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

by

© *Sahand Seifi*

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of *Science*

Department of *Computer Science*

Memorial University of Newfoundland

September 2014

St. John's Newfoundland

Abstract

We present DeReEs, a real-time RGBD registration algorithm for the scenario where multiple RGBD images of the same scene are obtained from depth-sensing cameras placed at different viewpoints, with partial overlaps between their views. DeReEs (Detection, Rejection and Estimation) is a combination of 2D image-based feature detection algorithms, a RANSAC based false correspondence rejection and a rigid 3D transformation estimation. DeReEs performs global registration not only in real-time, but also supports large transformation distances for both translations and rotations. DeReEs is designed as part of a virtual/augmented reality solution for a remote 3D collaboration system that does not require initial setup and allows users to freely move the cameras during use. We present comparisons of DeReEs with other common registration algorithms. Our results suggest that DeReEs provides better speed and accuracy especially in scenes with partial overlapping.

Acknowledgements

I would like to begin by thanking my academic supervisor, Dr. Oscar Meruvia-Pastor, for taking me in Memorial University, dedicating his resources and time to make this research possible and for his collaborative approach with his students.

I would also like to thank the Research and Development Corporation (RDC) of Newfoundland and Labrador for providing the funding used for this research and the Core Research Equipment and Instrument Training Network (CREAIT), especially Marc Bolli, for providing the spacious and advanced 3D Visualization lab in which I conducted the major part of this research.

I would also like to acknowledge the efforts of Afsaneh Rafighi for her sincere help with the data-set collection and Dr. Paul Gillard for his friendly guidance, sense of humour and the tough Computer Graphics course.

Thanks to my parents, Farhoud and Khadijeh, who did everything in their power to make this possible and for shaping me into who I am today. Thanks to my brother Babak, who initiated my interest in programming and encouraged me at every stage of my life to stick to my passion and continuous learning. Finally, thanks to all the people at MUN and especially my friends, whom I proudly call my second family, for their non-stop support and love and making the journey as meaningful as life can be.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

3D imaging and visualization has been used for many years in medical imaging (such as in ultrasound, CT and MRI scans), aerial imagery and radar and sonar scanners. Due to the high costs and complexity of such 3D imaging devices, applications of computer vision were mainly focused on using conventional 2D image cameras for visual input until recent years when this approach changed drastically with the introduction of low cost off-the-shelf depth sensing cameras, followed by their open-source SDKs which spawned a wave of publications and projects in the academia, open-source communities and the industry.

Depth sensing cameras provide 3D images, with each pixel representing the distance of a point in the scene to the camera. In some commercial devices, an integrated RGB camera also provides colour information for each point. Having the depth of a pixel, the real world coordinates of each point can be calculated by knowing the camera's horizontal and vertical field of view. This can be used to generate a 3D model of a scene. In other words, while 2D cameras provide us with colour information that represents a view of the scene, 3D cameras provide us with colour and coordinate information that represent a 3D scan of the scene.

Depth sensing cameras are used to 3D scan the environment. For example they are used in 3D scanners to build 3D models of physical objects or landmarks ([35], [14], [63]) which can be used for 3D printing, animation creation, real world modeling, etc. Having the 3D model of a scene, programs can detect and track objects and surroundings ([48], [31], [34]) which is useful in the field of robotics and autonomous vehicles for the purposes of manufacturing, localization and navigation through obstacles.

They can also identify the presence of a human body ([69]) in the scene along with its pose and gestures ([38]) which itself has been a source of innovation in the field of human-computer interaction and health-care. Depth cameras are used as input devices for a number of popular video game consoles for hands-free control of the video game and the players' avatars. They can also be used in health care to assist diagnosis, rehabilitation and monitoring of care receivers [17].

Depth cameras can also be used in virtual and augmented reality scenarios where the cameras can identify elements such as people, objects and the environment and then build avatars and representations of them in the virtual reality environment; or to augment their visualization with text or visual information in an augmented reality application.

The motivation behind this research is to create an immersive virtual reality collaboration environment using depth cameras. This would allow multiple groups of remote people to be present in one virtual environment as their avatars - which can be the 3D model of their physical body - to communicate and collaborate. They can move and walk around the virtual environment as they would in real world and see other participants in their 3D screens or head-mounted 3D monitors. Objects from the real world or purely virtual objects would exist in this environment as their 3D representation and participants would be able to interact with them using their hands or gestures. This can be useful for remote education and remote team collaboration especially when sharing and visualizing 3D models as protein molecules, 3D seismic data or models of mechanical parts with the whole team.

While such applications of depth sensing cameras are already possible, certain limitations exist: the 3D models of a scene or object generated from a single 3D image would be partial due to occlusions from objects in the scene and the limited field of view of the camera [66]. For example the 3D model generated from a camera in front of a participant would only include points representing the front side of the participant's body. With a static scene, a single camera can be moved around to accumulate 3D images from different poses into a single more complete model. In dynamic scenes, multiple cameras must be used at once to capture the scene and participants from different angles.

In either case, to combine the 3D images of a scene that are taken from different points of view (either at the same time or at different times), their relative pose must be known. For example, to combine two 3D images taken from an object from completely opposite sides, one of the 3D models should be rotated 180 degrees to match the other model. This is similar to generating 2D panoramic images by stitching two or more frames together: each frame is shifted to the left or right to align with the previous frame. In 3D models, the relative pose between two 3D images can be described as a 3D transformation with 3 parameters for displacement (translation) over the x, y and z axes and 3 parameters for rotation around each of the axes.

For the purpose of our proposed Virtual Reality application, we have to use multiple depth cameras to create complete models of our dynamic scenes. It is possible to find the relative pose and align the models from multiple cameras manually by trial and error. This can be done by visualizing all the models, then using mouse and keyboard controls to move and rotate them until they match. However, this is a tedious task that can be confusing and very time consuming considering there are 6 degrees of freedom (3 axes of translation and 3 directions for rotation) to match each pair of models.

Another problem arises when any of the cameras move even in the smallest amount. In this case the models will be misaligned which is visually unappealing and the whole manual process needs to be re-done to achieve alignment. This greatly degrades user experience if it happens during a collaboration session. In the proposed VR application, the position of the cameras may change from day to day to capture different parts of the environment or even with users slightly colliding and bumping into the cameras. We are interested in a solution where the setup process for each use of the application would be hassle free and also the cameras can move freely during use without breaking the alignment.

This can be achieved if finding the 3D transformation that aligns the input models together can be automated. In the field of computer vision, this problem is referred to as registration or alignment. Registration is the process of estimating the transformation between two images. 3D Registration algorithms would receive a pair of 3D models as input and generate the 3D transformation that converts the coordinates of one of the models (the source model) with the other (the target model), so that they are visually aligned together.

The transformation between two 3D images is actually the physical transformation between the positions of the camera(s) when the images were taken. Hence, as well as being used for creating complete models of a scene or an object, these algorithms are of great importance in robot localization. In scenarios when a depth camera is setup on a robot, the transformation between two frames taken at times t_1 and t_2 tells us how the robot has moved since t_1 and where it is currently located at time t_2.

For our application, registration algorithms can be performed for each pair of input streams at every frame, so all the inputs are aligned together even in case of camera movement. Unfortunately, existing registration algorithms are limited in terms of robustness and performance:

1) Some registration algorithms such as ICP ([5], [12]) can only tolerate small transformation distances between the 3D image pairs. In other words, they only perform well if the two 3D image pairs have very similar initial pose with minimal translation or rotation (e.g. when cameras are placed very close together). These algorithms are not suited for our intended use, since we are assuming that we will not be using too many cameras to capture a scene, hence the cameras have noticeable distances from each other. For example, 6 cameras might be used to capture a 360 degree view of the participants.

2) Other registration algorithms such as 3D-NDT ([37]) are not as restrictive, but they do not perform in real time. Proper registration method for the application of our interest must work in real-time to align frames instantly in case of camera movement during use.

3) Furthermore, many of the existing algorithms are designed with the assumption

that the 3D image pairs have major amounts of overlap. In other words, it is assumed that the majority of the points in the two models represent the same parts of the scene. This causes these algorithms to fail registration in scenarios when the images have noticeable non-overlapping (exclusive) portions, which can happen even with small transformations, especially with camera rotation. This usually happens because these algorithms aim to minimize a distance based error metric such as the sum of absolute distances between the points of the two 3D images. In the case of large non-overlapping portions, this type of metrics produces large undesired errors even with the right transformation, because the right transformation essentially puts the non-overlapping parts of the clouds away from each other.

This research is dedicated to introducing a registration technique that would perform in real-time with 6 degrees of freedom and minimal restrictions on the cameras initial poses and the amount of overlap between the images. The proposed registration technique would be usable with common off-the-shelf RGBD cameras.

1.1 Research Questions and Hypotheses

Following are the research questions with respect to the requirements of this research and related hypotheses.

1. Is it possible to register 3D images with large amounts of transformation distance and large non-overlapping portions? We are assuming that for the purpose of this work, two or three cameras are used to capture a scene or character with the goal of providing a wider field of view. This implies that the cameras are positioned far from each other and/or might be close to each other but are

pointing towards different directions such that only a small fraction of the scene is shared by both cameras' views. We are aware that some existing registration algorithms can tolerate relatively large transformation distances to some extent if the image pairs are capturing mostly the same scene, but none that we are aware of tolerates relatively large non-overlapping portions where noticeable amounts of mutually exclusive parts exist in both point clouds. We hypothesize that our proposed algorithm tolerates large transformation distances ($3-5$ meters in translation $30-50$ degrees for rotation) and images with only 50% overlap or more.

2. Is it possible to have a real-time registration method? We hypothesize that using GPU implementation our proposed algorithm will run at a rate such that both initial alignment of the collaborative environment and maintaining alignment in case of camera movement during use of the system would be performed instantly. For effectiveness of our algorithm in such a scenario, we are expecting the algorithm to perform at the rate of at least 12 frames per second.

3. Is it possible to have a hassle-free user experience where there is no need for user input regarding the camera setup and alignment? The main purpose of proposing a new registration technique for our scenario is to remove human assisted setup and make the collaboration experience as automated as possible. We think that the proposed registration technique can successfully initiate the registration even in case of large camera distances, without user's intervention. The number of the cameras and their relative position and orientation (or pose) can be estimated in real-time without any user's assistance. The same is true

for maintaining the alignment during the session. Since the pose estimation is performed in real-time, in case of changes to camera's physical setup, the system will automatically adapt to the change.

4. Does the registration rely on optimal scene configuration in terms of feature-richness, brightness and repetitive patterns? The accuracy of the algorithm with non-optimal scenes such as dimmed environments, feature-less scenes or scenes with repetitive patterns, heavily relies upon the quality of the correspondence detection of the algorithm. In this regard, we have three hypotheses:

    - The image registration system proposed can perform adequately under varying conditions of feature-richness in the scene.
    - The image registration system proposed can perform adequately under varying conditions of brightness in the scene.
    - The image registration system proposed can perform adequately under varying conditions of repetitive patterns in the scene.

    In each of these cases, the risk of faulty registration increases due to incorrect feature detection and matching, but we believe that by using more conservative parameters we can achieve better results with the feature detection and matching algorithms to detect corresponding pairs, as well as performing a more rigorous false corresponding pair rejection. This will cause the algorithm to display an unnoticeable increase in execution time, still maintain its real-time performance. We have assessed these hypotheses through our experiments.

1.2 Contributions

The primary contribution of this research is a novel yet simple registration technique for 3D coloured image streams which we refer to as DeReEs. The secondary contribution is a data-set of 3D image sets, with each set including 2 to 5 images of one scene captured from different poses with known ground truth transformations which would be helpful to the research community for experimenting with and benchmarking registration algorithms (accessible at [56]).

The proposed algorithm (described in details in chapter 4) is performed through the following steps: 1) Corresponding Pair Detection, 2) False Corresponding Pair Rejection and 3) Transformation Estimation. The first step is based on existing 2D RGB feature detection and matching algorithms. The major contributions of this work lies in the second step which consists of: 1) a Random Sample Consensus (RANSAC [22]) based outlier detection method for detecting false corresponding pairs captured from the previous step which can be repeated iteratively for more accurate results and 2) an accurate metric for scoring proposed transformations regardless of the amount of overlapping. The third step consists of a simple transformation estimation based on remaining corresponding pairs after false pair rejection.

Each major part of this technique (feature detection and matching, outlier detection, scoring and transformation refinement) is fully independent and modular. Throughout this thesis we demonstrate how our choices for each of the steps (either from the existing solutions or our proposed methods) perform better than the alternatives we have considered so far.

Our experiments show that the proposed registration technique performs at 27 frames per second with GPU implementation of its components with a 640×480 frame resolution in indoor environments. It places minimal restrictions over the 6 degrees of freedom. It performs successfully with images of only 23% overlap and in challenging conditions such as: feature-less scenes, low lit environments and scenes with repetitive patterns.

The data-set (described fully in section 5.1) is captured in indoor lab and office environments with a Kinect camera at 640x480 resolution. The images are saved in PCD format which can be directly used with the open-source Point Cloud Library (PCL [53]). The scenes and camera poses are selected in a way to produce a data-set that is varied in many aspects: feature-richness, amounts of overlap, transformation distance, translation and rotation direction, distance of the camera from the objects and brightness. The ground truth transformation between the images of a scene are visually estimated with $0.5 degree$ and $1 cm$ accuracy and are saved as 4×4 transformation matrices in plain text.

1.3 Methodology

The robustness of registration algorithms is highly dependent on the many parameters of the image sets it is used with; parameters such as transformation distance between image pairs, visual or spatial feature-richness, brightness and distance from the camera. This made us take two major decisions for the evaluation of our work:

- Existing 3D data sets for registration and pose estimation did not satisfy the different conditions required for our different types of experiments or were limited in their number of frames, variety of scenes and transformation distance.

We generated our own data-set which resembles the scenarios that might occur with the virtual reality collaboration tool under varying circumstances.

- To better understand the shortcomings of the existing registration algorithms for our intended use, we experimented with their implementations using our data-set rather than relying on the reported evaluations in their respective publications, since those evaluations were done using a variety of configurations and data-sets.

For the implementation of the existing registration algorithms, we chose the Point Cloud Library (PCL) for being the standard open-source library for 3D model processing in the research community and because it includes implementations of the ICP and 3D-NDT algorithms.

To experiment with RGB based registration algorithms, we used the Open Source Computer Vision library (OpenCV [9]) for RGB feature and edge extraction. By combining the tools in OpenCV and PCL, we were also able to implement one of the existing feature-based registration algorithms (RGBD-ICP [30]) for experimentation purposes.

Based on our experiments, we understood early on that algorithms that use distance based error metrics are not suitable for large transformation distance and non-overlapping models. An RGB-based registration approach was chosen since it offered a less restricted solution. Unlike ICP and 3D-NDT, RGB-based solutions require having colour information of the model, but since our intended use is for Virtual Reality, colour information is available from the RGB enabled depth sensors.

Our proposed technique is based on existing RGB feature detection algorithms. For these algorithms, we used the OpenCV library for being the standard open source image processing library in the research community and also for having GPU implementations of multiple feature detection algorithms. Our implementation also uses the PCL library for IO tasks, visualization of the models and model processing.

Using these two libraries allows the research community to easily build their implementations of our algorithm without having to implement many of the existing parts of it, as well as providing the modularity which allows researchers to replace any of its components with their own work. Likewise, Microsoft^{TM} Xbox 360 Kinect camera was used as our depth sensor which is compatible with the PCL library and is regularly used by the research community in robotics and computer vision.

1.4 Outline

Chapter 2 explains the main concepts required for understanding this work, including: depth sensing cameras, point cloud processing, the registration problem, its applications and RGB feature detection. This chapter can be skipped for the more advanced readers. Chapter 3 discusses the work of other researchers that has been done for solving the registration problem, including ICP and its variants, 3D-NDT and feature-based registration methods, their advantages and limitations. Chapter 4 explains our proposed algorithm in depth, also discussing different alternatives and approaches at each of its steps and proposes possible improvements. Chapter 5 explains the process of data-set collection and ground truth transformation estimation, followed by experimentation and comparison of our proposed algorithm to determine the validity of our work and hypotheses. Finally, at chapter 6 we provide a summary of the contributions and conclusions made throughout the work and possible future work and further improvements.

Chapter 2

Preliminaries

In this chapter we will explain some of the concepts in computer vision and image processing required for better understanding the context of this research. Advanced readers who are familiar with these topics can skip to chapter 3 for an overview of existing registration algorithms or chapter 4 for a detailed explanation of our proposed algorithm.

2.1 Range Sensing

A regular digital camera provides us with frames that represent the scene. Each frame consisting of an array of pixels, with each pixel representing a point in the real world. In an RGB image each pixel is defined by Red, Green and Blue values which make up the colour of the point in the real world. With a depth sensor (also referred to as 3D cameras), the pixel is defined by a depth value which indicates the distance of the respective point in the real world to the camera. The accuracy of this value depends on the sensor itself; off-the-shelf sensors can be accurate up to millimeters. Many of the commercialized depth sensing cameras also provide an RGB image along with the depth image.

Estimating the depth of scene elements to a sensor, also known as “range sensing”, is possible through a variety of methods. All of the approaches used for range sensing depend on emitting a signal (radio, light, etc.), reading its response from its reflection on the scene and analyzing it against the original signal. This makes most of these approaches vulnerable to reflective surfaces, since for specular surfaces the majority of the original signal is reflected in a direction other than the sensor, which is usually located close to the emitter; with the exception of where the specular surface is perpendicular to the direction of the signal itself. For diffuse surfaces, the signal is scattered back in many different directions, improving the chances that the sensor will pick up its response.

Below are some of the primary methods that have been used for estimating the depth value (also referred to as Z value) in traditional and recent range sensing devices.

2.1.1 Radar, Sonar and Lidar

Radar (short for Radio Detecting and Ranging) has been widely used specially in military and safety. Radars use an emitter to send a radio wave to the desired direction and sense the reflection. While the long wave length of the radio signals has an inverse effect on the resolution of the estimation, radio waves penetrate most materials, except for metals, seawater and wetlands and have a long range, making it desirable for aerial or sea vehicles. Using the Doppler effect, the movement of an object can be detected by the slight change in the frequency of the response signal

[37].

Sonar devices work similar to Radars, except that they use sound waves. They are mainly used in water as the long wavelength of the sound waves and the characteristics of water itself allow the signal to travel to extremely far distances without attenuation.

Lidar (Light Detecting and Ranging), also known as Ladar (Laser Detecting and Ranging), uses focused light beams as its signal. The short wavelength of the light makes the resolution finer, while limiting the signal range and making it vulnerable to fog, smoke and dust.

With any of these signals, there are three main methods to estimate distance based on the characteristics of the returning reflection: Time of Flight, Phase Shift and Triangulation, which we discuss independently in the following sections.

2.1.2 Time of Flight

Knowing the speed of the emitted signal, we can measure the distance of the object that reflected the signal based on the time it took for the signal to travel back. It is widely applied in many areas such as autonomous navigation, meteorology and geology. One of the most notable implementations of Lidar is the Velodyne HDL-64E camera widely used by scholars, such as in Kitti [24] and [45]. It uses an array of 64 laser emitters to provide a 3D scan using time of flight. The array rotates at high speeds providing 360° degree horizontal and 26.8° vertical scans of the surroundings at 5-15Hz [1].

2.1.3 Phase Shift

The shift in the phase of the emitted signal can indicate the distance of the object if the wavelength and the frequency of the signal are known. In cameras such as SICK LMS 200 that use this method, an array of infrared LEDs is used to illuminate the scene with different modulations. The drawbacks of Phase Shift calculation is the short range of the infrared beam (typically less than 10 meters). The other limiting factor for the range coverage of this method is the fact that for a signal with a wavelength of w, a reading of range x is not identifiable from reading ranges $w + x$, $2w + x$ and so on and the phase shift for all of these distances is the same [37].

2.1.4 Triangulation

In triangulation, the angle at which the reflected laser beam enters the sensor is measured. Having the measurement of this angle, the distance of the emitter to the sensor and the size of the sensor, distance of the object to the emitter can be calculated (Figure 2.2). Similar to the phase shift, the main drawback of triangulation based range sensing is its short maximum range for light beams [37].

2.1.5 Stereoscopic Vision

One of the techniques used by the human visual system to perceive depth of objects is Stereopsis. Since our eyes are positioned at slightly different places on our head, they provide the brain with two slightly different views of the scene we look at.

The amount of displacement of a particular element between the images of the two eyes varies depending on the distance of the object. This helps the brain perceive

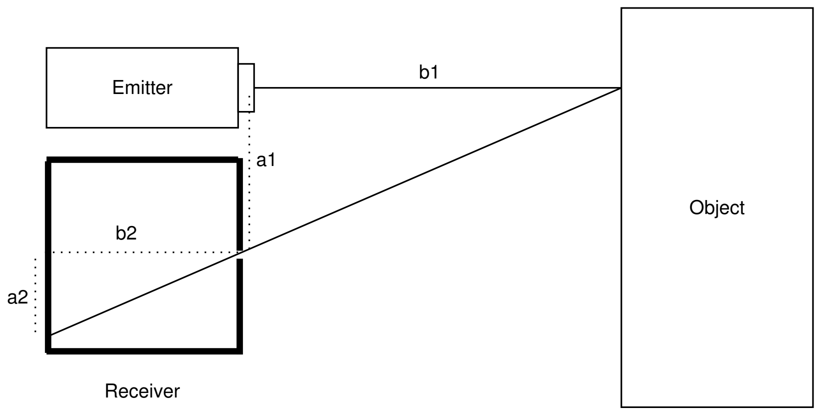


Figure 2.1: Triangulation: when the emitter and receiver are parallel, the triangular proportions can be used to determine $b1$, by knowing $a1$, $a2$ and $b2$. In a non-parallel setup, knowing the angle between emitter and receiver is required as well.

the element's depth. Objects that are closer to the viewer have greater position displacement, while further objects have less of a displacement.

