of DeReEs-4V until it is not possible to visually notice any misalignment artifacts, when a seamless registration of the point clouds is achieved.

4.3 Conclusion

In this chapter, we evaluated DeReEs-4V with respect to standard calibration. Experiments were conducted under varying amounts of overlap and DeReEs-4V was compared to calibration in terms of registration accuracy. The performance of the algorithm was also compared to the related work of [44] in terms of registration accuracy. Finally, an analysis of the speed of different stages of the system was presented.

Table 4.3: Execution times of camera motion detection and registration per iteration and across multiple iterations to achieve seamless registration, measured on different video stream capture scenarios: Case 1-One camera is located higher than the other (Figure 4.5). Case 2-Cameras are located far from each other with 45% overlap (Figure 4.1). Case 3-Two users with one being captured by only one of the cameras (Figure 4.2).

Algorithms	Case 1	Case 2	Case 3
Camera Movement Detection (ms)	68.55	63.42	72.75
DeReEs-4V Registration (ms)	77.81	66.40	57.73
DeReEs-4V Seamless Registration (seconds)	8.2	15.30	12.25

Chapter 5

Conclusion

In this chapter, we review our contributions and then discuss the potential future work to improve the system.

5.1 Contributions

Substantial research has been recently conducted in the field of computer vision to employ multiple depth sensing cameras to acquire a complete 3D model of the environment, or to improve accuracy and extend the field of view. Using multiple depth sensing cameras requires the calibration stage to be performed initially, and this has to be repeated if any of the cameras are displaced. Standard calibration techniques usually require a target such as chessboard which different snapshots of the target should be taken in different orientation. These methods are time consuming and may require experts to perform the task, making it unsuitable for non-expert users.

Our motivation in this research is to obtain a multiple depth sensing camera solution that does not require the standard calibration process while producing the

same accuracy. Besides accuracy, calibration has to be automatic, target free and perform in real time. In this research, we introduced DeReEs-4V, an automatic calibration technique that performs in real time ($60.79ms$ on average). Unlike other calibration techniques, our method is robust to camera movements. The system is designed to recognize camera movement and update the calibration when the cameras are moved. This is usually achieved in $63.62ms$ for a first approximation. The algorithm does not require the use of any targets, or motion from objects or people in the scene. Since it is an image based registration method, its only requirement is to be in an indoor space that is sufficiently illuminated with some identifiable items in the scene.

The point cloud that is produced after registration of the views from both cameras can be used in applications such as human tracking and gaming. To illustrate how the output of DeReEs-4v can be used in a wide variety of applications, we used and modified the Plane Model Segmentation Library in the Point Cloud Library [53]. As can be seen from Figure 5.1, the background and the person in the foreground are shared in both viewing regions and the colors show that they are regarded as a unified point cloud where each different colour shows a segmented part based on a foreground/background segmentation using the unified point cloud data from both cameras.