Stereo Vision techniques use the same approach using RGB cameras. Two digital cameras can be used to capture two slightly different images if they are positioned close together. Corresponding pixels or patches between the two images can be identified using the colour characteristics of the pixel and its neighbours, and the displacement in the projected images can be calculated in terms of pixels. Having the configuration of the camera at the time of image capture such as focal length which is also available as header information in compressed image formats, it is possible to estimate the distance of each pixel/patch based on its displacement.

Stereoscopic vision has been widely discussed in literature and also commercialized in devices such as Sony HDR-TD10 for producing 3D videos. There are many different stereoscopic depth estimation algorithms, many of which are compared by Scharstein and Szeliski [58], Sunyoto et al. [61] and in the Kitti Vision Benchmark Suite [24].

2.1.6 Structured Light

In this method, beams of light from a light source are projected onto the scene in a structured manner (e.g. stripes or a matrix of dots) and a sensor would capture how the structure is distorted when it is reflected back from the objects in the scene. An overly simplified example would be the flashlight: its projection would appear bigger on further objects and smaller on closer objects. This method is used solely such as in [67] and [54], or in conjunction with other methods such as in Microsoft Kinect and [8].

One drawback of this method is the choice of pattern which dictates the resolution of the sensing. In case of stripes, only the distance of points at which colours change can be measured. Another approach would be to use a coloured gradient so that every pixel can be measured, but this only works for scenes with minimal texture and changes in colour. Another problem with structured light methods is the interference between multiple light sources, which can be mitigated in some cases [10]. Different methods of capturing and analyzing in structured light are discussed in [23].

2.2 Point Clouds

A point cloud is a collection of points with 3D coordinates $p(x, y, z)$ that represents the point's position. Potentially, the point might have many other properties, including but not limited to: colour information (RGB), opacity or alpha (A), normal vector at that particular point ($N(x, y, z)$), etc.

Until so far, we have only discussed how the Z-value (depth) is measured using the depth cameras. Having a depth frame, the values for x and y can be calculated assuming that the Z value and the camera's field of view is known.

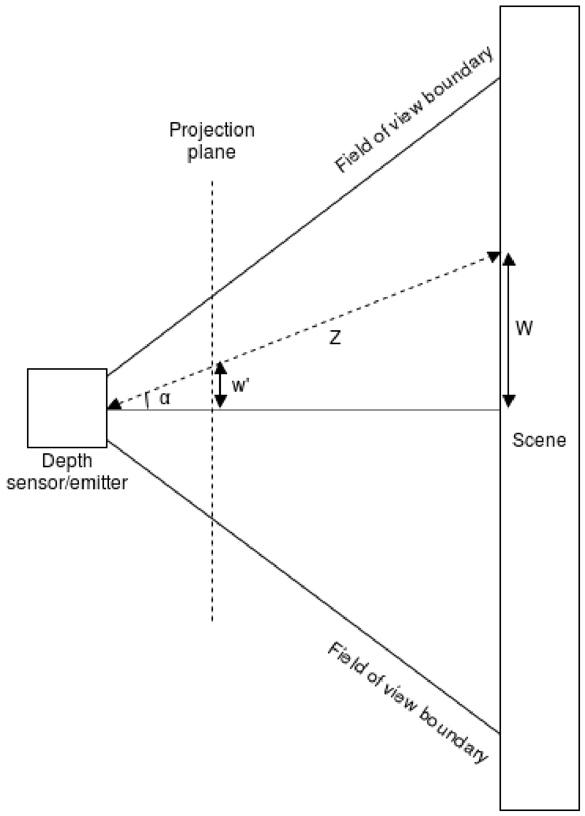
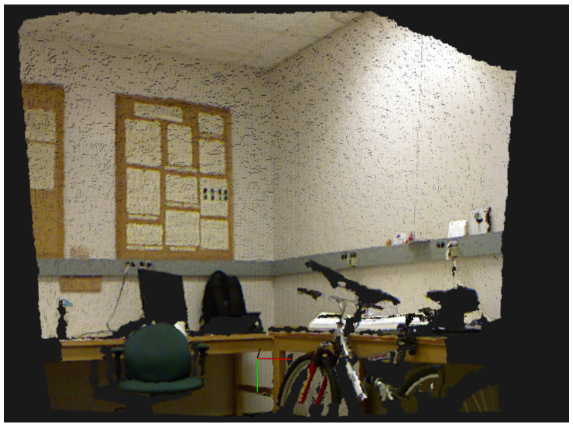


Figure 2.2: Top-down view of a sample scene in triangulation: the real-world coordinate of a pixel, such as the distance of the point to the camera in the horizontal place (W), can be determined by having the pixel depth value (Z) and the α angle. α is determined by knowing the field of view of the camera and the position of the pixel in the RGB frame (w').

To determine the colour information of a point, it should be noted that in depth cameras with RGB support (RGBD cameras), the depth sensor and the RGB camera are separate pieces, hence they are located at slightly different positions of the unit. This causes a slight misalignment between the depth frame and the RGB frame.


(a) View from the camera's point of view.

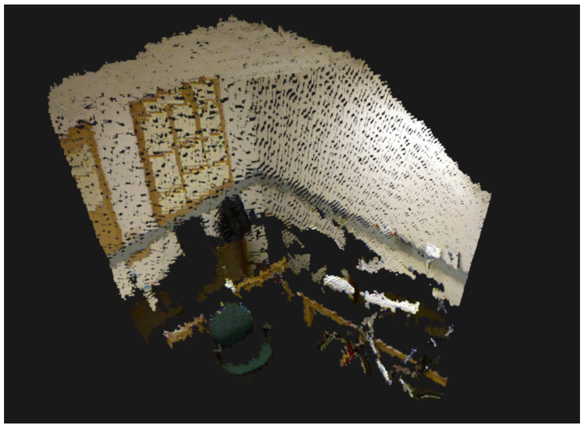

(b) View from above the camera.

Figure 2.3: Visualization of a point cloud from different angles.

Consequently, some pixels at the edges of the depth image lack colour information, and some RGB pixels lack depth. In order to be able to combine the depth frame and the RGB frame information into a point cloud, a 2D registration is required unless the exact misalignment of the cameras is known. In commercial devices such as Microsoft Kinect, the device driver automatically aligns the RGB and depth frames.

A point cloud can be easily visualized using any graphic rendering engine such as OpenGL or using more sophisticated tools such as game engines. Figure 2.3 shows the visualization of a single point cloud captured in our lab from different angles.

There are a number of algorithms and applications that are of interest in point cloud processing which is explained here briefly. Registration is described independently in the next section.

2.2.1 Surface Normal Estimation

Assuming that every point in the cloud is part of a surface, estimating the normal vectors of that surface for every given point is a basic yet important calculation that is mainly required for other advanced algorithms. There are many methods to estimate the normal vectors of a surface.

One of the simplest ones is by fitting a plane for the nearest neighbours of each point. By having the plane, the direction of the normal is known but there are two possibilities for the orientation of the normal. Knowing the fact that the front of a surface on a point cloud captured from a depth camera cannot be on the side which is opposite to the camera, the orientation can also be estimated by knowing the camera view point. Other methods may rely on surface reconstruction or the point cloud mesh. A more detailed discussion on normal estimation in noisy point clouds is found in [16].

2.2.2 Surface reconstruction and volume rendering

As mentioned earlier, each pixel in the depth map and RGB image which make up the point cloud represents only one point in the real world. Hence, the point cloud derived directly from a depth camera is essentially a collection of sparse points with gaps in between points. The size of the gap depends on the resolution of the camera and the distance of points to the camera. Figure 2.4 shows the same point cloud in figure 2.5a when zoomed in.

To improve the quality of the visualized point cloud, or to extract a more complete model for further processing, surface reconstruction techniques are employed to derive

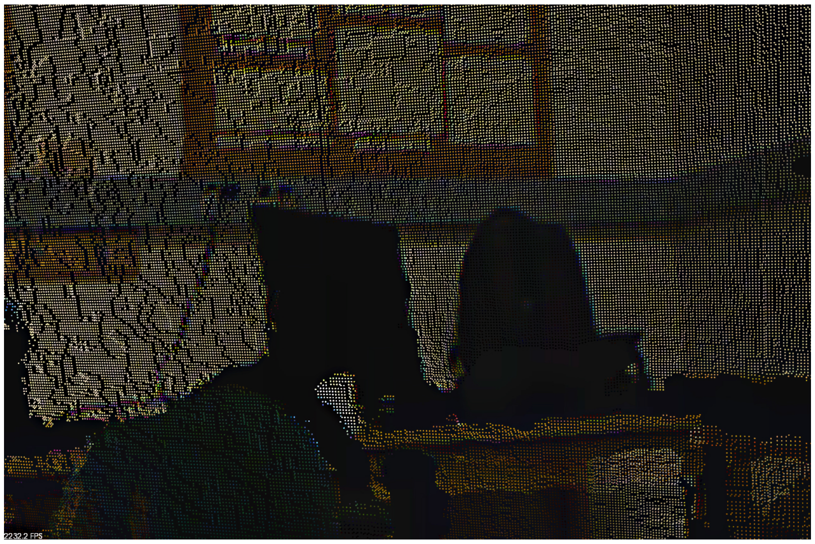


Figure 2.4: Visualization of a point cloud. Raw point clouds from depth sensing cameras are often a sparse collection of points with visible gaps in between.

a mesh from the point cloud. Volume reconstruction is used to fill the point cloud. One of the examples of surface reconstruction algorithms for visualization is surface splatting [73] which works based on the fact that the colour of every pixel in the final rendered image is a blend of the points that fall into or close to that pixel from a certain camera view. The colour of a point is determined by the colour and density of its neighbouring points.

Another method by Marton et al. [39] generates meshes using triangulation: it determines the points that belong to the same surface based on their position and connects them to form triangles which make up the mesh. The mesh can be either rendered or used for other applications.

2.2.3 Down Sampling (sub-sampling)

3D data processing can be a resource intensive task when the algorithms perform in orders higher than $O(N)$ (N being the number of points in the cloud). For example for surface reconstruction, processing one point requires processing its neighbouring points. In the ICP registration algorithm, processing each point from one cloud requires finding its corresponding pair in the other cloud, which requires $O(N^2)$ if not optimized. The resolution of off-the-shelf depth cameras is often high enough to hinder such algorithms considerably. For example the Kinect camera provides a 640×480 resolution (307,200 points) in real time. This number of points can make registration algorithms such as ICP take minutes per frame.

Furthermore, not all points in the cloud can further contribute to some algorithms. Many of the points in a cloud might be redundant due to the high density especially on surfaces that are very close to the camera. In such scenarios, down sampling is used to reduce the number of points in the cloud while preserving the model it represents.

The simplest approach for down sampling is filtering points randomly with equal chances from the point cloud. This is an extremely fast and efficient approach, but it may not be the best solution in scenarios when objects close to the camera contribute a large number of points, while objects far from the camera only make up a small portion of the point cloud. Choosing points randomly in this scenario will decrease the already limited quality of the point cloud for surfaces located farther away.

A simple approach to solve this problem introduced by Magnusson [37] is dividing the point cloud by uniform cells, randomly selecting one point from a random cell and repeating until a certain criteria is met (e.g. based the total number of selected points or ratio). This will assure that different areas of the point cloud have equal chance to be included in the sub-sampled point cloud since the number of points in an area does not increase the chance of points being selected in that area.

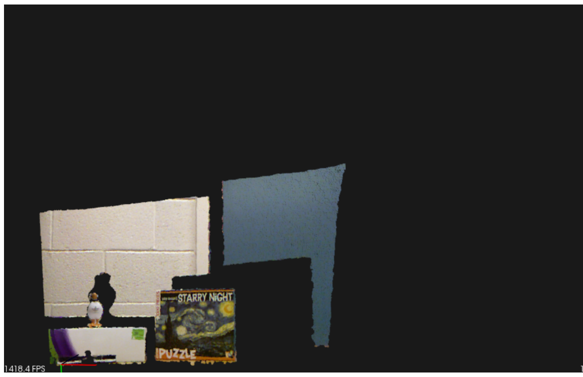
More intuitive down sampling approaches may be applied for other scenarios. For example, we may choose the sub-sampled points in a way that we favour preserving a wider range of normal vectors. We may also select more points from parts where changes in the normal vectors are detected, so that we preserve scene features such as edges.

2.3 Registration

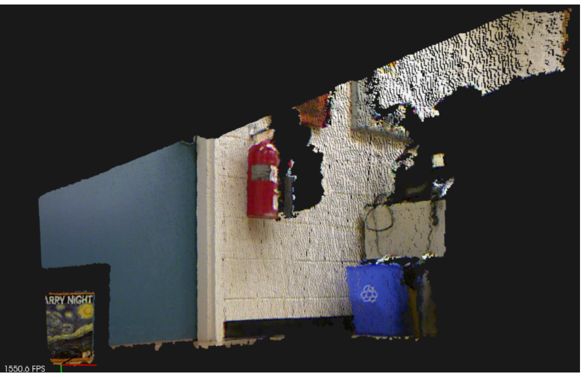
Each point cloud has its own independent coordinate system. For example, a point cloud captured from a camera might use a Cartesian coordinate system with the origin $O(0, 0, 0)$ at the camera position, using a right-handed approach with the Z-axis pointing forward from the camera and X-axis point downward.

When capturing one scene from two different positions, the coordinates of the generated point clouds differ. For example in image 1, object A is closer to the camera and its points have smaller Z values; while in image 2, the same object is further from the camera due to camera movement and the same points have larger Z values and in general, different coordinate values. As a result, if we visualize two point clouds captured at different positions from the same scene, the results would be two misaligned point clouds (figure 2.5c).

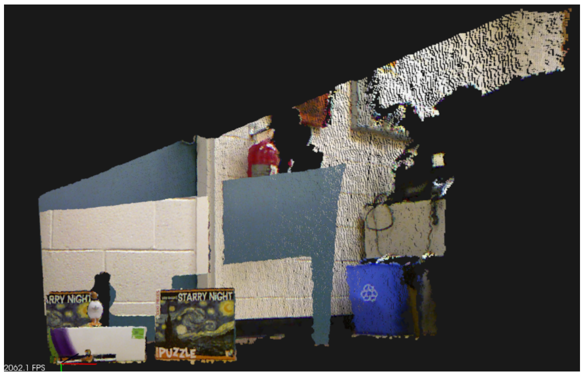
Assuming that we have two point clouds from the same scene (P_A and P_B), if the point clouds have some overlapping regions, then some of the points from P_A



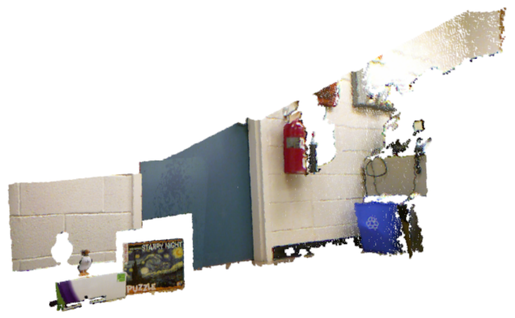
(a) First point cloud



(b) Second point cloud



(c) Visualization of the accumulation of both point clouds without registration



(d) Visualization of the accumulation of both point clouds after registration

Figure 2.5: Visualization of two point clouds captured from different points of view will generate misalignment.

represent the same areas that some of the points from P_B represent. This is known as correspondence. Two points (one from P_A and one from P_B) that represent the same area of the real world are referred to as corresponding pair. The term corresponding pair is not limited to point clouds and is used in 2D images as well.

Registration focuses on estimating a transformation that transforms one image to the other in a way that corresponding pairs would have the same or similar coordinates, so that the visualization of both inputs looks aligned (figure 2.5d) as the end-user would expect. The point cloud that is transformed is referred to as the source, and the point cloud that is used as the goal of transformation is referred to as target.

It should be noted that corresponding pairs are not strictly "corresponding", meaning that even after a successful registration, their coordinates will not be exactly the same. This is for two reasons: firstly, the depth estimation obtained from the device is not 100% accurate, and hence the values calculated for the coordinate of a point, which are all derived from the Z, are only accurate to some extent. Secondly, even if we assume that our camera is ideal, the areas that a pixel from an image covers is typically much larger than the area it represents in the real world, more so for far objects. Hence it is unusual for a corresponding pair to represent the exact same point in the world. The limitation of imaging resolution is tightly coupled with this limitation.

For 2D registration, the transformation can be represented by a 2D translation in the direction of (x, y) axes and a rotation. In 3D registration, there are 6 degrees of freedom: 3 directions for translation in (X, Y, Z) axes and 3 directions of rotation around each of the axes (also referred to as roll, yaw and pitch).

There are multiple ways to represent a 3D transformation. Translation is often represented by a vector of 3 elements. Rotation can be defined by Euler angles, which is a vector of 3 elements each describing the rotation around each axes.

$$\begin{aligned} Translation &= [t_x, t_y, t_z] \\ Rotation &= [r_x, r_y, r_z] \end{aligned} \tag{2.1}$$

It can also be represented in a 3D rotation matrix. Rotation matrices have useful properties such as: 1) multiplying two rotation matrices results in a rotation matrix that represents the same rotation from applying the two rotations sequentially. 2) Transpose of the rotation matrix is equal to the inverse rotation. The rotation matrix can be determined by the $[r_x, r_y, r_z]$ values shown below. c and s in equation 2.2 respectively represent *cosine* and *sine* functions.

$$Rotation = \begin{bmatrix} c(r_y)c(r_z) & c(r_y)s(r_z) & -s(r_y) \\ s(r_x)s(r_y)c(r_z) - c(r_x)s(r_z) & s(r_x)s(r_y)s(r_z) + c(r_x)c(r_z) & s(r_x)c(r_y) \\ c(r_x)s(r_y)c(r_z) + s(r_x)s(r_z) & c(r_x)s(r_y)s(r_z) - s(r_x)c(r_z) & c(r_x)c(r_y) \end{bmatrix} \tag{2.2}$$

Another representation of rotations is the Quaternion which is often used in computer graphics since they use less memory, compose faster, and are naturally suited for efficient interpolation of rotations [29].

A 3D transformation can be represented as a $4x4$ 3D transformation matrix such as below with the top left $3x3$ portion describing the rotation and the column on the right describing the translation.

$$Rotation = \begin{bmatrix} c(r_y)c(r_z) & c(r_y)s(r_z) & -s(r_y) & t_x \\ s(r_x)s(r_y)c(r_z) - c(r_x)s(r_z) & s(r_x)s(r_y)s(r_z) + c(r_x)c(r_z) & s(r_x)c(r_y) & t_y \\ c(r_x)s(r_y)c(r_z) + s(r_x)s(r_z) & c(r_x)s(r_y)s(r_z) - s(r_x)c(r_z) & c(r_x)c(r_y) & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

An in depth explanation of existing registration algorithms is provided in Chapter 3.

2.4 Applications of Registration

Most application of 3D registration can be generally categorized as: 1) Localization and 2) 3D Model Generation.

2.4.1 Localization

Recent commercialized depth cameras are widely used with robots due to their compact size, energy efficiency and low costs. The depth camera can offer a number of valuable information to the robot such as distance to obstacle, scene construction, presence of human or known objects, etc. One of the major fields of research in robotics is robot localization which aims to determine where the robot is located in the real world or in relevance to its starting position; also referred to as SLAM systems (Simultaneous Localization And Mapping). Use of 3D data for SLAM is important in scenarios where other methods such as GPS or odometry are not possible or accurate enough.

Registration algorithms can help with determining the movement of the robot. The transformation between two 3D images is the same physical transformation of the camera(s) that captured the images. If the depth camera which in mounted on a robot captures two frames while moving, the transformation between the two frames indicates how exactly the robot has moved. This helps indicate the position of the robot relevant to its starting point if frames are taken frequently and the transformation between the last two frames is accumulated with the previously calculated transformations. Outdoor SLAM [13], underwater SLAM [55] and underground localization [37] are examples of applications of registration in localization of autonomous vehicles.

2.4.2 3D Model Generation

3D model generation or 3D scanning is the process of modeling real world elements into 3D models. The use of the generated 3D model varies based on application. In this research, we are focused on modeling characters, environments and objects for the purpose of visualizing them in a virtual reality collaborative environment.

Since a single frame of 3D scan is not enough for most applications of 3D modeling due to occlusions and limited field of view, the depth camera is usually moved around the scene to capture the elements from all necessary points of view. In such cases ([31], [19], [63]), depth cameras can act as hand-held 3D scanners which can be used to scan objects, indoor environments and the human body. In some applications, the camera is fixed and the objects are placed on a moving platform to be scanned from different views.

In such scenarios, the different frames that are generated for the purpose of modeling need to be merged together to form a complete model. This requires estimating the transformation between frame pairs. Registration algorithms are designed to estimate this transformation by analyzing the input image pairs.

2.5 RGB Feature Detection and Matching

In image processing and computer vision, feature detection refers to algorithms that identify “interesting” pixels in an image. The exact definition of “interesting” varies among different methods, but in general, features refer to points or set of points that have noticeable difference in their properties (colour, intensity, etc.) from their neighbouring points. Feature detection algorithms can be divided into 3 categories based on the type of feature they extract:

2.5.1 Corners

The term corner is misleading since corner refers to not only corners, but any point in the image that stands out in its local region (e.g. a black spot on a white background). Corners are also referred to as interest points. SURF (Speeded Up Robust Features) by Bay et al. [3], SIFT (Scale Invariant Features) by Lowe [36] and ORB (Oriented BRIEF) by Rublee et al. [51] are some of the more popular works in this area.

2.5.2 Edges

These features refer to pixels/lines of an image that represent the connection of two or multiple regions (e.g. when pixels belonging to an element meet pixels of another element in the image). While simpler algorithms only identify pixels that are detected as edges, some algorithms detect clusters of pixels with each cluster representing the different parts of a single edge. Canny detector [11], Hough lines detector [42] and [62] are some of the well-known works in edge and line detection.

2.5.3 Blobs/Regions

Rather than points or lines, blobs refer to a set of points or regions of an interest in an image. MSER (Maximally Stable Extremal Regions) [41] and PCBR (Principal Curvature-Based Region detector) [15] are examples of region detection algorithms.

2.5.4 Feature Descriptors

An important part of the feature detection algorithms are feature descriptors. The extracted features are described by descriptors which can then be used for further processing. Descriptors are data structures consisting of measurements based on the observed properties, often between the interest point and its neighbours.

Feature detection algorithms are often the first step in the pipeline of computer vision applications. In some scenarios, a comparison of an image pair is required which can be achieved by matching the feature descriptors of the images. This is known as feature matching. Feature matching is used to identify similar elements in two images to detect known shapes or objects or to match images.

For this to work, the descriptors of the feature detector need to provide similar results for the same elements in different images. This defines repeatability as one of the most important properties of good feature detectors: the ability to yield the same results under different conditions such as different viewing angle or lighting.

Especially for the purpose of registration, invariant feature detectors are of great importance. With invariant feature detector, the results of the feature detection (descriptors) do not change if a transformation is applied on the input images. For example, in scale-invariant feature detectors the scale of the image does not affect the results: the descriptors of the features are not affected (invariant), although the positions of the features are affected according to the transformation. With rotation-invariant feature detectors, rotation of the camera or the image does not affect the descriptors. In registration the image pairs are taken from different viewing angles and possibly with camera rotation, thus, both scale and rotation invariance is required.

The effectiveness of the descriptors is an important factor in the effectiveness of the feature matching algorithms. The size of the descriptors affect the speed of feature matching, while an overly small sized descriptor might not describe the feature in a distinct way.

Brute force is the most straightforward method for feature matching. Each descriptor from one image can be compared to all descriptors from the other image and the closest descriptor is selected, however this yields $O(N^2)$ complexity and slows execution time. FLANN-based matchers use FLANN nearest neighbour search (Fast Library for Approximate Nearest Neighbour search) [46] for finding the closest descriptors to the trained descriptor set. This method might be faster than brute force with large number of descriptors.

For a comprehensive review of different feature detection algorithms, refer to "Local Invariant Feature Detectors: A Survey" by Tuytelaars and Mikolajczyk [65]. A detailed performance comparison of different feature detection and description methods for the purpose of feature matching is presented in [44].

Chapter 3

Related Work On Registration

A significant amount of research has been dedicated to the registration problem in computer graphics, vision and robotics community. Since we are interested in an automated solution for the virtual reality collaboration environment, in the remainder of this thesis and in this review of related work on registration we only focus on registration algorithms and techniques that are not assisted by human input.

There are three main challenges regarding existing registration algorithms that are particular to our application domain and prevents us from using them in our solution:

1. Initial Pose: initial pose refers to how the point clouds are positioned relative to each other. In most scenarios, the initial pose is the default pose generated by the cameras without any transformations applied to it. Many of the algorithms start from the initial pose and try to minimize or maximize a certain metric iteratively until they reach the ideal pose.

   Over the solution space which consists of every possible transformation, the ideal transformation is considered a global optimum which minimizes an error metric or maximizes a score better than any other transformation. The minimizing/maximizing function and metric implemented in many algorithms searches for the global optimum locally around the initial pose. Hence, such algorithms are prone to local optimums (see figure 3.1).

In our proposed application scenario, multiple cameras are used simultaneously to capture the scene and users from different angles to provide an almost complete model of the scene to other users who are virtually present in the environment. To prevent using numerous cameras with minimal initial pose, only a handful of cameras are used with noticeable angle differences and distances. Hence, our solution requires to successfully register images that have large initial pose distances.

2. Speed: many of the current registration techniques are not capable of processing the large input from depth cameras directly in real-time (refer to Chapter 5 for experiment results). This is due to the fact that depth cameras such as Microsoft Kinect provide 640x480 frames which make up 307,200 points in each frame; hindering the algorithms that perform in orders of bigger than $O(n)$. To speed this up, down sampling (discussed in 2.2.3) is applied to the point cloud prior to passing it to the registration algorithms. A rigorous down sampling will degrade the quality of the point cloud which might cause errors in estimating the transformation, while a less restricted down sampling does not allow the algorithms to perform in real time. Moreover, it should be noted that the down sampling process itself consumes time.

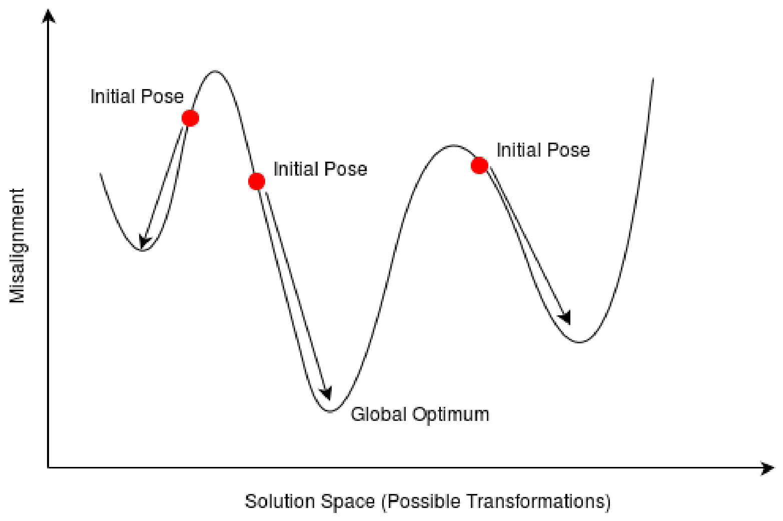


Figure 3.1: A simplified view of the initial pose problem. Red circles represent initial pose and arrows indicate the algorithms' solutions. Score maximizing or error minimizing functions in many registration algorithms is prone to local optimums when the initial pose is far from the optimal solution.

3. Extent of Overlap: we define overlap as the portion of the two input point clouds that represent the same elements in the scene. In our application, the non-overlapping portions are a result of different placements of the capturing devices when taking the image pairs. Many algorithms only perform well when the input point clouds share most of the scene and no major portion of the scene belongs to only one of the 3D images. These algorithms are designed to either minimize an error metric or maximize a score metric, both of which describe how good a proposed transformation is. However, if the metric does not work well under different conditions (including with small amounts of overlap), then the measurements based on that metric under those conditions will not represent the correctness of the proposed transformations, making the algorithm discard the correct solution.

   The problem with non-overlapping sections happens with many of the distance based metrics. As an example, Sum of Absolute Distances (SAD) can be calculated between the points of one cloud to the closest points of the other cloud, and the algorithm may aim to minimize this distance to achieve the solution transformation. Assuming two point clouds which capture almost the same scene from different angles with minimal non-overlapping portions, this metric generates lower errors when the clouds are pushed together in a way that overlapping parts are aligned. On the other hand, in two point clouds that have minimal overlapping, most parts of both clouds will not be aligned with any point. In this case, a correct transformation between the clouds places them almost apart from each other. In this case, measuring the sum of absolute distances between these clouds produces a large result, while the error metric produces better results when the clouds are incorrectly pushed together and their points are by average closer to each other. The algorithm that uses such metrics favours the incorrect transformation since it minimizes the error metric. In figure 3.2b, we can see an example of how such algorithms push clouds together to minimize the error metric.

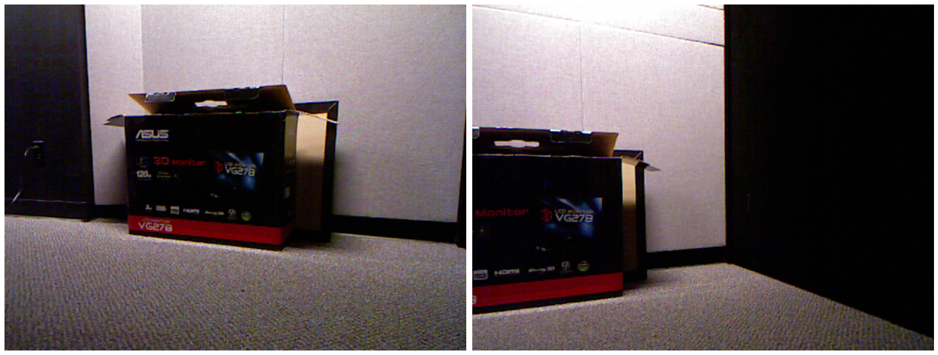
In the following sections we will have a close look at some of the well-established registration algorithms:

3.1 ICP

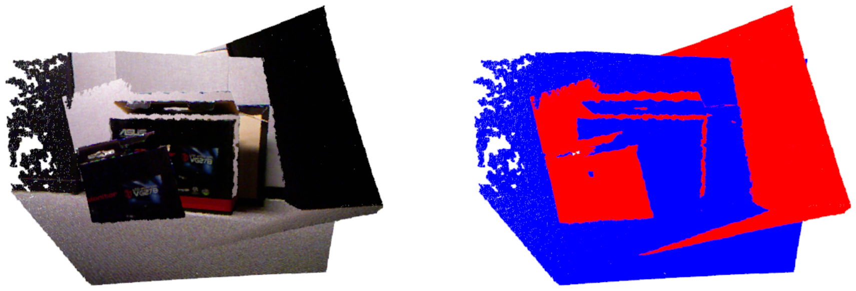
The ICP (Iterative Closest Point) algorithm was originally introduced by Chen and Medioni [12] and Besl and McKay [5], which is easy to understand and implement. In its basic form, ICP aims to minimize the sum of squared distances between corresponding point pairs in the two point clouds. ICP consists of two core steps:

1. Corresponding Pair Detection and Rejection: The ICP algorithm starts from an initial pose. For every point in the source point cloud, the corresponding pair is detected as the closest point from the target cloud based on Euclidean distance. This step is the most computationally complex part of the algorithm that consumes most time. Nearest Neighbour Search methods such as Kdtree [4] can be applied to speed up the process.

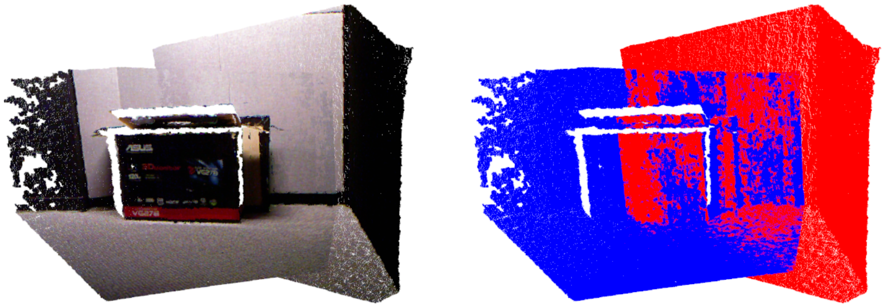
   A large number of false corresponding pairs can greatly affect the result of the registration, hence it is important to reject pairs that might not be representa-



(a) The views of the input point cloud pair



(b) Registration by minimizing distance based metric using ICP [12][5]



(c) Registration using DeReEs

Figure 3.2: Registration algorithms that minimize distance based metrics between all points are prone to point clouds with noticeable non-overlapping sections.

tive of the same elements. A simple rejection approach widely used is to reject any pairs that have a distance higher than a certain threshold. Note that with this correspondence detection and rejection approach, we are assuming that points that have a large distance from each other have a lower possibility of being corresponding pairs, compared to points that are placed closest to each other. This introduces the core dependency of the ICP algorithm to a good initial pose as explained earlier.

2. Transformation: After corresponding pair detection and rejection, a rigid transformation can be easily estimated in a way that best minimizes the sum of squared distances between the remaining corresponding pairs. Finding such transformation is explained in the paper by Besl and McKay [5].

These two steps are performed iteratively so that the point clouds converge to the solution pose, until a certain criteria is met. The criteria is usually set as reaching a maximum number of iterations, or when the most recent proposed transformation has an insignificant effect which is indicated by a threshold.

The ICP algorithm is well suited for registration of 3D point clouds with minimal transformation distances or good initial poses. For example in scenarios where the depth camera is capturing a scene with high frame-rates while the camera is moving slowly (such as in hand-held 3D scanners), the transformation between consecutive frames is minimal.

The performance of the ICP algorithm with large point clouds is extremely slow, but after applying down sampling, real-time performance can be easily achieved with 5-10k points per cloud which is often enough for fine alignment, as shown in our experiments in Chapter 5.

The two main problems with the ICP algorithm, in the context of our application, are the initial pose and partially overlapping clouds. As mentioned earlier, the ICP algorithms assumes that point pairs which are closest to each other are true corresponding pairs, or a closest pair would guide the points in the source cloud to their true corresponding pairs in the target cloud iteratively. This assumption does not hold true in many real world situations.

The metric used for ICP is the sum of squared distances which is prone to local optimums existing in between the initial pose and the ideal pose. In the case of point clouds with minimal overlapping, the ICP algorithm fails to register clouds. Even with a good initial pose, the ICP algorithm finds corresponding pairs for points that basically do not have any corresponding points in the other cloud. Based on this false correspondence, the points on parts of the source cloud that are exclusive to the source cloud are pushed to be placed near the points of the target cloud, to minimize the sum of squared distance error metric.

Setting a threshold for rejecting far corresponding pairs can initially help to reject exclusive points of the clouds in the pair matching process, but after a number of iterations, those points will slowly converge toward the other cloud and will be included in the process again. Figure 3.2 illustrates two point clouds of a real scene used in our experiments with a correct transformation found by our algorithm, and the transformation proposed by ICP. Notice that the error metric that ICP uses would naturally report a more favourable error measurement for the incorrect ICP solution, compared to the correct solution which produces very large squared distances.

While these seem like extreme examples with the overlapping and the initial pose problem, based on our experiments it happens quite often in indoor environments. Point clouds of indoor scenes include large surfaces such as walls, floors and ceilings. The problem with such flat surfaces is that from the ICP's perspective, flat surfaces are identical. In other words, the error measurement generated from transforming surface A incorrectly over surface B is a small error since both surface A and B are flat and match together easily. At the same time, note that surfaces such as walls contribute most of the points that exist in the cloud. The contribution of this majority of points to the error metric overcomes the contribution of the smaller portion of points which represents smaller features and objects in the scene. This leads to the existence of numerous local optimums in indoor scenarios, which the ICP algorithm falls into in case of a non-ideal initial pose.

To improve the ICP algorithm, numerous extensions and variants have been introduced, which mostly aim at improving individual steps of the algorithm. We will overview some of these alternatives in the following section.

3.2 ICP Variants

Numerous variants and extensions to the ICP algorithm have been proposed, some of which are surveyed in [52] as well as in [37]. These variants are classified into 5 categories based on the targeted component of the ICP algorithm that they aim to improve. We briefly discuss proposed variants for each category:

3.2.1 Selection

This component of the ICP algorithm selects the points from the point cloud that are passed to the ICP algorithm. Originally in the work by Besl [5], all points of the point cloud are passed to the algorithm, regardless of their distance. In a work by [64], to prevent non-overlapping parts of the clouds to cause misalignment two criterion are applied to the points that contribute to the proposed transformation and error metric at each iteration: 1) points that are too far apart and 2) points that are positioned on the boundaries of the mesh are not included in the calculations.

Both [64] and [40] suggest an iterative approach to ICP itself: performing the ICP in iterations with different amounts of input points and details to fine tune the registration over time. By starting from small amounts of input points which represent the overall structure of the scene, the algorithm can quickly estimate transformations close to the final solution, after which, a larger number of input points which represent details of the scene, helps the algorithm to estimate a better alignment.

With colour or intensity data, another option would be to select points where the colour or intensity values change drastically to include corners and edges into the selection [68]. Likewise, with point clouds that include normal vectors for the points, or after normal vector estimation for the point cloud, points can be selected in a way that preserves the distribution of normal vectors [52] or in a way that points with drastic normal vector changes are preferred to preserve features in the scene.

3.2.2 Matching

In the original work by Besl [5], the correspondence detection between the points works by selecting the closest point in the other point cloud as the corresponding pair. Chen and Medioni [12] suggest normal projection or normal shooting: for a source point, the corresponding pair is the point in the other cloud that intersects with the source point's normal vector. Another approach would be projecting the source point into the destination cloud based on the destination cloud's camera point of view [7][47]. With any of these methods, assessment of the compatibility of the source point and the proposed target point have been explored using the colour and normal vector information [26][50].

The algorithms that use a variant of projection are prone to very large transformation distances where the projection might not produce an accurate result or any result at all. On the other hand, projection methods are less prone to measurement errors since they do not work based on distance, hence they converge to the solution faster [52].

3.2.3 Weighing

One approach to decrease the effect of false corresponding pairs on the transformation estimation is to weigh matches based on our assumed "quality" of the match, and to take the weighing into account for transformation estimation. The original ICP proposed by Besl can be considered as a weighted matching with constant weighs for all matches. Godin [26] proposes lower weighs for pairs with larger distances. He also proposed using the colour distance for weighing and weighing based on the difference of normal vectors.

3.2.4 Correspondence Rejection

Similar to the selection and weighing component of the ICP algorithm, correspondence rejection tries to minimize the false correspondences that cause errors in the transformation estimation. As mentioned earlier, pair rejection based on a distance threshold and their placement over the boundaries are two adopted methods. Another method is to reject a percentage of pairs based on the metric used for correspondence such as the distance [50] or to reject pairs that are not close to the standard deviation [40].

A more sophisticated but time consuming approach is to reject pairs that do not have the same consistency with their neighbours [18]. In other words, if the placement of source point relative to its neighbours is different than the placement of its target point with its neighbours, the chances are that these two points represents different parts of the scene or errors in depth measurement exists.

3.2.5 Error Metric Minimization

Originally, the sum of squared distances was proposed as the error metric for the ICP algorithm. Colour information can also contribute to the error metric. Chen [12] suggests a point to plane error metric: sum of squared distances from source points to the surface of the target point. Other than the metric itself, there are also different mathematical approaches to formulate the process of minimizing the error metric. Some of these approaches are surveyed in [20].

3.3 3D-NDT

NDT (Normal Distributions Transform) was first used by Biber and Strasser [6] for 2D registration and was extended to 3D registration by Magnusson[37]. In his work, Magnusson explains the shortcomings of using point clouds: 1) lack of surface characteristics such as orientation or smoothness, 2) when extracted from sensors, point clouds have unnecessarily large number of points on surfaces close to the sensor and much less information for further surfaces.

Instead, Magnusson suggests a surface representation approach: instead of using a point cloud, the model is transformed into a smooth surface representation. This representation consists of a set of local Probability Density Functions (PDFs) which describe different sections of the surface. This is done by dividing the model into a grid of cells (cube for 3D, square for 2D) and computing the Probability Density Function for each cell. This function describes the likelihood of the existence of a point in a certain location of that cell. The PDF function is a piecewise smooth representation of the surface. Hence, it can be used to extract its derivatives. The derivatives help derive other data such as the orientation and general smoothness of the surface.

After converting the target cloud into the surface representation, the registration works based on finding a transformation that maximizes the PDF function results, with the points of the transformed source cloud as input. In other words, the registration aims at maximizing the probability that the source point exists at the location where it is transformed into, based on the probability functions produced by the target cloud.

The PDF function uses normal distribution; the output of the function is determined by all points in the cell. Consequently, it will blur features that are small compared to the cell. Overly large cell sizes will cause the algorithm to neglect details of the model. In other words, large cells lead to less detailed registration. Furthermore, a cell only contributes to the registration of points within itself. Overly small cell sizes will cause the algorithm to fail if the initial pose of the models are not close to the solution. Hence, choosing the right cell size is a challenge in 3D-NDT.

Magnusson explains many extensions to the 3D-NDT algorithm that aim to cover its limitations, especially with cell sizes. Three of these extensions are described below:

- Iterative: a straight-forward option would be to perform 3D-NDT with different cell sizes, starting from large sizes then small ones. The initial large cell sizes allows the algorithm to converge to a good initial pose, with successive small cells fine-tuning the transformation.

- Octrees: The structure of the cells is created using Octrees [33]. The algorithm starts by fixed cells, with each cell being the root of an Octree data structure. Each node of an Octree contains 8 children. The space covered by the root node is split between the 8 children. Then the same is applied to the children nodes recursively, until the leaf nodes contain points less than a certain threshold. The search for transformation can then be executed from the root level to the leafs in an iterative approach.

- Clustering: using a clustering approach such as k-means, the point cloud can be divided into clusters of points. Each cluster refers to one cell, which causes the cell size to be adaptive to the cluster size.

Based on [37], 3D-NDT performs better than ICP in terms of accuracy and with large transformation distance for underground mining environments (the motivation and application behind 3D-NDT), but it is not suited for real-time applications, as registration of each frame pair requires more than 1 second to complete.

3.4 Feature-Based Registration

Feature-based registration on 2D images can be used in most of the commercial depth-sensing cameras as they incorporate a RGB sensor and provide RGB frames of the scene as well as the depth frames. Feature-based registration works on the basis that RGB feature detection and matching can be used to find corresponding points in the two images, which then can be translated to the points in the 3D point clouds. Different approaches can be then implemented to use this correspondence information for estimating a transformation. Scale Invariant Feature Transform (SIFT [36]) or Speeded Up Robust Features (SURF [3]) algorithms are used with the majority of feature-based registration techniques.

There are some existing challenges with feature-based registration methods:

1. False Correspondence: In the ICP algorithm each point pair in the clouds can be used to create a correspondence. With clouds of about 307200 points, the number of corresponding pairs is very large. Compared to the ICP, the correspondence accuracy is much higher with feature-based methods, but the number of corresponding pairs is much less. Hence, a small number of false correspond-

ing pairs can easily affect the whole set and create significant errors in the transformation.

2. Dynamic or Repetitive Scene Structure: The majority of the features are detected around colour variations such as edges or colourful objects. In dynamic scenes where an object might move between the times that the two frames are taken, the corresponding features that refer to that object are incorrectly representing the transformation between the point clouds. As an example, imagine an extreme case where the two frames are taken from the exact same location, but an influential object in the scene is rotated upside down. The transformation based on the features of that object will indicate a 180 degree rotation for the transformation, while no transformation is needed at all. The same problem exists with scenes that include similar objects in the scene, which is the case with many man made environments such as buildings and offices that include elements like bricks, panels, windows and furniture which are created identically. In this scenario, the feature matching algorithm matches features that belong to different objects in the scene, such as identical edges of numerous identical chairs in a classroom.