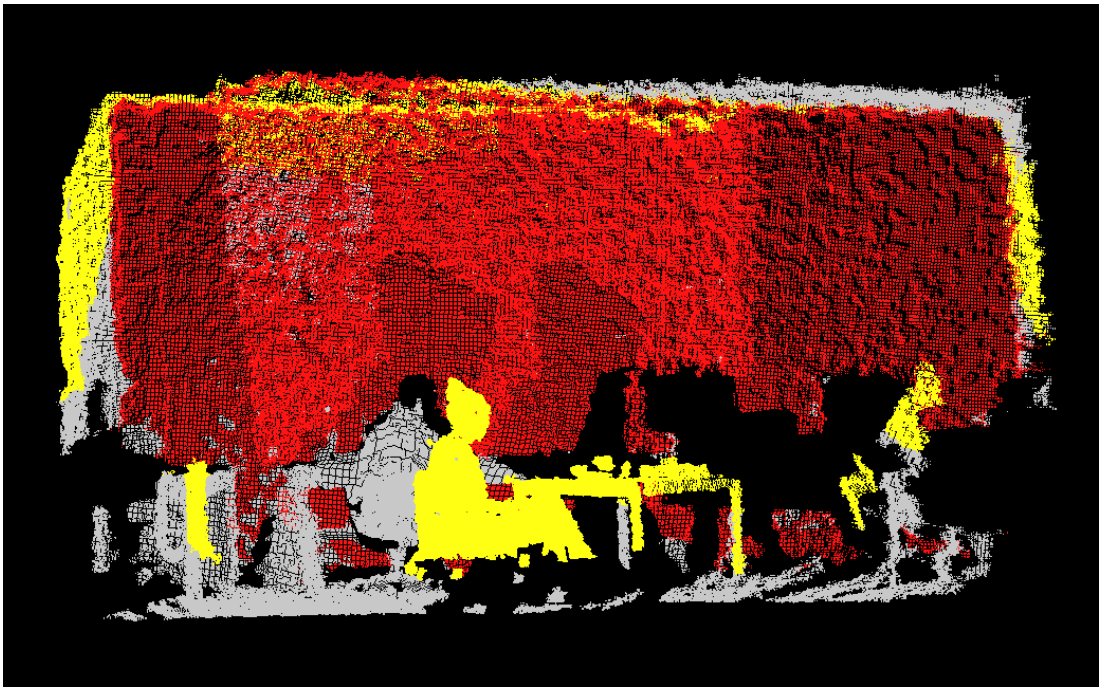


Figure 5.1: An example of plane segmentation method performed on the registered point cloud

In our experiments, we tested the algorithm in different scenarios where one, two or three users are present in the overlapping, non-overlapping or boundaries of the viewing regions of each camera. We also evaluated the method against the standard calibration technique and the work of Miller et al. [44]. For these experiments we evaluated it under different conditions of overlap(25%, 45%, 65% and 85%) and compared the translation and rotation accuracy to the two mentioned methods for 10 iterations in each overlap condition. The results show successful registration with an average of $0.63cm$ translation and 0.67° rotation error. The results show that, in terms of registration accuracy, the metric of obtaining a larger number of feature pairs leads to stable results under most conditions of overlap.

5.2 Limitations and Possible Improvements

In this system, Kinect cameras are employed; consequently, the limitations of the Kinect cameras are relevant. First of all, Kinect cameras are designed for indoor applications. Outdoor application of DeReEs-4V is unknown and may require using a different type of camera. Moreover, Kinect cameras are suitable for capturing objects further than $50cm$. This limits the application of DeReEs-4V for close-ranged applications, such as desktops, laptops or mobile phones.

The system proposed here could be used to register video from depth sensing cameras with smaller depth-sensing range, such as Intel's Creative [33], or it could be used to facilitate the combination of different types of cameras. For instance, a new type of stereo camera that produces RGB and Depth information along with point clouds has been introduced. ZED cameras [32], have a range of 1.5m to 20m that can capture in outdoor space as well as indoors. Exploring DeReEs-4V using this new type of camera can be an interesting research topic.

Another important limitation is that DeReEs-4V is an image based registration method. Therefore, it is highly dependent on colour information and the lighting conditions of the environment. Being a feature based registration also restricts the algorithm to perform with overlapping viewing regions of at least 25%.

Many potential extensions to this work and applications can be envisioned, for example, active people tracking over the extended field of view of both sensors could be explored, and other algorithms that can make use of the registered point cloud as a whole. To improve the rendering quality, hole-filling and point-rendering algorithms can be used to present a smoother display of the environment.

In terms of applications, more point cloud processing algorithms could be used on the resulting point cloud produced after registration, and the system could be used to support more users fitting inside the wider field of view, or can provide a wider range of motion for a single user, which could be of benefit for motion capture and gaming, robot and computer vision, simulation, training, and health-related applications.

Bibliography

[1] N. Ahmed. A system for 360 degree acquisition and 3d animation reconstruction using multiple rgb-d cameras. In *International Conference on Computer Animation and Social Agents*, 2012.

[2] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze. Point cloud library. *IEEE Robotics & Automation Magazine*, 1070(9932/12), 2012.

[3] S. Alexiadis, G. Kordelas, K. C. Apostolakis, J. D. Agapito, J. Vegas, E. Izquierdo, and P. Daras. Reconstruction for 3d immersive virtual environments. In *Image Analysis for Multimedia Interactive Services (WIAMIS), 2012 13th International Workshop on*, pages 1–4. IEEE, 2012.