3. Speed: Depending on the implementation of the feature detection and matching algorithms, the performance of the method might not be suitable for real-time applications, as reported in [30]. Also, other than feature detection and matching, the steps required after correspondence detection add to the execution time of the algorithm.

4. Non-Overlapping Point Clouds: In some feature-based registration methods

such as [30], the correspondence results are used to generate a coarse alignment, which is then used as a good initial pose for other registration algorithms such as ICP to find a finer transformation. The main problem with registration pipelines that finish by performing ICP for fine-tuning the results is that they contain the shortcomings of the ICP algorithm. As discussed before, this includes the problem with non-overlapping point clouds which causes the transformation estimation to fail even with a relatively good transformation.

Considering the first two challenges, it is necessary for feature-based registration techniques to incorporate a robust method for rejecting false feature pairs. RANSAC [22] is often used, as seen in RGBD-ICP by Henry et al. [30]. RGBD-ICP uses image-based feature detection algorithms with true pair detection for coarse registration and combines it with the ICP algorithm for further refinement of the transformation. This work successfully addresses the Initial Pose problem with the ICP algorithm, however by using ICP it inherits ICP's weakness against partially overlapping 3D images.

In the case of a large non-overlapping set of images, the first stage of the algorithm provides the ICP algorithm with a good estimate of the solution, but ICP tends to move away from this solution by pushing the non-overlapping parts of the images together to minimize the error. In other words, not only incorporating the ICP algorithm in such scenarios does not contribute to the refinement of the alignment, it renders it useless by moving away from the pose. In order to achieve fine alignment, the ICP algorithm needs fine details; hence down sampling is not performed rigorously so that the down sampled clouds would contain a fairly large number of points in order to preserve scene details. Processing large number of points with the ICP algorithm adds a significant performance cost to the algorithm (500ms per frame).

While our solution is a feature-based registration method, we aim to address all the aforementioned issues. The details of the proposed algorithm are discussed in the next chapter.

3.5 Datasets

A list of many existing RGBD datasets is available at [21]. Following is an overview of the registration specific datasets and why they were not fit for our experiments:

- TUM Benchmark Dataset [60]: the conditions required for our experiments such as different levels of lighting and complexity were not captured in this dataset. Furthermore, the number of frames with ground truth information were limited.
- RGB-D Dataset 7-Scenes [25]: this dataset was published by Microsoft, but only includes frames from 7 scenes.
- IROS 2011 Paper Kinect Dataset [49]: this dataset only includes frames from one scene in 3 different complexity conditions.
- "When Can We Use KinectFusion for Ground Truth Acquisition?" [43]: perhaps the most suitable of the available datasets, this dataset consists of frames from 57 indoor environments. Unfortunately, the ground truth is generated automatically by KinectFusion software, which employs existing registration algorithms such as ICP. This defeats the purpose of our experiments, as we require benchmarking Derees versus the existing algorithms.

- DAFT Dataset [27]: this dataset only includes frames of planar scenes such as walls or floors.
- ICL-NUIM Dataset [28]: this dataset only includes 8 sequences of frames from 2 environments.
- "Automatic Registration of RGB-D Scans via Salient Directions" [70]: this dataset includes images from long range laser scans which does not conform to our application which applies to indoor short range environments and consumer cameras.
- Stanford 3D Scene Dataset [71]: this dataset only includes frames from 6 scenes.

Ultimately, we opted for creating our own dataset as the existing datasets were either limited in the number of images, or did not contain the expected variety of conditions required for the different types of experiments that we envisioned.

Chapter 4

DeReEs

In this chapter we focus on the registration technique that we have proposed, which is referred to as DeReEs, short for the 3 main steps of the registration pipeline: Detection, Rejection, Estimation.

4.1 Registration Pipeline

Derees operates within 3 main steps which are explained further below: correspondence detection based on 2D feature detection and matching, false corresponding pair rejection and fine transformation estimation.

The pipeline of the registration method we propose is depicted in Figure 4.1. The inputs of the Derees algorithm are the two point clouds (or depth frames for less memory consumption) and their respective RGB information, captured from cameras placed at different positions which share a certain portion of the scene. The input can be generated in real time from any RGB capable 3D camera such as Microsoft Kinect. The output of the algorithm is a 4×4 transformation matrix that transforms

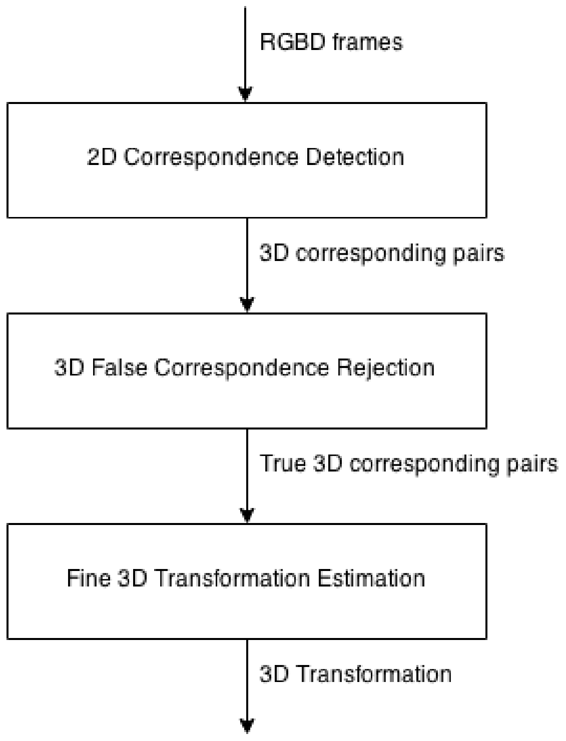

RGBD frames
2D Correspondence Detection
3D corresponding pairs
3D False Correspondence Rejection
True 3D corresponding pairs
Fine 3D Transformation Estimation
3D Transformation

Figure 4.1: DeReEs registration pipeline consists of 3 main steps: correspondence detection, false correspondence rejection and transformation estimation.

the source cloud to the coordinates of the target cloud, where the source is the input from the left camera and the target is the right camera, although this can also be the other way around.

4.1.1 Step 1: Corresponding Pair Detection

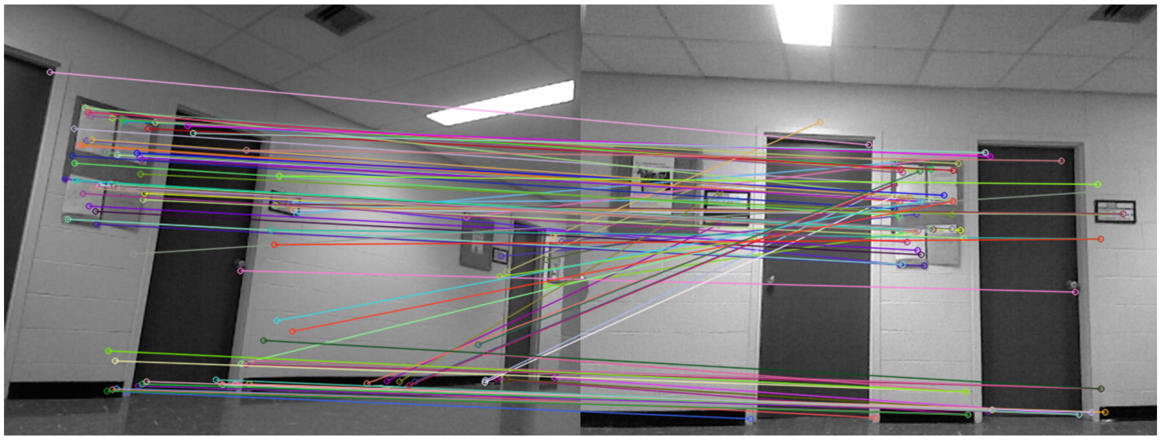
The aim of this step is to generate 3D corresponding pairs to be used as a base for transformation estimation. 2D feature detection algorithms are used to extract features from the RGB information. Depending on the algorithm used, these features correspond to corners, edges or other visual features in the image. Feature descriptors are extracted and features from each image are matched together to derive corresponding feature pairs. The output of the feature detection and matching algorithms is a set of point pairs, with each point of the pair belonging to one of the images, and it is estimated that the points in the pair are visually corresponding to each other. This is described by two arrays of points known as features, defined by their 2D position in the RGB image and feature descriptor, as well as a matching data structure which indicates which point in one image relates to which point in the other image. Multiple algorithms might be executed and results can be aggregated for a potentially more accurate alignment in exchange of speed.

3D coordinates of the corresponding feature pairs are calculated based on their position in the 2D image and their corresponding values in the depth channel. In the case of working with point clouds, an easier solution to extract the 3D coordinates of the feature pairs is to look up the point cloud's point array, accessing the point in the point cloud that represents the same feature pixel in the RGB image. The point data structure will contain its 3D coordinates. Assuming that the image and point cloud have exactly the same size and order in storing point information, this can be as easy as accessing the point cloud array at the same index that the feature pixel uses in the image's pixel array; which was the case in our implementation, where we use PCLs PointCloud class for storing the point cloud [53] and OpenCVs Mat class for storing the image [9].

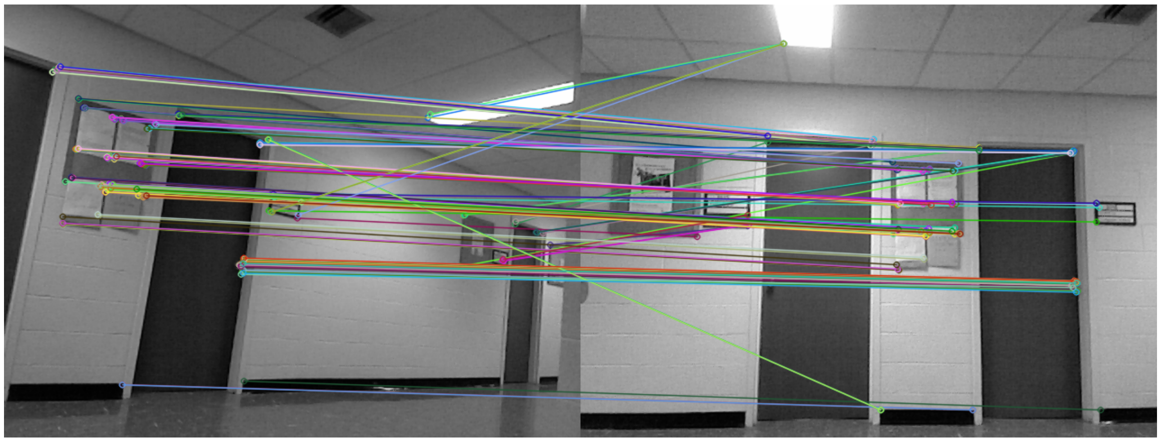
In the implementation of our algorithm, we can use either OpenCV's SURF or ORB RGB feature detection algorithms for determining the initial correspondence and then perform brute force feature matching to find pairs among the discovered features between the two images. Although we tested both SURF and ORB, we opted to use SURF for the experiments since it provided a larger number of feature pairs compared to ORB and performed more consistently across different scenes. The time complexity for SURF and ORB algorithms without parallelization is $O(W \times H)$ with W and H representing the width and height of the images. The complexity of brute force feature matching without parallelization is $O(N^2)$ with N representing the number of features extracted by feature detectors. The GPU implementations of SURF and ORB feature detection algorithms, as well as the GPU implementation of the brute force feature matching algorithm have been used in this project, thus decreasing the time complexity significantly.

Matching algorithms provide a correspondence score for each pair matching which we used to select only the top 10% to 20% of the feature pairs (usually between 100 to 500 pairs after feature matching depending on the images). Figure 4.2 shows the results from feature detection and matching algorithms for one of the scenes.

Considering that the depth map from depth sensing cameras is incomplete com-

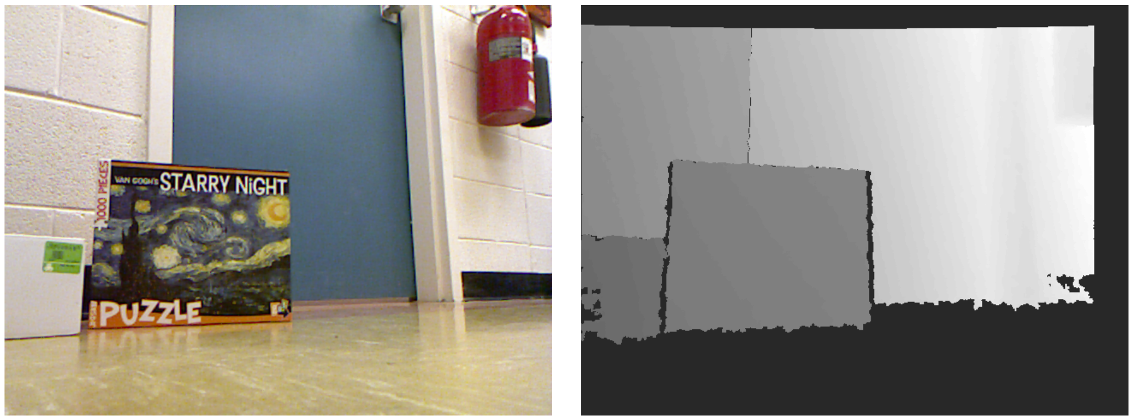


(a) SURF feature detector



(b) ORB feature detector

Figure 4.2: The resulting corresponding pairs from performing brute force feature matching on features detected by (a) the SURF and (b) ORB feature detectors.



(a) RGB frame (b) Depth frame

Figure 4.3: Point clouds do not include information for all the points present in the RGB frame. The black parts in (b) represent the missing parts of the depth map due to reflective surfaces and cropped edges.

pared to the RGB information, a noticeable number of feature pairs are lost at this step (10%-40%). Missing depth information mainly occurs for two reasons:

- Depending on the type of depth camera, depth values may not be present for shiny surfaces, very dark objects or light sources.
- The RGB sensor and the depth sensor are located slightly apart from each other in the depth sensing camera. Hence, depth information may not be available for parts of the RGB information, particularly around the vertical edges of the depth map.

Figure 4.3 depicts the RGB image, depth frame and the produced point cloud of a scene. The missing parts are visualized as black. Other than the occluded parts (e.g. behind the box), it can be noted that the depth frame and consequently the point cloud are missing parts of the scene in its borders, as well as on some of the reflective surfaces such as the floor.

The output of this step is a data set of feature pairs described by their 3D coordinates. The number of pairs in the data set is denoted as F throughout this chapter.

4.1.2 Step 2: False Corresponding Pair Rejection

Not all the pairs from the previous step are true corresponding pairs. False corresponding pair are formed when the detection and matching algorithms have matched two points that do not represent the same or corresponding point in the real world. With feature-based registration methods the number of corresponding pairs is relatively low (hundreds or thousands of pairs) compared to other methods [37] such as ICP or 3D-NDT which consider all points of the point clouds as potential pairs (300000 pairs for VGA images). Consequently, even a small number of false pairs will degrade the alignment significantly. The aim of this step is to detect all false pairs and remove them from the data set.

To detect such pairs, we follow this approach: assuming we know a roughly successful transformation, this transformation can be applied to the coordinates of the features in the source cloud. After the transformation, the points of a true feature pair are placed closely together, while points of a false feature pair will have relatively further distance from each other. Considering this, only a rough transformation estimation is required to detect and reject false feature pairs. Notice that in 3D space, only 3 corresponding pairs are required to estimate a transformation. In this case, if 3 feature pairs were found to be true pairs, we can estimate a first approximation to map them, which we refer to as the "coarse transformation", and detect the false pairs afterward.

For this purpose, 3 pairs can be randomly selected from the data set to be considered as the true pairs; but the reliability of this random selection is determined by the ratio of true and false pairs, which we refer to as α.

$$\alpha = \frac{TrueCorrespondingFeaturePairs}{AllCorrespondingFeaturePairs} \tag{4.1}$$

Considering that α is the ratio of true corresponding feature pairs to all pairs, the probability of selecting one true pair randomly is α. The selection of 3 true pairs randomly from the set in one try requires 3 consecutive successful true selections, with no false selection in between. The probability of such sequence of events is α^3. With scenes where $\alpha = 0.5$ the chances of finding a correct coarse transformation would be of only 12.5%.

We assume that by repeating this random selection for enough times, 3 true feature pairs will be eventually selected with high certainty. While the probability of missing 3 true features with one try is $1-\alpha^3$, if the selection is repeated n times, the probability that none of the selections consist of 3 true feature pairs, P_{miss}, is $(1-\alpha^3)^n$. The parameter n can be selected in a way that P_{miss} falls below the desired certainty threshold, where n is selected based on scene complexity as described below.

$$P_{miss} = (1-\alpha^3)^n \tag{4.2}$$

$$n = \log_{1-\alpha^3} P_{miss} \tag{4.3}$$

With $\alpha = 0.5$ and 69 repetitions, the probability that no 3 true feature pairs are randomly selected falls below 0.01%.

This can be considered similar to a bare-bone version of RANSAC [22] outlier detection approach. In the RANSAC algorithm, the data points that are initially selected randomly are often used to extend the selection to all data points, based on a function that computes how queried data points match the accepted data points. Similar to our case, we look for a selection of 3 correct data points and those 3 points basically determine the outlier data points that we are looking to reject.

Any of the randomly selected 3 true pairs can be used to generate the coarse transformation we require. The remaining problem is: the coarse transformation is required for detecting true and false pairs, while we need 3 true pairs to complete this task. We need a way to determine which of the random selections contain 3 true pairs, in order to derive the coarse transformation from that selection. A score or error metric is required to differentiate the good selection (a selection with 3 true corresponding pairs) from an incorrect one.

4.1.2.1 Transformation Evaluation Metric

The correctness of a random selection can be assessed based on the correctness of the transformation it produces. If the transformation generated by 3 corresponding pairs has signs of being close to the correct transformation, the 3 selected pairs may be true pairs. Conventionally, the correctness of a transformation is determined by applying the transformation to the point clouds and then measuring metrics such as Sum of Absolute Distances or Squared Sum of Distances between corresponding pairs, defined as closest points in two clouds or through other methods [52]. This is inaccurate for the similar reasons mentioned before: non-overlapping areas of two clouds will cause the distance metric to increase while the clouds might be aligned correctly, since points in mutually exclusive parts of the clouds have no neighbouring points in the other cloud. In other words, an incorrect alignment that pushes the non-overlapping parts together may produce a better value for the distance metric. It is also time consuming since it applies the transformation on all or most points in the clouds.

The proposed metric for evaluating the correctness of the transformation in this work is a scoring metric that calculates the number of all features for which the distance to their respective points in the other point cloud is less than a certain distance threshold (d_T) after applying the proposed transformation only on the 3D feature pairs extracted from previous step. As mentioned before, a correct transformation will transform the point cloud in a way that truly corresponding pairs will be placed in close proximity, while false apparent pairs will have noticeable distances defined with a threshold value (d_T). Our metric counts the number of true pairs by taking advantage of this observation. After applying the proposed transformation from a random selection of 3 pairs on all pairs, the number of pairs that have smaller distance than the threshold are counted as the score for that random selection. The proposed transformation from 3 pairs is calculated by translating and rotating 3 points to their respective matches, based on their centre points and angular differences.

Note that due to the noise in the depth data and inaccuracy in the depth map, camera resolution and inaccuracy of the feature detection algorithm, even true pairs would not have identical coordinates after the transformation is applied. The need for using a distance threshold in the scoring process arises from this issue. The distance threshold is selected in a way that these inaccuracies along with the inaccuracy of a coarse transformation does not cause too many true pairs to be rejected.

It is important to note that any set of 3 true pairs generates a coarse transformation which is very similar to transformations generated by other sets of 3 true pairs, a coarse transformation which is relatively close to the solution transformation. Unlike true pairs, transformations between most false pairs are arbitrary and it is extremely unlikely that an incorrect transformation results in a majority of false pairs being placed close by. In the case of an incorrect transformation, the 3 feature pairs will have relatively small distances to their pairs and most true pairs, as well as the majority of false pairs, will have noticeably large distances, so the transformation that minimizes the total amount of pairs with distances under the threshold is most likely to be the true transformation which we set as the coarse transformation produced in Step 2.

The following is the process followed to obtain the coarse transformation:

1. Random selection: 3 feature pairs from the set generated in Step 1 are randomly selected - $(A1, B1, C1)$ from the source cloud and their respective pairs $(A2, B2, C2)$ from the target cloud.

2. Transformation estimation: transformation T is estimated in way that $T(A1, B1, C1)$ is equal/close to $(A2, B2, C2)$.

3. Transformation: T is applied to all the feature pairs in the data set.

4. Scoring: the pairs for which their distance is less than the threshold are counted as the score for transformation T.

5. Repeat steps 1 to 4 n times.

6. The transformation with the best score is selected as coarse transformation (T_c).

Finally, the coarse transformation obtained is applied to all feature pairs and all the pairs that have a greater distance than the distance threshold d_T are removed from the set. In our experiments it is observed that 5% to 30% of the pairs remain at this step, depending on the difficulty of the scene for feature detection and matching algorithms.

This step has a complexity of $O(n \times F)$, where n and F respectively represent the number of iterations and the number of feature pairs obtained from Step 1. The number of iterations is selected between 20 to 300. Through empirical observation, we have found that values ranging from 150 for simple scenes and values closer to 200 for complex scenes produce optimal results.

4.1.3 Step 3: Fine Transformation Estimation

The coarse transformation calculated in the previous step is the result of a transformation estimated by using only 3 true pairs. By considering all true pairs in transformation estimation, a finer transformation can be achieved, since averaging the results from all pairs can help decrease the effect of measurement errors from 3 pairs. The remaining feature pairs from the previous step are used to estimate a finer transformation which best transforms the features from one cloud to their corresponding features by minimizing the least squares error between the two sets. Performing this with what we expect to be only the true pairs reduces the risk that false feature pairs affect the overall accuracy. Minimizing the least squares is performed by an algorithm based on singular value decomposition (SVD) proposed by Arun et al. [2] which has a linear time complexity. We used PCL's implementation of this algorithm in our project, with equal weights for all corresponding pairs.