[4] R. Altamimi and G. Skinner. A survey of active video game literature. *Journal of Computer and Information Technology*, 1(1):20–35, 2012.

[5] K. Arun, T. Huang, and S. Blostein. Least-squares fitting of two 3-d point sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-9(5):698–700, Sept 1987.

[6] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[7] S. Beck, A. Kunert, A. Kulik, and B. Froehlich. Immersive group-to-group telepresence. *Visualization and Computer Graphics, IEEE Transactions on*, 19(4):616–625, 2013.

[8] K. Berger. A state of the art report on multiple rgb-d sensor research and on publicly available rgb-d datasets. In *Computer Vision and Machine Learning with RGB-D Sensors*, pages 27–44. Springer, 2014.

[9] K. Berger, K. Ruhl, Y. Schroeder, C. Bruemmer, A. Scholz, and M. A. Magnor. Markerless motion capture using multiple color-depth sensors. In *Vision, Modeling, and Visualization (VMV)*, pages 317–324, 2011.

[10] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[11] J.-Y. Bouguet. Camera calibration toolbox for matlab. `https://www.vision.caltech.edu/bouguetj/calib_doc/`, 2004. Accessed: September 2015.

[12] G. Bradski. Dr. dobb's journal of software tools. `http://www.opencv.org`, 2000. Accessed: October 2015.

[13] N. Burrus. Rgbdemo: Demo software to visualize, calibrate and process kinect cameras output. `https://github.com/rgbdemo`, 2012. Accessed: February 2015.

[14] M. Caon, Y. Yue, J. Tscherrig, E. Mugellini, and O. A. Khaled. Context-aware 3d gesture interaction based on multiple kinects. In *Proceedings of The First International Conference on Ambient Computing, Applications, Services and Technologies, AMBIENT*, 2011.

[15] X. Chen, J. Davis, and P. Slusallek. Wide area camera calibration using virtual calibration objects. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 520–527. IEEE, 2000.

[16] E. Cippitelli, S. Gasparrini, S. Spinsante, and E. Gambi. Kinect as a tool for gait analysis: Validation of a real-time joint extraction algorithm working in side view. *Sensors*, 15(1):1417–1434, 2015.

[17] S. Czarnuch and M. Ploughman. Automated gait analysis in people with multiple sclerosis using two unreferenced depth imaging sensors: Preliminary steps. In *Newfoundland Electrical and Computer Engineering Conference, IEEE, Newfoundland and Labrador Section*. IEEE, 2014.

[18] R. Deklerck, B. Jansen, X. Yao, and J. Cornelis. Automated estimation of 3d camera extrinsic parameters for the monitoring of physical activity of elderly patients. In *XII Mediterranean Conference on Medical and Biological Engineering and Computing 2010*, pages 699–702. Springer, 2010.

[19] D. Demirdjian, A. Zisserman, and R. Horaud. Stereo autocalibration from one plane. In *Computer Vision—ECCV 2000*, pages 625–639. Springer, 2000.

[20] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM transactions on graphics (TOG)*, 21(3):257–266, 2002.

[21] K. Essmaeel, L. Gallo, E. Damiani, G. De Pietro, and A. Dipandà. Multiple structured light-based depth sensors for human motion analysis: a review. In *Ambient Assisted Living and Home Care*, pages 240–247. Springer, 2012.

[22] K. Essmaeel, L. Gallo, E. Damiani, G. De Pietro, and A. Dipanda. Comparative evaluation of methods for filtering kinect depth data. *Multimedia Tools and Applications*, pages 1–24, 2014.

[23] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[24] S. Foix, G. Alenya, and C. Torras. Lock-in time-of-flight (tof) cameras: a survey. *Sensors Journal, IEEE*, 11(9):1917–1926, 2011.

[25] S. Fuchs and S. May. Calibration and registration for precise surface reconstruction with time-of-flight cameras. *International Journal of Intelligent Systems Technologies and Applications*, 5(3-4):274–284, 2008.

[26] A. L. Fuhrmann, J. Kretz, and P. Burwik. Multi sensor tracking for live sound transformation. *Proceedings of New Interfaces for Musical Expression, Daejeon, Korea*, pages 358–362, 2013.

[27] M. Gabel, R. Gilad-Bachrach, E. Renshaw, and A. Schuster. Full body gait analysis with kinect. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 1964–1967. IEEE, 2012.

[28] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012.

[29] D. Herrera, J. Kannala, and J. Heikkilä. Accurate and practical calibration of a depth and color camera pair. In *Computer analysis of images and patterns*, pages 437–445. Springer, 2011.

[30] M. Hödlmoser and M. Kampel. Multiple camera self-calibration and 3d reconstruction using pedestrians. In *Advances in Visual Computing*, pages 1–10. Springer, 2010.

[31] M. Hossny, D. Filippidis, W. Abdelrahman, H. Zhou, M. Fielding, J. Mullins, L. Wei, D. Creighton, V. Puri, and S. Nahavandi. Low cost multimodal facial recognition via kinect sensors. In *Proceedings of the land warfare conference (LWC): potent land force for a joint maritime strategy. Commonwealth of Australia*, pages 77–86, 2012.

[32] S. Inc. Zed stereo camera. `https://www.stereolabs.com/`, 2015. Accessed: September 2015.

[33] C. S. Intel. Creative senz3d. `http://us.creative.com/p/web-cameras/creative-senz3d`, 2013. Accessed: 2015-04-09.

[34] A. Kadambi, A. Bhandari, and R. Raskar. 3d depth cameras in vision: Benefits and limitations of the hardware. In *Computer Vision and Machine Learning with RGB-D Sensors*, pages 3–26. Springer, 2014.

[35] K. Kamarudin, S. M. Mamduh, A. Y. M. Shakaff, and A. Zakaria. Performance analysis of the microsoft kinect sensor for 2d simultaneous localization and mapping (slam) techniques. *Sensors*, 14(12):23365–23387, 2014.

[36] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[37] S. Koenig, A. Ardanza, C. Cortes, A. De Mauro, and B. Lange. Introduction to low-cost motion-tracking for virtual rehabilitation. In *Emerging Therapies in Neurorehabilitation*, pages 287–303. Springer, 2014.

[38] Y. Liu, G. Ye, Y. Wang, Q. Dai, and C. Theobalt. Human performance capture using multiple handheld kinects. In *Computer Vision and Machine Learning with RGB-D Sensors*, pages 91–108. Springer, 2014.

[39] R. Lun and W. Zhao. A survey of applications and human motion recognition with microsoft kinect. *International Journal of Pattern Recognition and Artificial Intelligence*, 2015.

[40] R. Macknojia, A. Chávez-Aragón, P. Payeur, and R. Laganiere. Calibration of a network of kinect sensors for robotic inspection over a large workspace. In *Robot Vision (WORV), 2013 IEEE Workshop on*, pages 184–190. IEEE, 2013.

[41] A. Maimone and H. Fuchs. Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 137–146. IEEE, 2011.

[42] A. Maimone and H. Fuchs. A first look at a telepresence system with room-sized real-time 3d capture and life-sized tracked display wall. *Proceedings of ICAT 2011, to appear*, pages 4–9, 2011.

[43] S. Milani and G. Calvagno. Joint denoising and interpolation of depth maps for ms kinect sensors. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 797–800. IEEE, 2012.

[44] S. Miller, A. Teichman, and S. Thrun. Unsupervised extrinsic calibration of depth sensors in dynamic scenes. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2695–2702. IEEE, 2013.

[45] M. Nakazawa, I. Mitsugami, Y. Makihara, H. Nakajima, H. Habe, H. Yamazoe, and Y. Yagi. Dynamic scene reconstruction using asynchronous multiple kinects. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 469–472. IEEE, 2012.

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

[47] S. M. Olesen, S. Lyder, D. Kraft, N. Krüger, and J. B. Jessen. Real-time extraction of surface patches with associated uncertainties by means of kinect cameras. *Journal of Real-Time Image Processing*, 10(1):105–118, 2012.