4.2 Alternative Approaches Explored and Proposed Improvements

Different alternatives have been tried at each step of the algorithm, which are discussed in this section. Furthermore, possible improvements that have not been applied during the course of this research are proposed. Time constraints prevented us from examining these possible improvements on the algorithm during the research period. These subjects are categorized based on the components of the algorithm that they would be applied to:

4.2.1 Correspondence Detection

4.2.1.1 Feature Detection on the Depth Image

Feature detection on the depth image was also tried, with little success. As mentioned earlier, the depth images include numerous missing pieces of information. These missing pieces of the dataset must be replaced with some value for the feature detector. The two choices are either 0 or the maximum acceptable value, respectively representing the minimum distance or an infinitely far distance. In both cases, the point

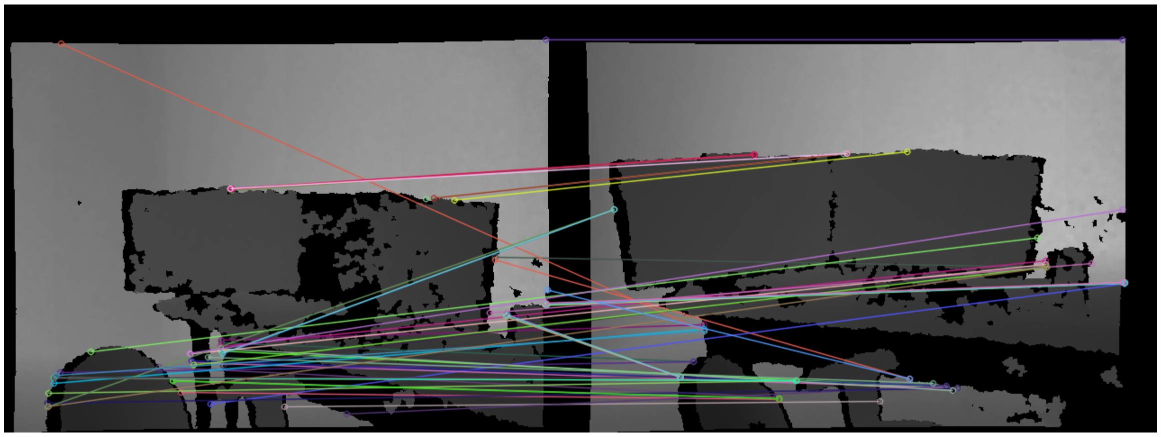


Figure 4.4: Corresponding pairs from the SURF and ORB feature detectors and brute force matching algorithms.

is represented as black or white in the depth image. In our implementation we chose 0 for missing information.

This sudden change in colour to black (or white) causes the feature detector to treat these elements as edges of objects. Since the missing pieces are vastly different between the two images, the detected features around the missing information do not correspondence to each other between the 3D image pair. Furthermore, many of the features will be selected on these elements with missing depth information, which prevents us from calculating the 3D coordinate for such pairs, rendering them useless (see Figure 4.4).

4.2.1.2 Edge Detection

Edge line detection has also been tried for this purpose (see Figure 4.5) with little success due to low correspondence between detected lines. There are a number of problems with this method:

1. The edges detected between the two image pairs have different lengths, making it difficult to determine a point to point correspondence for transformation estimation.

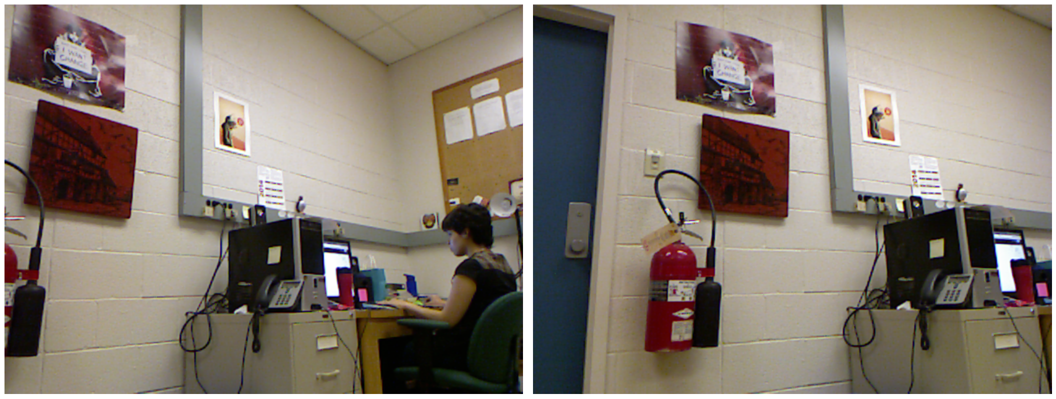
2. A single edge detected in two of the images does not necessarily correspond to the same object. Assume the edge detected between a background and foreground object. In one point cloud, the edge might be placed over the background object, while in the other point cloud the corresponding edge might be placed over the foreground object. This causes the corresponding points in such edge to have extremely different positions in 3D coordinates.

3. The low number of edges makes the algorithm highly prone to incorrect edge matching.

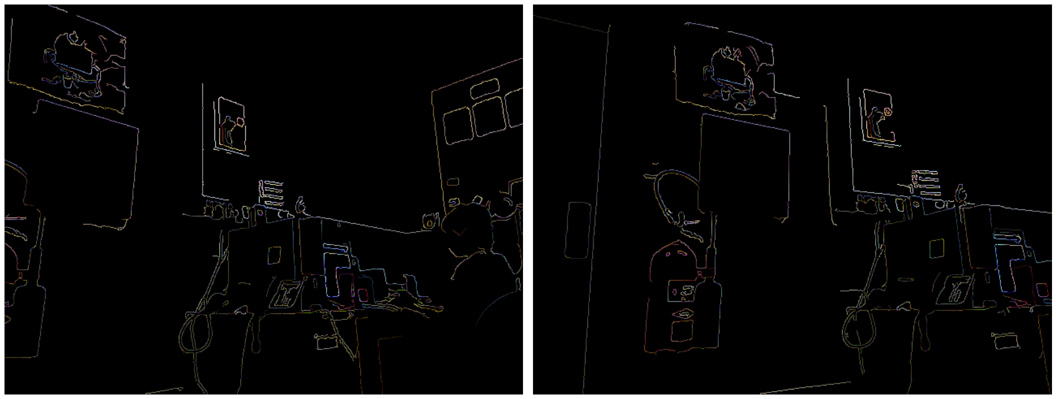
4.2.1.3 Object Matching Using Colour Information

We also tried the possibility of detecting an object in both image pairs using colour information. We assumed that in both point clouds, an object with a unique colour exists, which can help in determining a coarse transformation estimation or correspondence. We filtered the pixels based on different colour ranges and tried fitting the remaining pixels into blobs. Unfortunately, the results are highly dependent on the scene conditions. Low light environment causes the RGB images to have low colour quality. The high level of noise causes the different colour ranges to exist all over the image, making it hard to robustly determine a blob that represents an object.

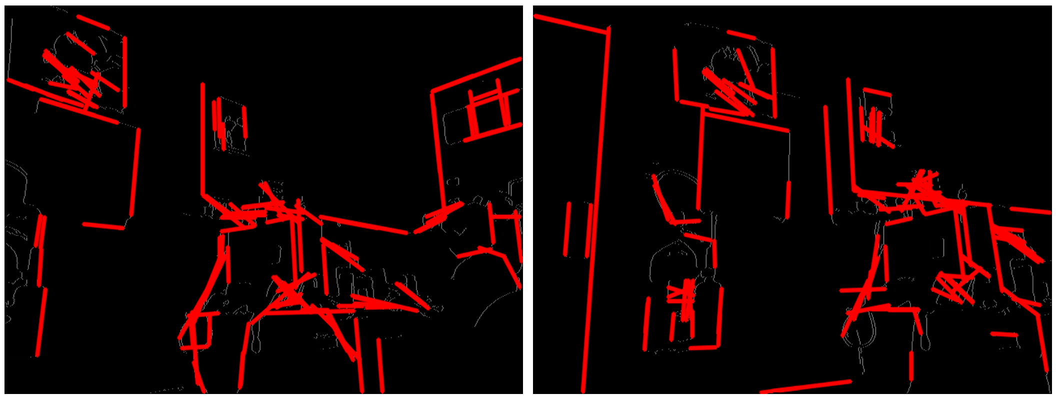
A possible improvement which we have not tried is to use multiple RGB feature detectors (e.g. SURF and ORB and SIFT) in parallel and accumulate their results,



(a) Original image pair



(b) The image pair after applying canny edge detection



(c) The image pair after applying Probabilistic Hough line transform over the canny edge detection result

Figure 4.5: Probabilistic Hough line transform over Canny edge detection.

in return for lower speed.

4.2.2 False Corresponding Pair Rejection

Different scoring metrics were tried for the transformation evaluation component of the algorithm: Triangle Conformity: having selected 3 random pairs from the data set, the quality of the selection can be determined by how the 3 pairs conform to each other, independent of the other pairs. Assuming that the 3 pairs are true pairs, the triangular properties (angles and vector sizes) created by the 3 points in the source point cloud are consistent with the triangular properties of their pairs in the target point cloud.

We took advantage of this fact and calculated an error based on how different the two triangles are, considering the sum of absolute vector length differences. P_1 and P_2 represents the perimeters of the triangles formed by the 3 points in target and source point clouds:

$$Metric_1 = |P_1 - P_2| \tag{4.4}$$

After executing the random 3 pair selections for n iterations, the set with the least error value would be selected to determine the coarse transformation. The problem with the previous metric is that in the case of a false random selection with very small vector lengths (small triangle) and a true random selection with very large vector length (large triangle), the metric would favour the false random selection. The vector lengths differences between two large triangles are potentially larger than vector lengths differences of two minuscule triangles regardless of how well they conform. To fix this problem, we scaled the error metric based on the triangle size in a way that the size of the triangle would impact the error metric.

$$Metric_2 = \frac{|P_1 - P_2|}{max(P_1, P_2)} \tag{4.5}$$

While the previous metric performed well in most cases, it would become problematic when the 3 selected points were linear or closely linear even when they are true pairs. In case the points can be fit on a straight line, the transformation estimation fails due to not being able to determine a correct rotation. As mentioned before, even the true corresponding pairs are not ideally corresponding due to feature detector accuracy or depth sensor noise. When the 3 pairs are placed in a way that they closely fit a straight line, a very small location error on the position of the points causes a significant rotation error in the transformation estimation.

To avoid this, we tweak the metric in a way that it favours the triangles that show equiangular properties. We determine how well the 3 points are equiangular by using the sine of their smallest angle. In the formula below, A represents the array that contains the angle sizes between points in both points clouds.

$$Metric_3 = \frac{|P_1 - P_2|}{max(P_1, P_2) \times sin(min(A_i))} \tag{4.6}$$

In this metric, the sine of the smallest angle from linear points is close to 0, causing the error metric to increase toward infinity. On the other hand, the sine of the smallest angle for perfectly equiangular 3 points would be $\sin(60) = \frac{\sqrt{3}}{2}$, causing the error metric to increase slightly.

A common issue with all of these metrics proposed during our research is the problem of the position error: While two sets of 3 3D points might have a small error metric, showing that they both contain true pairs, one of them might perform significantly better due to less position error and how the position error transforms the point cloud. So, it becomes important to consider other points in the cloud rather than just the 3 points, which led us to choosing our current metric: a scoring based on the number of all corresponding pairs that are placed closely after applying the transformation estimated based on the 3 pairs.

A possible improvement to the current process of the algorithm would be to mix the current scoring metric with the triangle conformity error metric: depending on the ratio of true feature pairs, a large number of incorrect random selections can be rejected using the triangular properties before going ahead with calculating the scoring metric for them. Considering that the scoring metric requires transformation of all feature pairs in the cloud and calculating their distances for numerous iterations, this can reduce the execution time of the algorithm significantly.

4.2.3 Transformation Estimation

Initially, we only used the transformation from the 3 true pairs with the best score as the final transformation to save execution time. Considering that this transformation aligns many true pairs in a way that they are placed close together, the transformation would be very close to the ideal solution. The problem is that the scoring is discrete, hence many of the sets of 3 pairs have the exact same score: the number of all true features pairs; while based on position error their transformations are not the same.

While we opted to perform a rigid transformation between the pairs that are counted by the scoring (the true pairs), possible improvements exist: the scoring can be converted to a non-discrete metric by taking into account the distance of the pairs that are close together. In other words, rather than the number of pairs, the scoring metric would measure the distance of pairs that fall below the threshold and affect the score on a weighted basis.

We have also experimented with using ICP for fine transformation estimation. The 3 pairs from the previous step would indicate a coarse transformation, then the ICP is used to fine-tune the alignment. In cases of large non-overlapping sections, the ICP algorithm breaks the alignment. In other cases, the results are similar to the rigid transformation estimation between the remaining true pairs, but the performance time is significantly worse due to the large number of input data that is required for a fine-tuned ICP algorithm.

One possible solution to the non-overlapping problem might be to crop out the non-overlapping parts of the clouds based on the alignment determined by the true pairs, then perform the ICP algorithm on the remaining points of the clouds for further fine-tuning.

An improvement to the current solution is to consider the feature matching confidence metrics in the final rigid transformation estimation. The matching confidence or the matching score is assigned to two features from the two images by the feature matching algorithm. This metric indicates the confidence of the algorithm in identifying the two features as a pair. In other words, pairs with better matching scores are assumed to have a higher chance of being true pairs compared to feature pairs with lower scores. This metric is calculated by different methods depending on the feature descriptors used (different methods surveyed in [72]). For example, the metric can be calculated by assessing the similarities of the feature descriptors of the two features by using the Euclidean or Hamming distance of the feature descriptor matrices. In our current solution, all remaining true features in the final step of the algorithm have the same weight in determining the final transformation. Assuming that the feature pairs with better confidence evaluation have less position errors, corresponding pairs in the final step can be assigned weights based on their matching confidence from step 1 for a more accurate transformation, similar to weighted variants of ICP (see Section 3.2).

Chapter 5

Evaluation

In this chapter, we focus on the evaluation on our proposed algorithm against other registration algorithms. First, we introduce our registration data set which can be used by the research community for experimentation with registration. We will then describe the implementation and configurations used for our experiments. Experimentation hypotheses are presented next and then we continue to assess the performance of our algorithm under different conditions to validate our hypotheses.

5.1 Data-Set Collection

For the purpose of experimentation, we require 1) a collection of 3D image pairs under different conditions in indoor environments and 2) their known transformation (ground truth). Existing data sets suitable for registration and pose estimation experiments either lacked large number of images or were not collected with the various conditions we required for our experiments such as different lighting, complexity, etc. We created our own data set containing 50 sets of 3D images and their ground truth transformation and have made it publicly available at [56] for the research community.

Images were taken with the Microsoft Xbox 360 Kinect camera at 640×480 resolution. Each set contains 2 to 5 images of the same scene from different viewing points. For each image, we recorded 3 files to encapsulate different use cases:

- A Point Cloud library file (.PCD) that contains the data structure used by Point Cloud Library (PCL) for storing the point cloud information in memory. This includes color and coordinate information, as well as other point cloud properties. The file can be loaded using the PCL to create a point cloud object.
- A loss-less image file (.PNG) for the RGB information of the camera.
- A loss-less image file (.PNG) for the depth information of the camera.

The images were taken in indoor office environments under various conditions of feature richness, transformation distance, transformation type (rotation vs. translation), lighting levels, distance to the scene, complexity, repetitiveness. Some images also include dynamic scenes where the scene elements have changed position between image pairs.

The ground truth transformation is estimated between the first image and the rest of the images in each set. The ground truth is estimated by performing our algorithm on the image pair and then manually fine tuning the alignment until misalignment in the cloud is not visually detectable (Figure 5.1). Manual fine tuning is performed by rotating and moving the cloud in all 6 degrees of freedom using keyboard input, with precision of $1cm$ for translation and $0.5°$ for rotation. The process of manual fine alignment is assisted by an interactive application which measures the square of

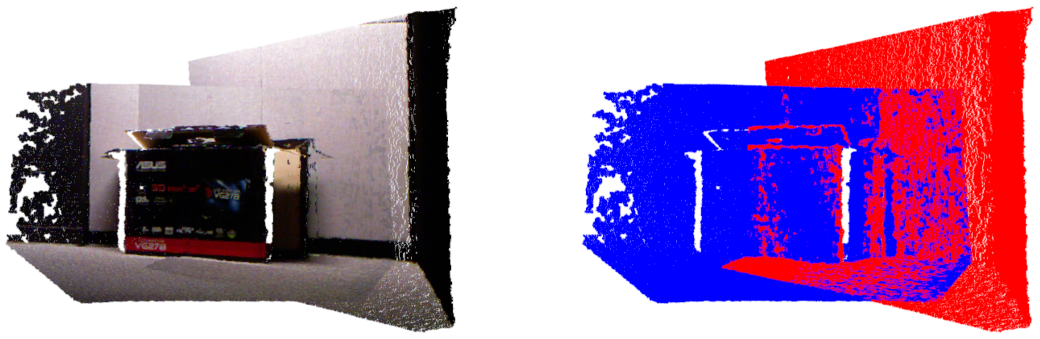


Figure 5.1: Visualization of the registration with ground truth transformation.

distance error metric between points of an overlapping section of the cloud in real-time as user moves the clouds. The overlapping section where the square metric is calculated for is selected manually by the user performing the alignment. The ground truth transformation is a 4×4 3D transformation matrix which is saved as plain text in the data set.

5.2 Implementation, Experimentation Environment and Configuration

This algorithm is implemented with the help of open-source libraries and C++ programming language. The OpenCV [9] library is used for the GPU implementation of the SURF and ORB feature detection algorithms, as well as GPU implementation of brute force feature matching algorithm. Point cloud I/O (file or device), visualization and simple point cloud processes are implemented using the Point Cloud Library (PCL) [53]. Implementations of the ICP and 3D-NDT algorithms in the PCL are

Method	Success rate (%)	$Error_t$ (cm)	$Error_r$ (degree)	Speed (ms)
3D-NDT	10	17.86	28.48	21063
ICP	40	17.65	10.98	42.37
RGBD-ICP	NA	12.3	3.3	730
Derees	97	2.45	2.41	36.68

Table 5.1: Comparison of Derees, ICP, 3D-NDT and RGBD-ICP in terms of success ratio, transformation error and speed.

used for comparison purposes in the following experiments.

The configuration of the machine that performed our experiments is as following: Intel Core i7 CPU 960 @ 3.20GHz (8 cores), 12GB memory, NVIDIA Quadro K5000 graphics card (4GB memory - 1536 CUDA cores), 64-bit Ubuntu 12.04 LTS operating system.

5.3 Experiments

5.3.1 General Comparison

Derees is compared to ICP and 3D-NDT using a data-set of 10 RGBD image pairs with low to moderate transformation distances, each being executed 10 times for every algorithm (100 executions in total for each algorithm). Table 5.1 outlines the success ratio, transformation error and speed of all algorithms. Due to lack of access to implementations or data sets of RGBD-ICP, direct benchmarking was not possible. The RGBD-ICP values in table 5.1 represent the evaluations reported in [30]. Transformation error for each transformation is calculated as: 1) the Euclidean translation error in the 3 axes ($Error_t$) and the average of rotation errors around the 3 axes ($Error_r$). We defined any transformation estimation with more than $0.5m$ error in translation or more than $30°$ error in rotation as an unsuccessful registration as to identify completely flawed matches; any other similar values can be used, as unsuccessful registrations tend to have noticeably large errors. The error measurements in the table represent the average of the errors only for the successful registrations, to prevent unsuccessful registrations from skewing the error measurements.

The speed is the average execution time of all executions. For both ICP and 3D-NDT algorithms, it is essential to down-sample the $300k$ input from the Kinect camera. The clouds were down-sampled to $6-7k$ points for ICP and $2-3k$ points for 3D-NDT to achieve an execution time relatively similar to what has been reported in other publications such as [30][52][37].

Our algorithm performs at 36.68ms per frame (including the 2D feature detection step), capable of processing RGBD images at $27fps$. As seen in the table 5.1, Derees outperforms other algorithms in all metrics. It should be noted that it is possible to run both ICP and 3D-NDT faster by a stricter down-sampling, which in turn decreases accuracy. A direct speed comparison with the RGBD-ICP cannot be made due to different settings and configurations, although it is known that the majority of the time that RGBD-ICP uses is dedicated to the ICP algorithm with dense input for transformation refinement ($500ms$). It should be noted as mentioned in section 4.2.3, we tested the RGBD-ICP approach by using ICP in the final step of the algorithm. Unfortunately, the results indicate with no noticeable improvement on the accuracy, and as described earlier in Section 3.1, this caused the algorithm to be prone to failure with image pairs that include a noticeable amount of exclusive parts (small overlap), so we have discarded it as a viable solution for us.

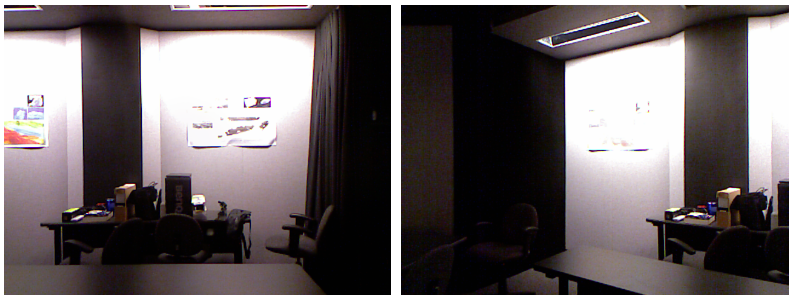
5.3.2 Robustness with varying amounts of overlap

To evaluate how our algorithm performs with varying degrees of partial overlap between image pairs against other algorithms, we generated a set of overlapping images from one scene. We did this by capturing an initial image from the scenes, then rotating the camera by 5 degrees before taking the next image and so on. The first image is used as the source point cloud, while consequent images are used as target point clouds. We performed this for one indoor scene, generating 12 images for the scene. Considering Kinect camera's field of view of 57 degrees, the amount of overlap between each image pair is calculated as the ratio of the overlapping portion of the clouds with respect to the whole coverage of the two clouds based on the amount of camera rotation r, where r is the amount of rotation from the initial pose (5° for second image, 10° for third image, etc.).

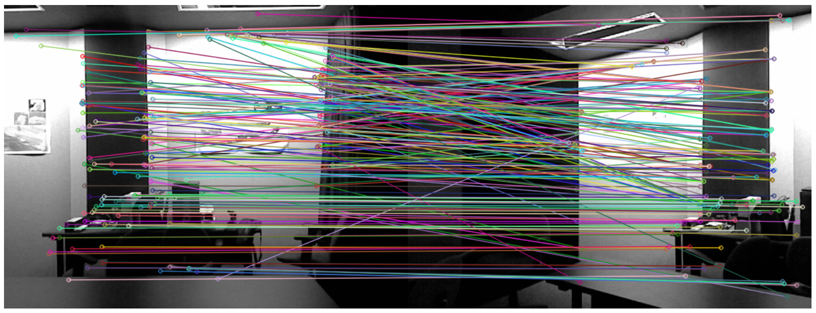
$$OverlappingRatio = \frac{57 - r}{57 + r} \tag{5.1}$$

Table 5.2 represents the average of error measurements and the success rate for different amounts of overlapping with different algorithms. Note that unlike the previous experiment, the error measurements in this experiment are the average for all executions. Figure 5.3 illustrates the success rate between the three algorithms.

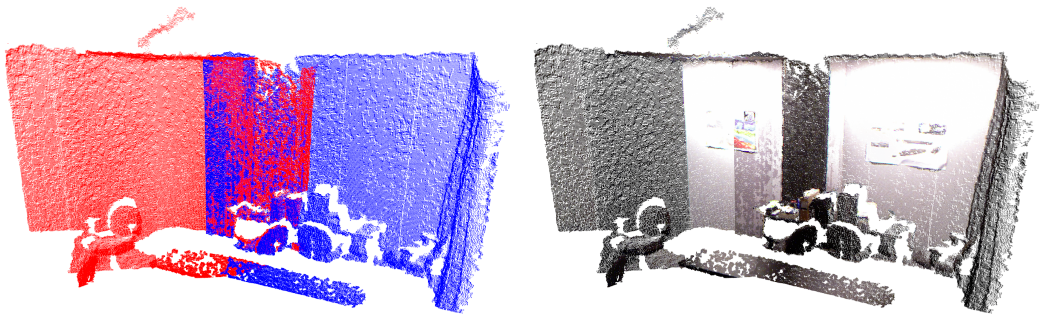
Derees outperforms other algorithms by successfully registering image pairs (such as Figure 5.2) with only 23% of overlap, compared to requiring of 70% and 83%



(a) The original image pair.



(b) Surf feature detection and brute-force matching results on the original image.



(c) Final registration result from DeReEs.

Figure 5.2: Successful registration of image pairs with only 23 percent overlapping using Derees.

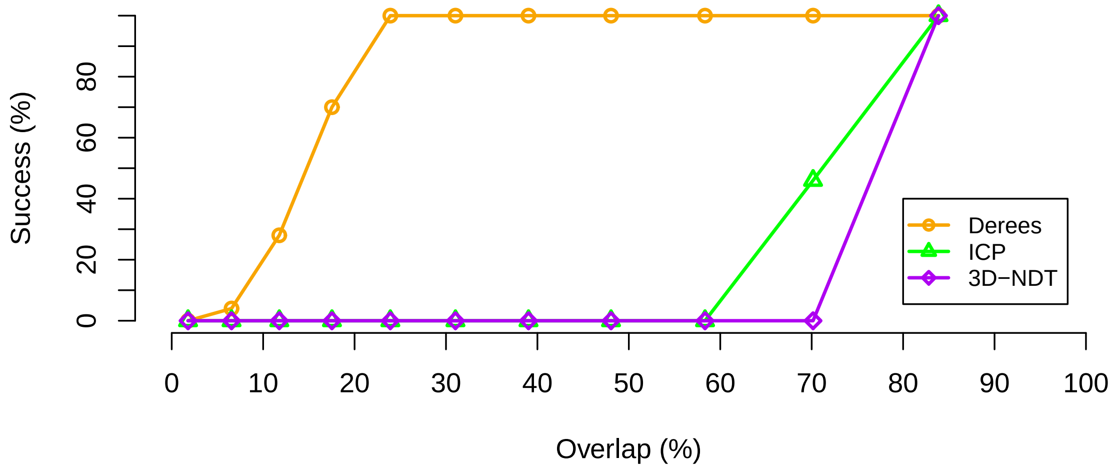


Figure 5.3: Success rate of Derees, ICP and 3D-NDT under different amounts of overlap between the image pairs of the same scene.

overlap for ICP and 3D-NDT. As expected, robustness of our algorithm deteriorates with decreasing overlap as the feature detection and matching algorithms begin to provide a large number of false corresponding pairs. For this experiment, RANSAC iterations has been set to 200, while still providing a real-time solution ($25fps$).

5.3.3 Robustness in feature-less scenes

Texture-less surfaces and low number of detectable features in the scene greatly affect the performance of the 2D feature detection algorithm. In such conditions, the randomized selection of 3 pairs for coarse registration estimation has a lower chance of success. Hence, the number of iterations used at this step becomes the crucial determinant in successful registration in poor conditions, which in return affects the speed of the algorithm.

We evaluated the accuracy and the speed of our algorithm with different number of

	Derees			ICP			NDT		
Overlap (%)	E_t (cm)	E_r (degree)	Success (%)	E_t (cm)	E_r (degree)	Success (%)	E_t (cm)	E_r (degree)	Success (%)
83	1.72	1.05	100	6.63	2.80	100	35.16	5.80	100
70	2.21	0.49	100	48.72	3.54	46	108.87	85.67	0
58	2.22	0.76	100	100.94	16.04	9	107.14	77.04	0
48	1.42	0.73	100	145.89	19.90	0	131.23	162.28	0
39	0.80	1.14	100	170.90	39.62	0	247.46	154.93	0
31	1.36	0.79	100	271.40	144.14	0	251.14	142.95	0
23	1.23	1.33	100	302.23	165.17	0	268.45	159.14	0
17	36.90	73.29	64	341.46	182.22	0	286.44	138.02	0
11	136.04	100.74	27	367.31	210.54	0	295.15	168.96	0
6.5	346.22	101.11	3	381.31	201.39	0	312.26	146.74	0
1.7	356.94	147.26	0	397.64	213.73	0	347.20	144.29	0

Table 5.2: Performance of Derees, ICP and 3D-NDT under different amounts of overlap between the image pairs of the same scene.

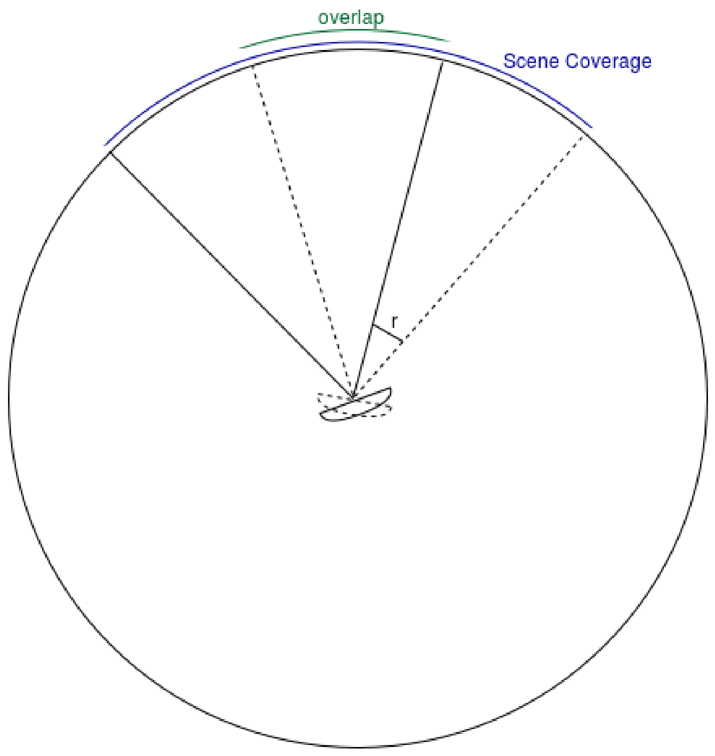


Figure 5.4: The overlapping ratio is calculated as the overlapping portion of the clouds with respect to the whole coverage of the two clouds, based on the amount of camera rotation r.

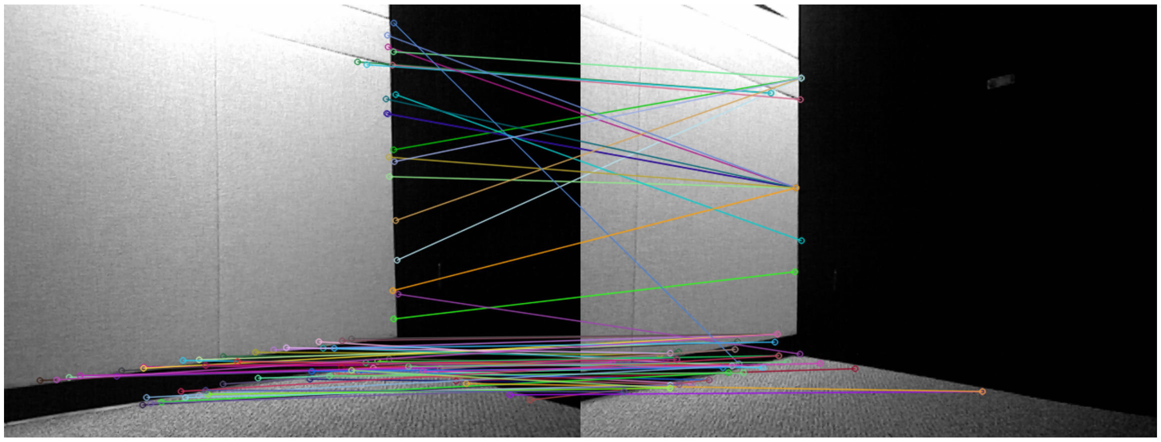
RANSAC iterations from 10 to 300. We performed this experiment with a feature-less image pair against a feature-rich one (see Figure 5.5). The feature detection algorithm has also been configured to produce less certain results in order to 1) increase the number of detected features in the feature-less scene and 2) emphasize the effect of RANSAC iterations in rejecting false corresponding pairs. Figure 5.6 illustrates the effect of increasing the number of iterations on success rate and execution time on each of these pairs. Figure 5.7 illustrates the error measurements for successful transformations over the number of iterations. Each data point is the average result of 100 executions (per image pair per iteration number).

As seen in Figure 5.6, increasing the RANSAC iterations has a direct effect on increasing the accuracy of the results, which is necessary for maintaining the registration accuracy of the feature-less scenes, although this effect is only valid to a certain extent (about 150 iterations). With a feature-less scene and large number of iterations, our algorithm has a similar success rate and error measurements as the results of the general comparison (subsection 5.3.1), while still maintaining the real time performance (14 frames per second at 150 iterations). The algorithm performs slower compared to the General Comparison experiment in overall due to the feature detection and matching configurations in this experiment, which results in a larger number of feature pairs to be processed (300 to 400 pairs compared to 70 to 150).

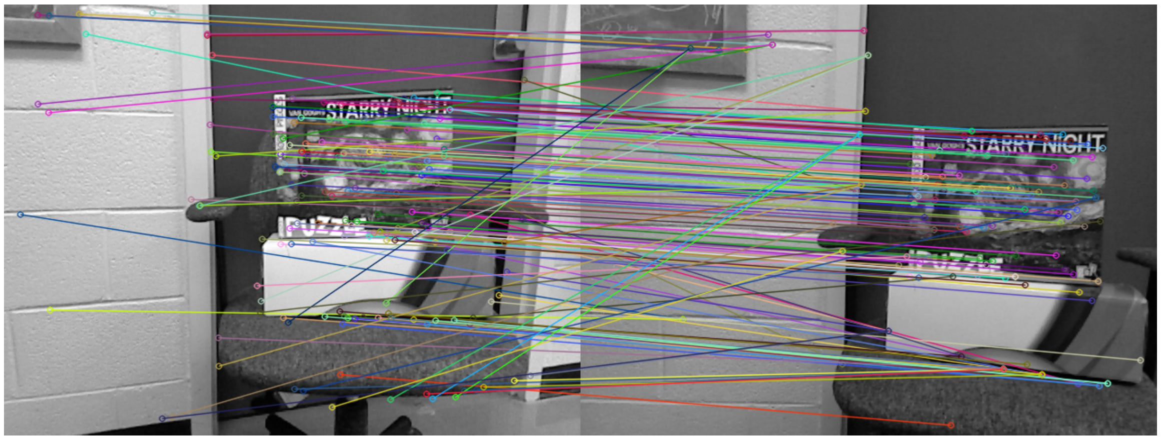
5.3.4 Robustness in low lit environments

Lightning in the scene affects the quality of the registration in feature-based registration methods. Low lit environments cause lower image quality and less colour variance in the image. Consequently, the effectiveness of the RGB feature detection and matching algorithms degrade. We evaluated the accuracy and speed of the algorithm in 3 different lighting conditions (bright, dimmed and dark) with 3 different scenes (figure 5.8). Figure 5.9 illustrates one of the scenes in 3 different lighting conditions and how the feature detection and matching results degrade with low brightness.

We performed the experiment with different numbers of RANSAC iterations (from 10 to 200) to see how the iterations contribute to the quality of the algorithm. In

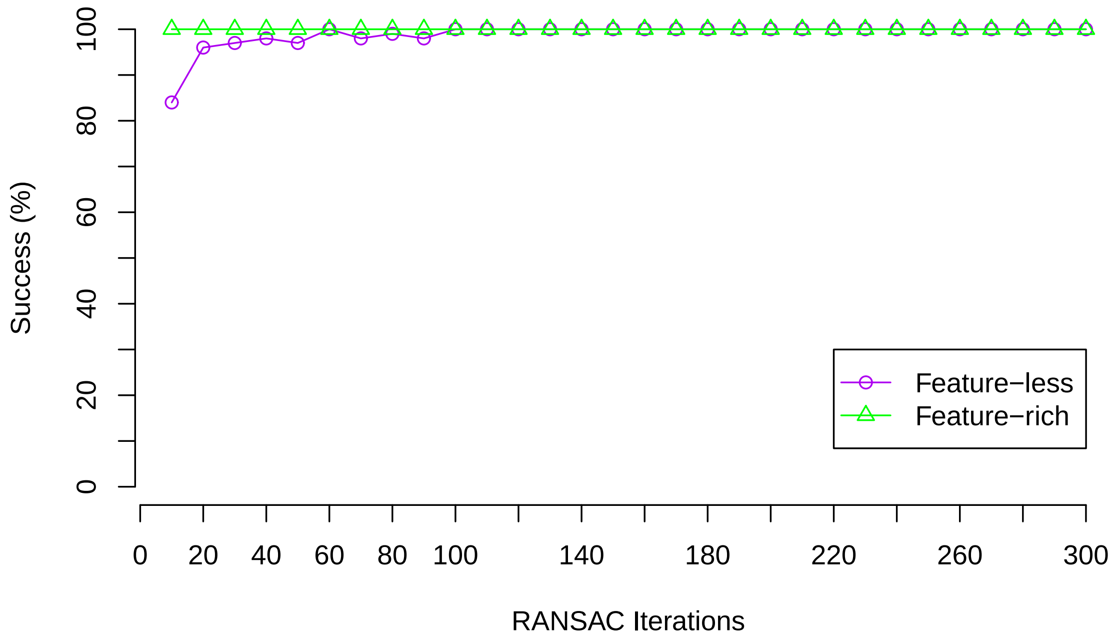


(a) Feature-less scene

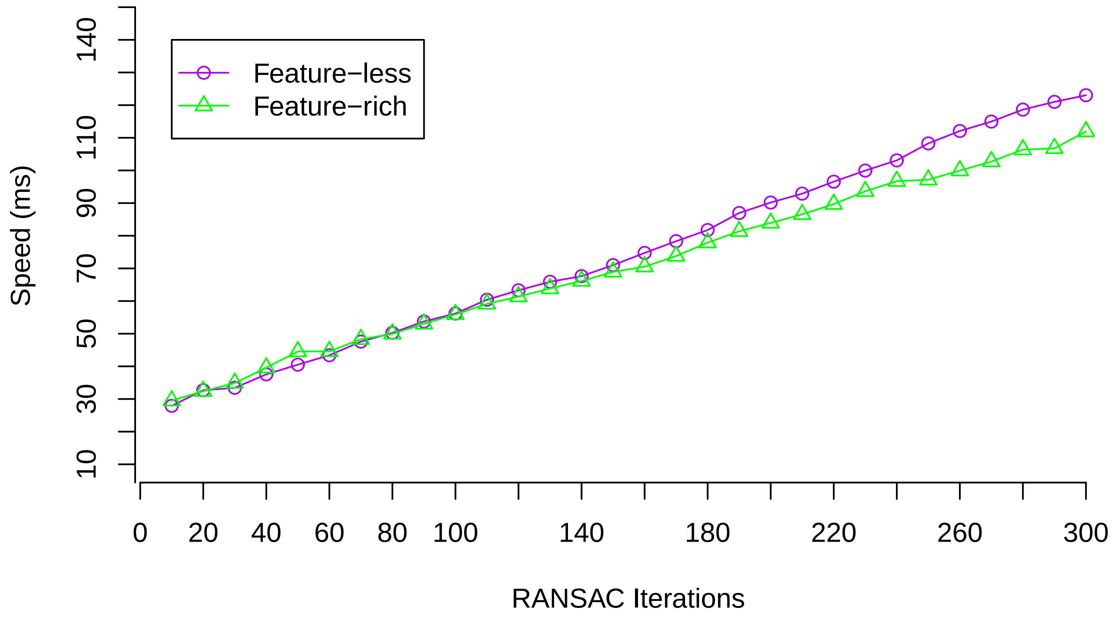


(b) Feature-rich scene

Figure 5.5: Comparing the performance of the algorithm in feature-less scenes against feature-rich scenes.

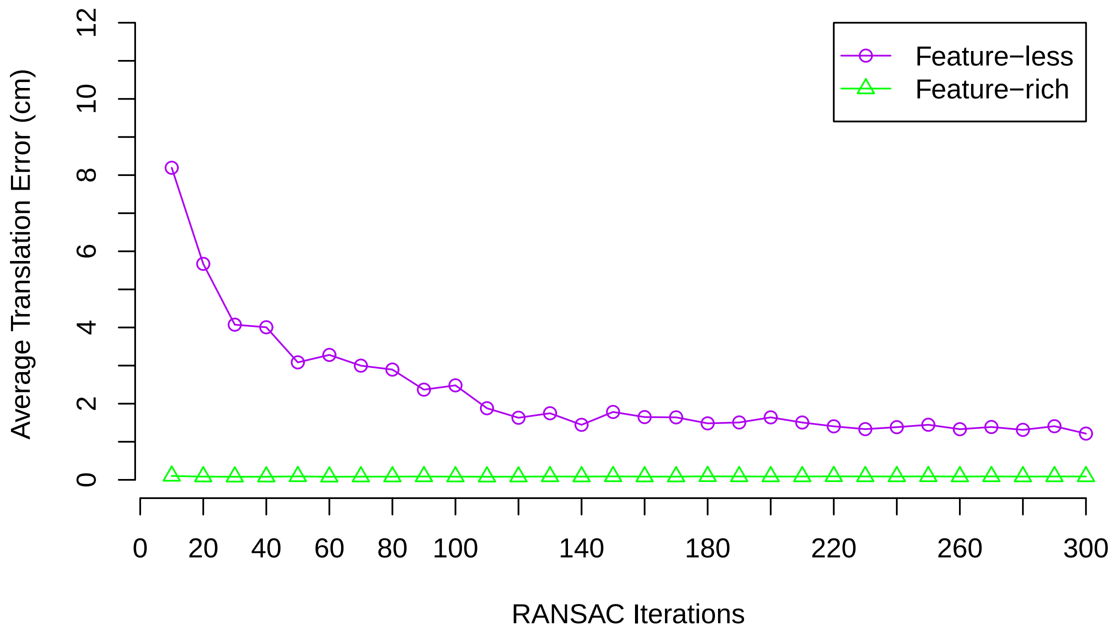


(a) Success rate

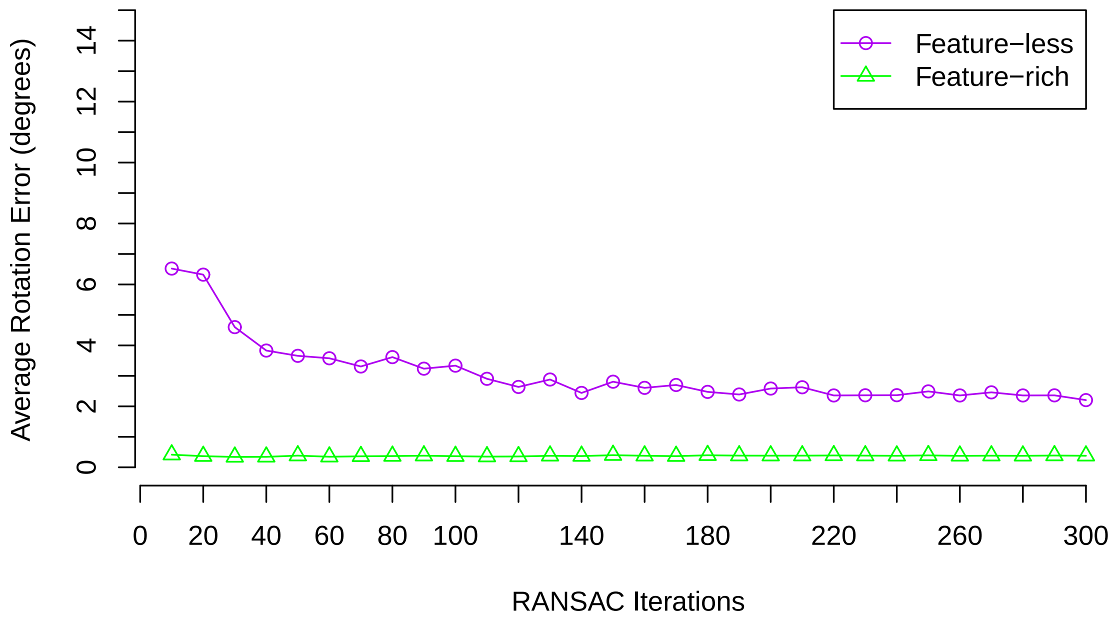


(b) Execution Time

Figure 5.6: Effects of increasing the number of RANSAC iterations on the success rate and the execution time of the Derees, compared between a feature-less and a feature-rich 3D image pair.