[48] S. Paris, P. Kornprobst, and J. Tumblin. *Bilateral Filtering*. Now Publishers Inc., Hanover, MA, USA, 2009.

[49] N. Rafibakhsh, J. Gong, M. K. Siddiqui, C. Gordon, and H. F. Lee. Analysis of xbox kinect sensor data for use on construction sites: depth accuracy and sensor interference assessment. In *Constitution research congress*, pages 848–857, 2012.

[50] A. Rafighi, S. Seifi, and O. Meruvia-Pastor. Automatic and adaptable registration of live rgbd video streams. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, MIG '15, pages 243–250, New York, NY, USA, 2015. ACM.

[51] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[52] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001.

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

[54] S. Seifi. Derees: real-time registration of rgbd images using image-based feature detection and robust 3d correspondence estimation and refinement. Includes bibliographical references (pages 104-115)., September 2014.

[55] S. Seifi, A. Rafighi, and O. Meruvia-Pastor. Derees: Real-time registration of RGBD images using image-based feature detection and robust 3d correspondence estimation and refinement. In *Proceedings of the 29th International Conference*

on Image and Vision Computing New Zealand, IVCNZ 2014, Hamilton, New Zealand, November 19-21, 2014, page 136, 2014.

[56] J. Shen, W. Xu, Y. Luo, P.-C. Su, and S.-C. S. Cheung. Extrinsic calibration for wide-baseline rgb-d camera network. In *Multimedia Signal Processing (MMSP), 2014 IEEE 16th International Workshop on*, pages 1–6. IEEE, 2014.

[57] D. A. Simon. *Fast and Accurate Shape-based Registration*. PhD thesis, Pittsburgh, PA, USA, 1996. AAI9838226.

[58] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics (TOG)*, volume 25, pages 835–846. ACM, 2006.

[59] A. Staranowicz and G.-L. Mariottini. A comparative study of calibration methods for kinect-style cameras. In *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*, page 49. ACM, 2012.

[60] W. Susanto, M. Rohrbach, and B. Schiele. 3d object detection with multiple kinects. In *Computer Vision–ECCV 2012. Workshops and Demonstrations*, pages 93–102. Springer, 2012.

[61] T. Svoboda, H. Hug, and L. Van Gool. Viroom—low cost synchronized multicamera system and its self-calibration. In *Pattern Recognition*, pages 515–522. Springer, 2002.

[62] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images.

In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.

[63] G. Tuna, K. Gulez, V. Gungor, and T. Veli Mumcu. Evaluations of different simultaneous localization and mapping (slam) algorithms. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pages 2693–2698, Oct 2012.

[64] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318. ACM, 1994.

[65] D. Webster and O. Celik. Systematic review of kinect applications in elderly care and stroke rehabilitation. *J. Neuroeng. Rehabil*, 11(1):108, 2014.

[66] A. D. Wilson and H. Benko. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 273–282. ACM, 2010.

[67] C. Zhang, Q. Cai, P. A. Chou, Z. Zhang, and R. Martin-Brualla. Viewport: A distributed, immersive teleconferencing system with infrared dot pattern. *MultiMedia, IEEE*, 20(1):17–27, 2013.

[68] C. Zhang and Z. Zhang. Calibration between depth and color sensors for commodity depth cameras. In *Computer Vision and Machine Learning with RGB-D Sensors*, pages 47–64. Springer, 2014.

[69] D. Zhang, Y. Yao, D. Liu, Y. Chen, and D. Zang. Kinect-based 3d video conference system. In *Global High Tech Congress on Electronics (GHTCE), 2013 IEEE*, pages 165–169. IEEE, 2013.

[70] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

[71] B. Zhao, P. An, C. Liu, J. Yan, C. Li, and Z. Zhang. Inpainting algorithm for kinect depth map based on foreground segmentation. *Journal of Electronics (China)*, 31(1):41–49, 2014.

[72] Y. Zou, W. Chen, X. Wu, and Z. Liu. Indoor localization and 3d scene reconstruction for mobile robots using the microsoft kinect sensor. In *Industrial Informatics (INDIN), 2012 10th IEEE International Conference on*, pages 1182–1187. IEEE, 2012.