(a) Translation Error



(b) Rotation Error

Figure 5.7: Effects of increasing the number of RANSAC iterations on the accuracy of Derees, compared between a feature-less and a feature-rich 3D image pair.

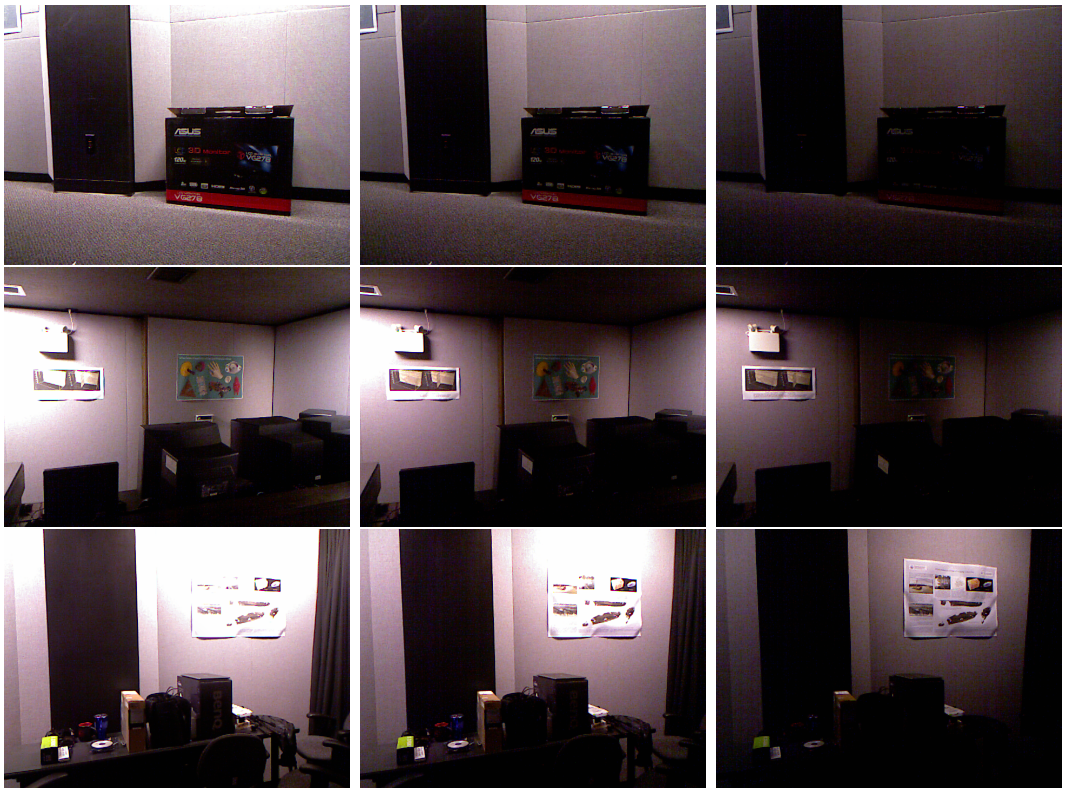
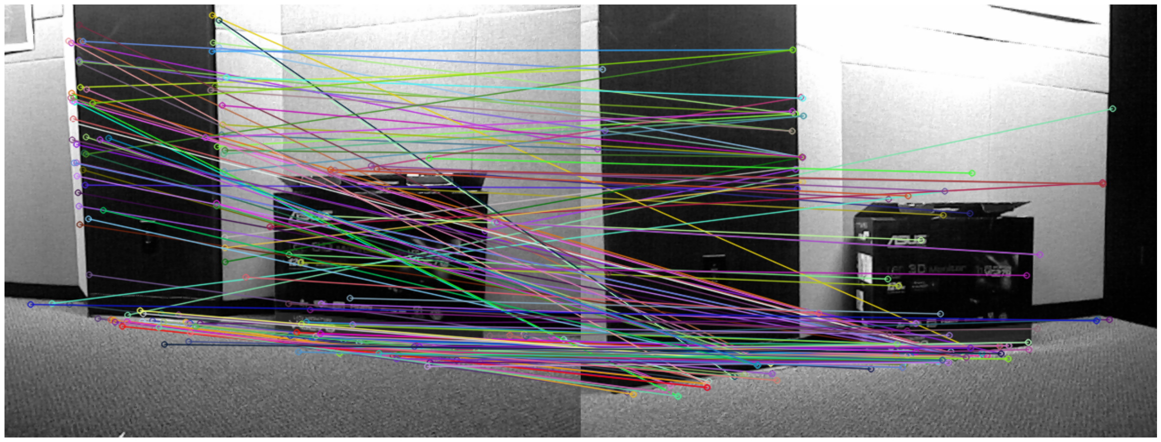


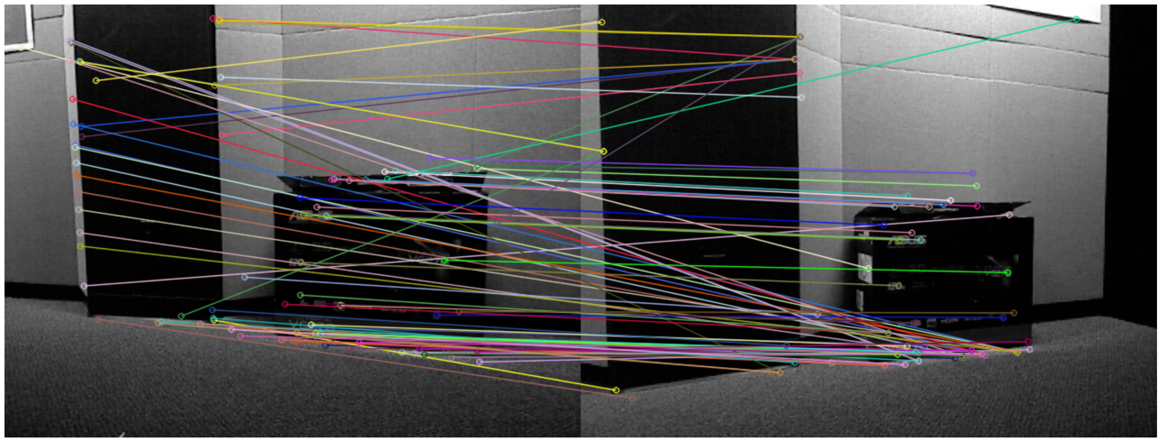
Figure 5.8: Scenes used to test for the different lighting conditions (complete set available at [56]).

this experiment, we rigorously defined unsuccessful registrations as registrations that are 10cm off in translation in any direction or 15 degrees off in rotation. We opted to set stricter success thresholds to emphasize the distinction between the results of the algorithm in different conditions. The robustness, speed and accuracy of the algorithm in each lighting condition can be compared in figure 5.10 and figure 5.11, as well as the effect of different iterations on the results. The experiment is executed 100 times per RANSAC iteration number for each of the 9 image pairs. The results are averaged between the 3 scenes for each lighting condition.

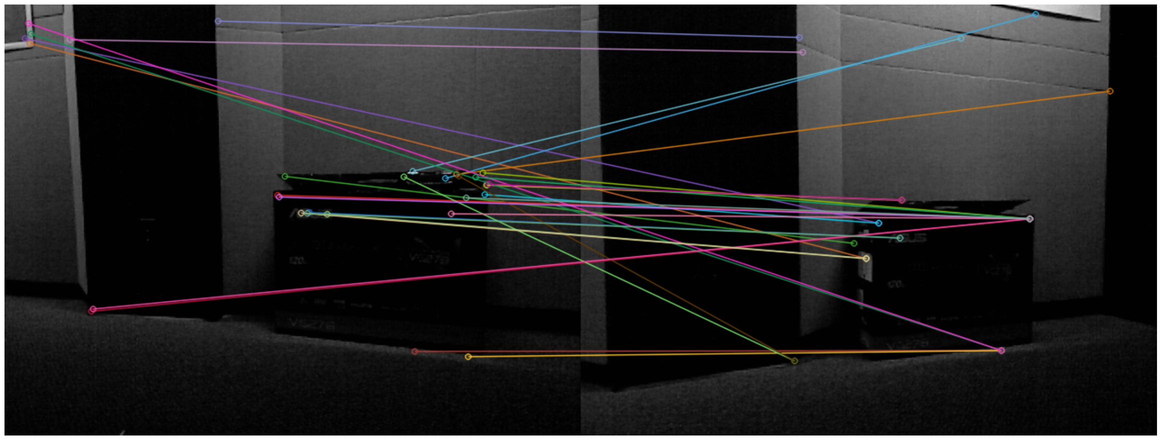
It can be noted that brightness does affect the accuracy of the algorithm in overall,



(a) Bright scene

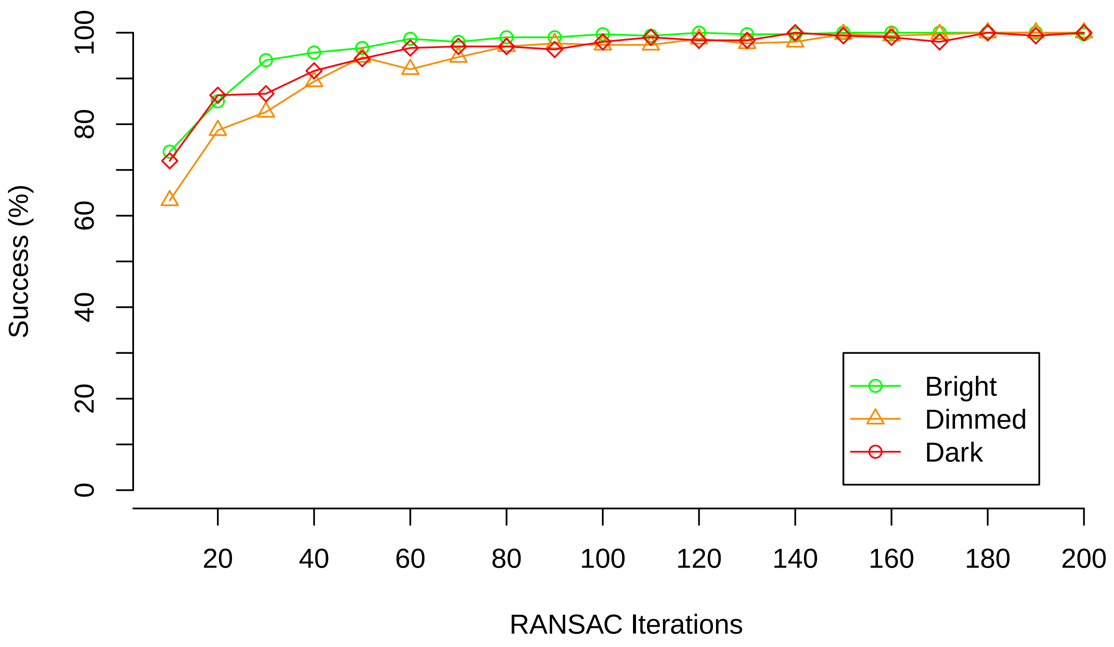


(b) Dimmed scene

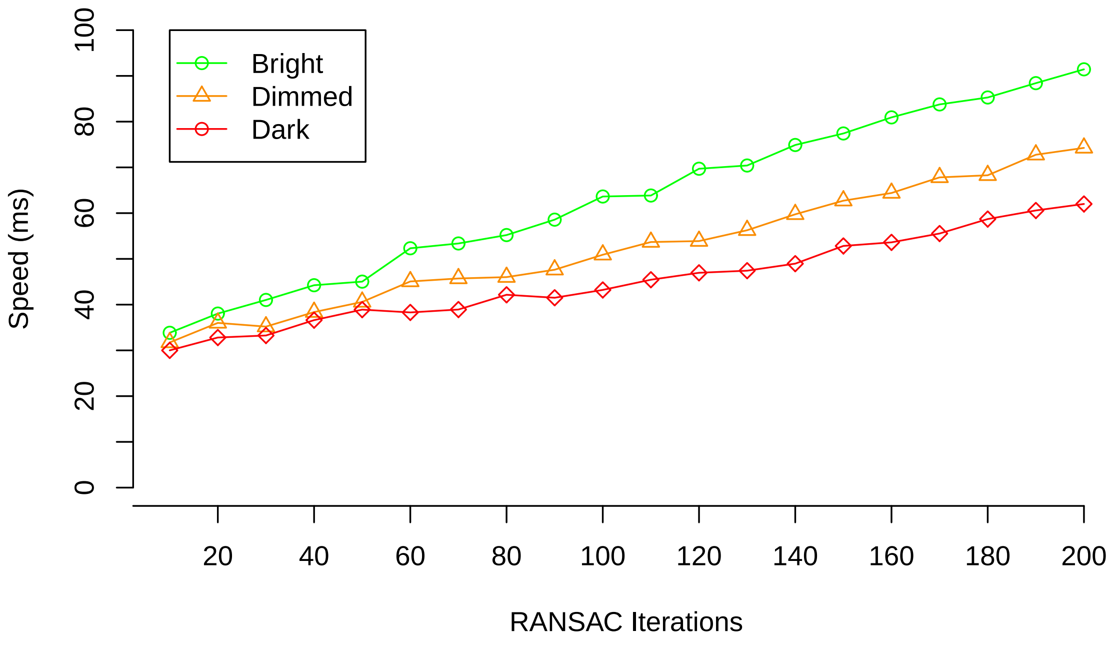


(c) Dark scene

Figure 5.9: The effect of lighting on the accuracy of the feature detection and matching algorithms.

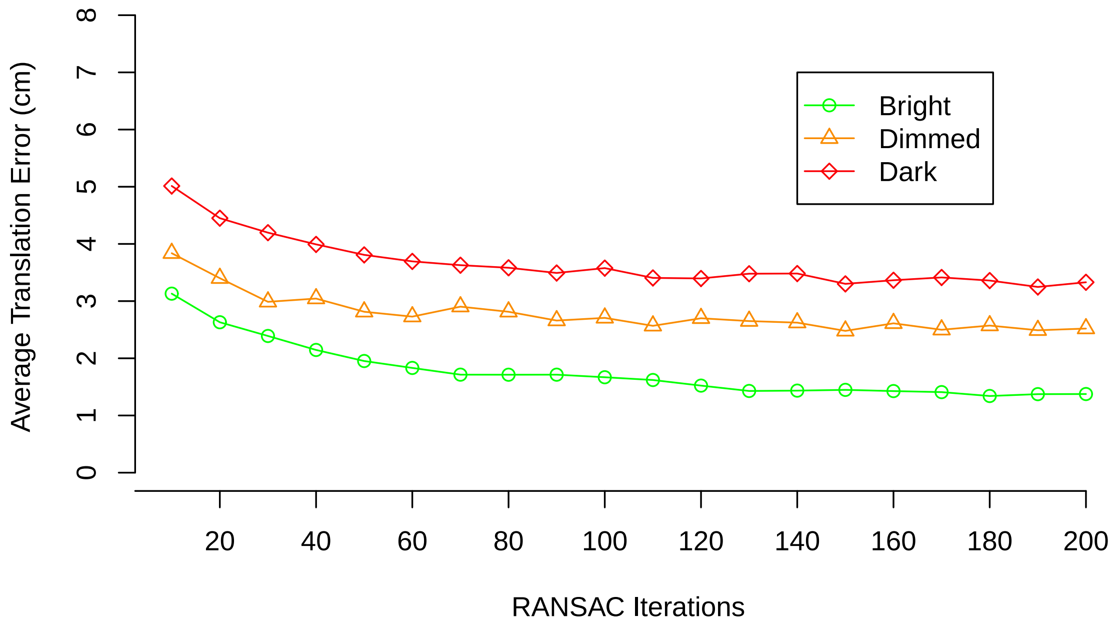


(a) Success Rate

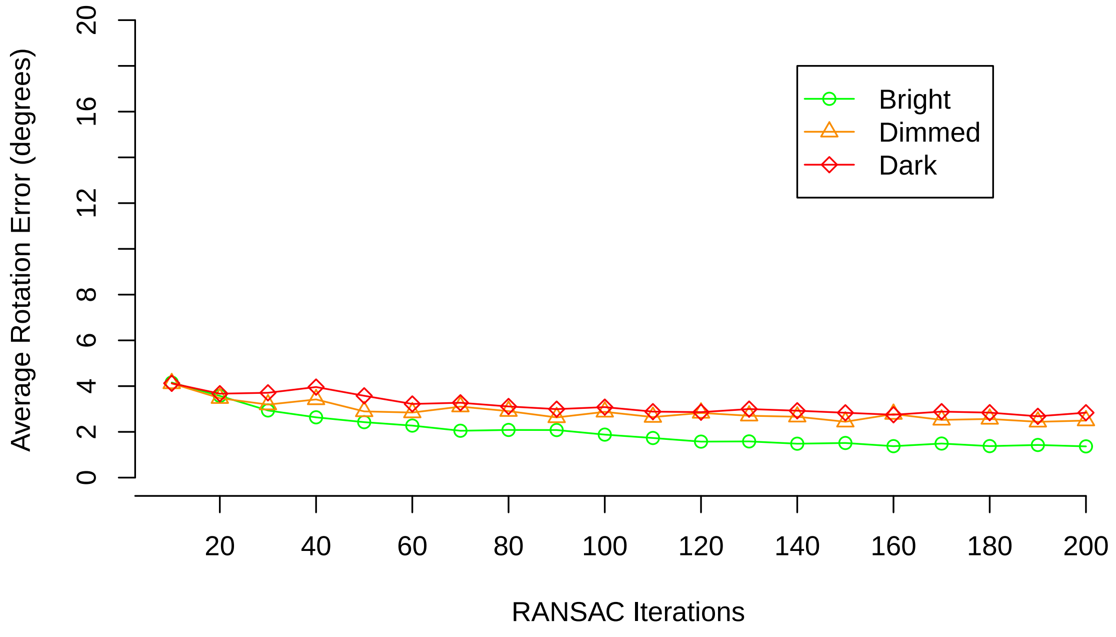


(b) Execution Time

Figure 5.10: The effect of brightness on success rate and speed of Derees.



(a) Translation Error



(b) Rotation Error

Figure 5.11: The effect of brightness on accuracy of Derees.

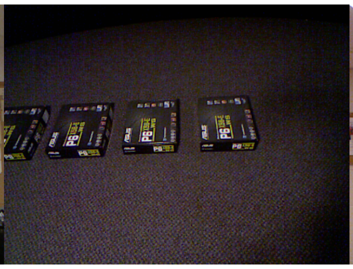
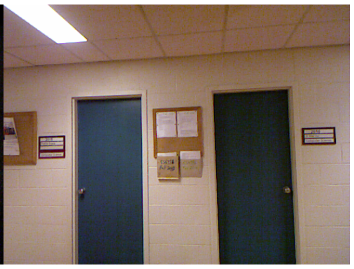
considering that the accuracy in low lit condition does not match the accuracy in well-lit condition. Although, this can be compensated to a great extent by increasing the number of RANSAC iterations. It is also notable that the speed of the algorithm is slower in well-lit condition, due to processing a larger number of feature pairs.

5.3.5 Registration of scenes with repetitive patterns

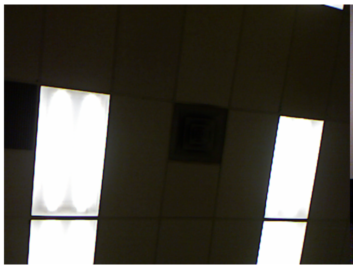
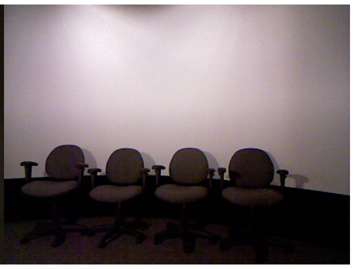
Since our registration method is feature-based, it is highly dependent on the success of the RGB feature detection and matching. Feature matching accuracy is compromised when elements in the scene are visually similar, causing the matching algorithm to match unrelated features of the scene together. This is especially true for human made environments which usually contain similarly manufactured items such as tiles, desks, etc. This negatively affects the accuracy of feature-based registration methods.

In this experiment, we evaluate the robustness of our algorithm with scenes that contain repetitive objects and visual patterns. We executed our algorithm with different numbers of RANSAC iterations over 5 image pairs (Figure 5.12) with 100 iterations per image pair. In this experiment, we rigorously defined unsuccessful registrations as registrations that are $10cm$ off in translation in any direction or 15 degrees off in rotation. Figure 5.13 depicts the success ratio of each of the 5 pairs, over different numbers of iterations.

It is immediately evident that our algorithm is not reliable in scenes with repetitive patterns. What is interesting is the different behaviours present regarding such scenes. The algorithm performs fine in scenes number 1 and 2. In scene 3, a moderate number of iterations (80-100) compensates the lack of accuracy. In contrast, the algorithm

(a) Scene 1 (b) Scene 2 (c) Scene 3

(d) Scene 4 (e) Scene 5

Figure 5.12: Scenes with repetitive patterns.

performs poorly in scene 4 and 5 and the number of iterations does not improve the results noticeably.

In a closer look, while scene 2 contains 4 supposedly indistinguishable objects specifically chosen to mislead the algorithm, the algorithm performs perfectly with even moderate number of RANSAC iterations. From the 217 2D features in this scene, 37% of the feature pairs are true pairs based on the best score from successful registrations, indicating the satisfying accuracy of the feature matching algorithm. Whereas in scene 5, only 6.8% of the feature pairs are true pairs. Looking back at the image pairs (figure 5.12), it can be noted while all scenes do contain repetitive patterns, the algorithm only performs poorly with scenes that contain less complexity and less variety of colour (scenes 4 and 5), which goes back to the effect of feature-richness of the scene on the feature detector and matcher. Figure 5.14 compares the

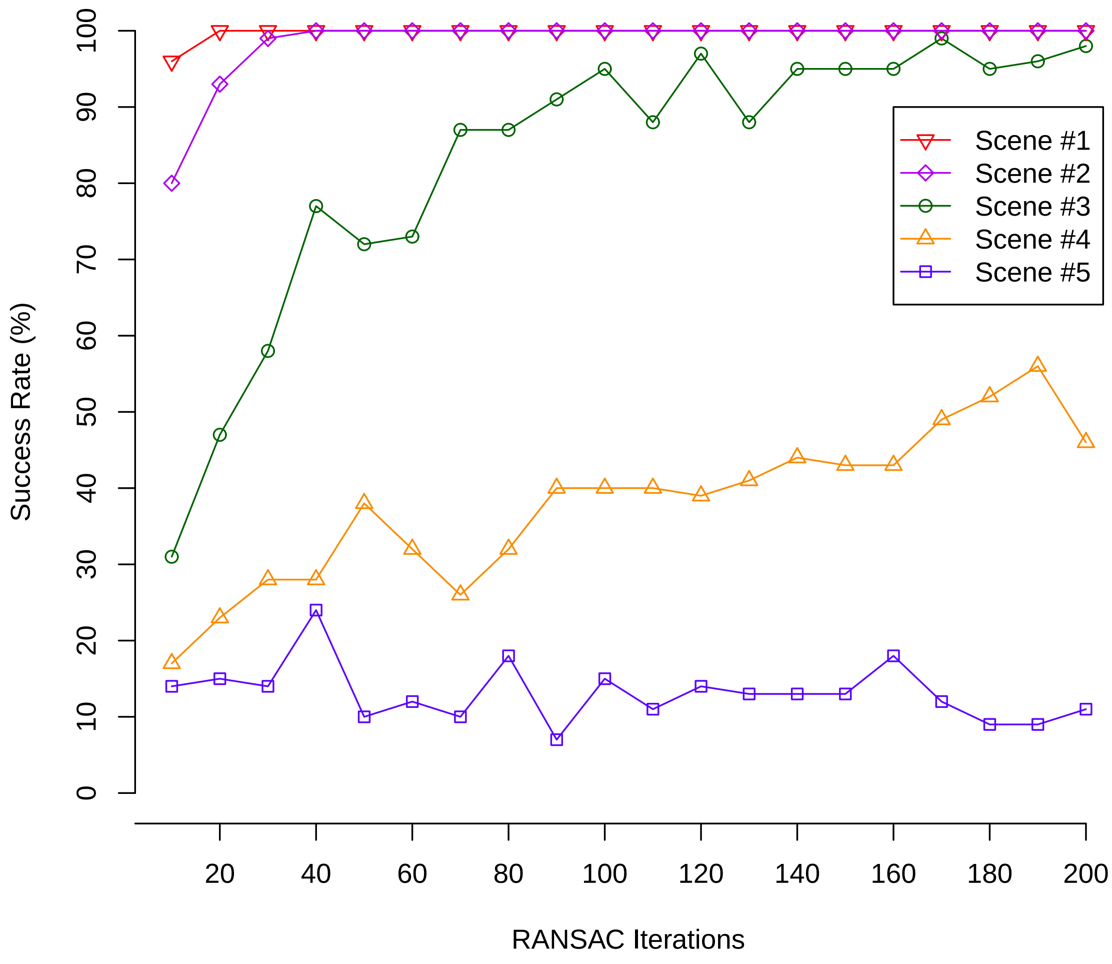
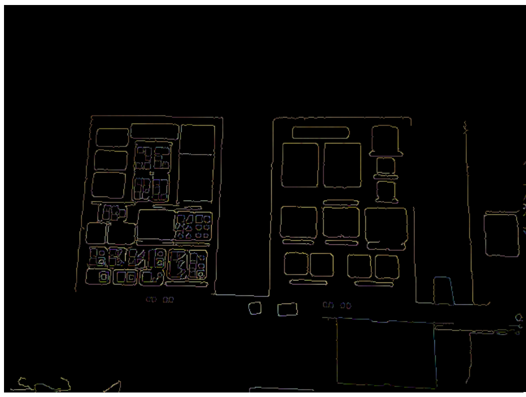
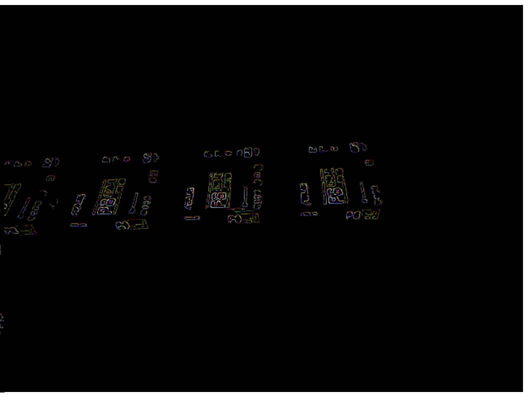


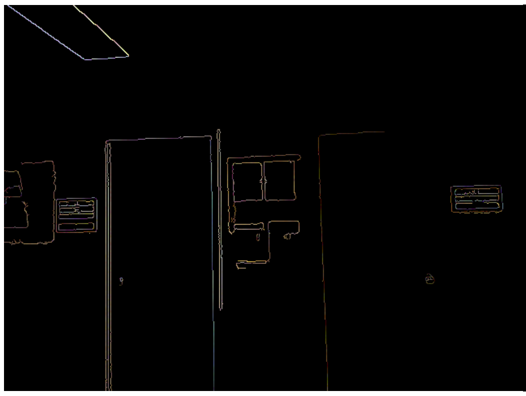
Figure 5.13: The robustness of the algorithm in scenes with repetitive patterns.

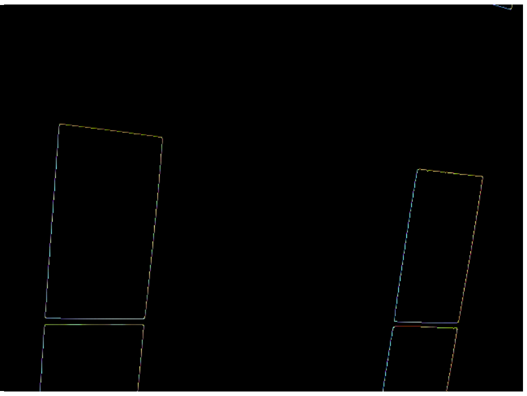
results of Canny edge detection [11] for all 5 scenes. The noticeable difference in complexity of the last two scenes is vivid in these figures.

To further explore the effect of repetitive patterns on the success of the algorithm, we executed the experiment with the ORB feature detector (Figure 5.15) instead of the SURF feature detector which has been used for all other experiments. It is noticed that the robustness does not change for the feature-less scenes, but it does affect other scenes positively, which suggests that the accumulation of feature matching results from multiple feature detectors might improve the robustness of the algorithm in some cases.


(a) Scene 1


(b) Scene 2


(c) Scene 3


(d) Scene 4

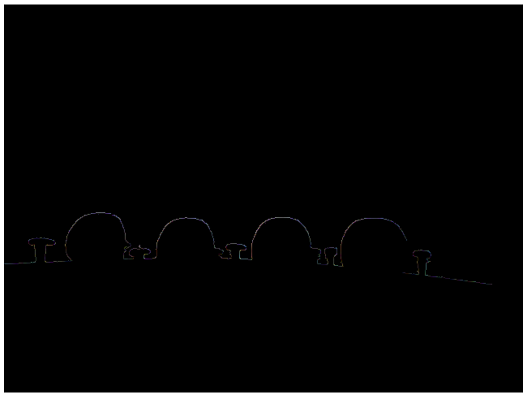

(e) Scene 5

Figure 5.14: Canny edge detection results for scenes with repetitive patterns.

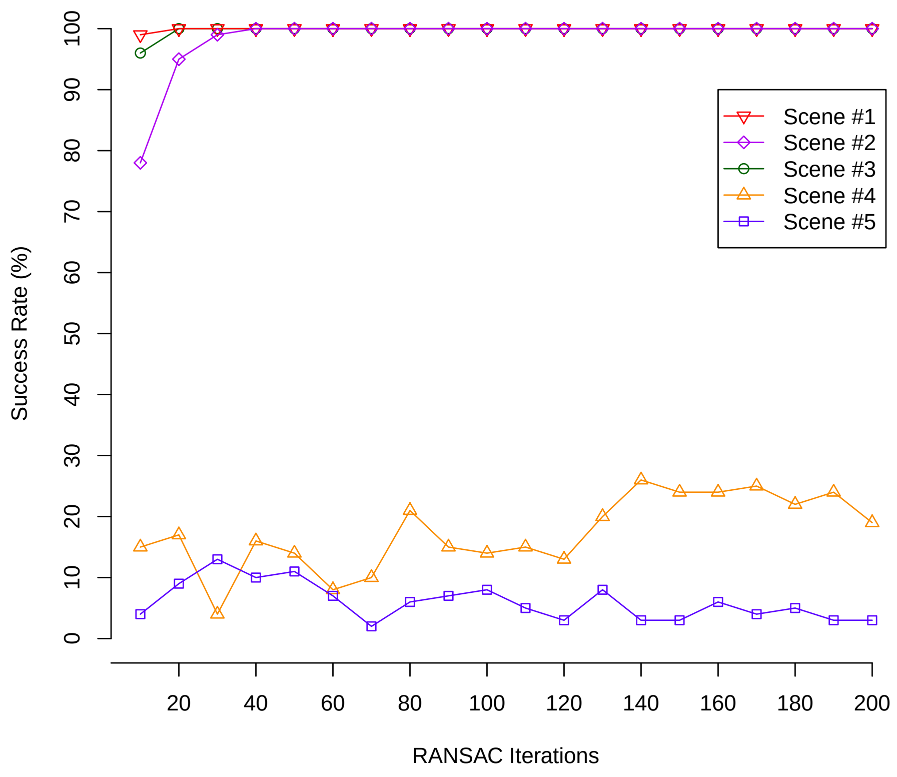


Figure 5.15: The robustness of the algorithm in scenes with repetitive patterns using the ORB feature detector.

Chapter 6

Conclusions

6.1 Contributions

In this research, we introduced a new registration algorithm (DeReEs) based on 2D RGB features and depth information. This technique is aimed at scenarios that require the input of multiple depth cameras in a scene to be aligned and merged together, reducing the burden of set up (calibration and/or manual alignment) or misalignment in case of accidental camera displacements. It can also be used with single depth cameras in static scenes. Our application of interest for this algorithm is a virtual reality collaboration tool with multiple cameras setup.

For our experiments, we collected 144 images from 45 indoor scenes. The images are taken under different conditions (lighting, overlap, etc.) to evaluate the overall robustness of the algorithm. Our dataset is available to the research community at [56] for comparison and experimentation with registration algorithms.

Based on our experiments, the algorithm runs in real-time (27 fps) with success rate of 97% under regular conditions with less than $3cm$ of translation error and 3 degrees of rotation error. DeReEs support transformations with 6 degrees of freedom and performs well with partially overlapping image pairs, supporting successful registration of images with only 23% overlap.

Similar to other visual feature based registration techniques, this algorithm is dependent on RGB information which is provided by some of the commercialized depth sensing cameras such as Microsoft Kinect. This dependency threatens the performance of the algorithm in not ideal scene conditions such as scenes with low brightness, repetitive patterns and feature-less scenes. We experimented with each of these situations, finding that the algorithm performs well with more conservative configuration in return of minimal speed reduction, except for scenes that include repetitive and feature-poor elements at the same time. Setting the false correspondence rejection component of the algorithm to execute large numbers of iterations, success rate of 90% and more can be achieved in feature-less scenes (30 fps) and low lit environments (16 fps).

6.2 Publications

Parts of this work have been published in two scientific publications: [57] and [59].

6.3 Limitations

Inherently, the fundamental limitation of this technique is its dependency on colour information. The execution of the algorithm requires having coloured point clouds and RGB enabled depth sensors which is not the case for some devices. Furthermore, the complexity of the RGB information plays an important role in the accuracy and robustness of the registration. For most indoor scenes, there is enough colour disparity and complexity that the feature detector and matcher can successfully determine true pairs with a reasonable true pair ratio. While large number of RANSAC iterations for false pair rejection can help the algorithm compensate for low true pair ratio in low lit, feature-less or repetitive scenes, the algorithm fails to robustly provide successful results with relatively difficult scenes that present all these unfavoured conditions. This places DeReEs at a disadvantage in scenarios such as autonomous robotics for underground mines or autonomous underwater vehicles in deep water which mainly consist of low-lit and feature-less scenes [37][32].

The other main limitation of our algorithm along with other feature-based registration methods is in scenarios where the images are taken at different times and elements of the scene are moved between the two images. In such dynamic scenes, if elements that contain large number of features are misplaced, feature-based registration algorithms suggest transformations that would try to align the misplaced elements in accordance with the feature pairings. This makes this registration algorithm not the best choice for robot localization in dynamic scenes which registers images from one camera over time.

The performance of DeReEs with outdoor scenes is an unexplored matter. Depth sensors such as Microsoft Kinect which are used in this research are designed for indoor environments and can only estimate depth confidently for objects between 0.5 to 10 meters away. In outdoor environments where the distance of points to the camera are far greater, it is expected for the feature pairs to have less accuracy since the real-world distance between points is larger than in indoor scenes. The effect of such inaccuracy of the overall accuracy of the registration is unknown.

6.4 Possible Improvements

While the algorithm is not designed for a specific depth camera, we have only tested it with Microsoft Xbox 360 Kinect camera. It is expected that DeReEs would perform similarly with other short range RGBD cameras such as Creative Senz3D or Google's Tango project devices. It would be valuable to extend the underlying software to support these cameras.

To further enhance the feature detector and matching results without largely affecting the performance speed, it is possible to accumulate results from multiple feature detection and matching algorithms; as different detection methods perform better in different scenarios.

To enhance the performance of the algorithm in the random selection stage, it is possible to reject many of the randomly selected pairs prior to calculating the transformation score which is computationally complex. Based on simple conditions or metrics such as triangular conformity between the 3 pairs (explained in details in subsection 4.2.2), improper pairs can be rejected.

The transformation scoring metric used with our algorithm is numerically discrete since it counts the number of closely placed pairs after the proposed transformation. This results in multiple proposed transformations returning the same score, while some of them might be more accurate (better coarse transformation). Converting our metric to a non-discrete metric can further enhance the transformation scoring.

This can be achieved by calculating the scoring metric based on the distance of corresponding pairs for pairs which have a distance less than the distance threshold d_T after transformation. For example, multiple transformations with the same number of accepted pairs can be further ranked based on the sum of absolute corresponding point distances.

At the transformation estimation stage, after the final transformation based on remaining true pairs, applying the ICP algorithm might provide more accurate results if non-overlapping portions did not exist in the scene, considering that the ICP algorithm takes all points of the cloud into account, minimizing the errors caused by a subset of points. To prevent the non-overlapping sections to mislead the ICP algorithm, it is possible to apply the transformation based on remaining true pairs and then ignore the points in non-overlapping parts and edges for the ICP refinement. This will also limit the number of points that are passed to the ICP algorithm. In a scenario without ICP refinement, the confidence scores from the feature detection and matching algorithms can be used to assign weights to each corresponding pair for a more accurate transformation estimation.

Bibliography

[1] Velodyne hdl-64e s2 product datasheet, velodyne lidar. http://velodynelidar.com/lidar/products/brochure/HDL-64E\%20S2%20datasheet_2010_lowres.pdf. Accessed: 2014-08-20.

[2] 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.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, Sept. 1975.

[5] P. Besl and N. D. McKay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, Feb 1992.

[6] P. Biber and W. Strasser. The normal distributions transform: a new approach to laser scan matching. In *Intelligent Robots and Systems, 2003. (IROS 2003). Pro-*

ceedings. 2003 IEEE/RSJ International Conference on, volume 3, pages 2743–2748 vol.3, Oct 2003.

[7] G. Blais and M. Levine. Registering multiview range data to create 3d computer objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):820–824, Aug 1995.

[8] T. Botterill, R. Green, and S. Mills. Reconstructing partially visible models using stereo vision, structured light, and the g2o framework. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*, IVCNZ '12, pages 370–375, New York, NY, USA, 2012. ACM.

[9] G. Bradski. Open source computer vision (opencv) library. 2000.

[10] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim. Shake'n'sense: Reducing interference for overlapping structured light depth cameras. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1933–1936, New York, NY, USA, 2012. ACM.

[11] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, Nov 1986.

[12] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145 – 155, 1992. Range Image Understanding.

[13] D. Cole and P. Newman. Using laser range data for 3d slam in outdoor environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1556–1563, May 2006.

[14] Y. Cui, S. Schuon, S. Thrun, D. Stricker, and C. Theobalt. Algorithms for 3d shape scanning with a depth camera. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(5):1039–1050, May 2013.

[15] H. Deng, W. Zhang, E. Mortensen, T. Dietterich, and L. Shapiro. Principal curvature-based region detector for object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[16] T. K. Dey, G. Li, and J. Sun. Normal estimation for point clouds: A comparison study for a voronoi based method. In *Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics*, SPBG'05, pages 39–46, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[17] J. S. Dhillon, C. Ramos, B. C. Wünsche, and C. Lutteroth. Leveraging consumer sensing devices for telehealth. In *Proceedings of the 13th International Conference of the NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction*, CHINZ '12, pages 29–35, New York, NY, USA, 2012. ACM.

[18] C. Dorai, G. Wang, A. Jain, and C. Mercer. Registration and integration of multiple object views for 3d model construction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):83–89, Jan 1998.

[19] H. Du, P. Henry, X. Ren, M. Cheng, D. B. Goldman, S. M. Seitz, and D. Fox. Interactive 3d modeling of indoor environments with a consumer depth camera. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, pages 75–84, New York, NY, USA, 2011. ACM.

[20] D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-d rigid body transformations: A comparison of four major algorithms. *Mach. Vision Appl.*, 9(5-6):272–290, Mar. 1997.

[21] M. Firman. List of rgbd datasets. `http://www0.cs.ucl.ac.uk/staff/M.Firman/RGBDdatasets/`. Accessed: 2015-04-08.

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

[23] D. Fofi, T. Sliwa, and Y. Voisin. A comparative survey on invisible structured light. volume 5303, pages 90–98, 2004.

[24] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361, June 2012.

[25] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi. Real-time rgb-d camera relocalization. In *International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, October 2013.

[26] G. Godin, M. Rioux, and R. Baribeau. Three-dimensional registration using range and intensity information. In *Proceedings of SPIE*, volume 2350, pages 279–290, 1994.

[27] D. Gossow, D. Weikersdorfer, and M. Beetz. Distinctive texture features from perspective-invariant keypoints. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2764–2767, Nov 2012.

[28] A. Handa, T. Whelan, J. McDonald, and A. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1524–1531, May 2014.

[29] A. J. Hanson. *Visualizing Quaternions (The Morgan Kaufmann Series in Interactive 3D Technology).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[30] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In O. Khatib, V. Kumar, and G. Sukhatme, editors, *Experimental Robotics*, volume 79 of *Springer Tracts in Advanced Robotics*, pages 477–491. Springer Berlin Heidelberg, 2014.

[31] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, USA, 2011. ACM.

[32] A. Kim and R. Eustice. Real-time visual slam for autonomous underwater hull inspection using visual saliency. *Robotics, IEEE Transactions on*, 29(3):719–733, June 2013.

[33] R. P. I. I. P. Laboratory and D. Meagher. *Octree Encoding: a New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer*. 1980.

[34] H. Lategahn, A. Geiger, and B. Kitt. Visual slam for autonomous ground vehicles. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1732–1737, May 2011.

[35] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 131–144, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[36] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004.

[37] M. Magnusson, A. Lilienthal, and T. Duckett. Scan registration for autonomous mining vehicles using 3d-ndt. *Journal of Field Robotics*, pages 803–827, 2007.

[38] S. Malassiotis, N. Aifanti, and M. Strintzis. A gesture recognition system using 3d data. In *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pages 190–193, 2002.

[39] Z. Marton, R. Rusu, and M. Beetz. On fast surface reconstruction methods for large and noisy point clouds. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3218–3223, May 2009.

[40] T. Masuda, K. Sakaue, and N. Yokoya. Registration and integration of multiple range images for 3-d model construction. In *Proceedings of the 1996 International Conference on Pattern Recognition (ICPR '96) Volume I - Volume 7270*, ICPR '96, pages 879–, Washington, DC, USA, 1996. IEEE Computer Society.

[41] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *In proceedings of British Machine Vision Conference (BVMC)*, pages 1–10, 2002.

[42] J. Matas, C. Galambos, and J. Kittler. Robust detection of lines using the progressive probabilistic Hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000.

[43] S. Meister, S. Izadi, P. Kohli, M. Hammerle, C. Rother, and D. Kondermann. When can we use kinectfusion for ground truth acquisition? In *Workshop on Color-Depth Camera Fusion in Robotics, IROS, 2012*, 2012.

[44] O. Miksik and K. Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2681–2684, Nov 2012.

[45] F. Moosmann and C. Stiller. Velodyne slam. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 393–398, June 2011.

[46] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[47] P. Neugebauer. Geometrical cloning of 3d objects via simultaneous registration of multiple range images. In *Shape Modeling and Applications, 1997. Proceedings., 1997 International Conference on*, pages 130–139, Mar 1997.

[48] Y. Park, V. Lepetit, and W. Woo. Multiple 3d object tracking for augmented reality. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 117–120, Washington, DC, USA, 2008. IEEE Computer Society.

[49] F. Pomerleau, S. Magnenat, F. Colas, M. Liu, and R. Siegwart. Tracking a depth camera: Parameter exploration for fast icp. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3824–3829, Sept 2011.

[50] K. Pulli. Multiview registration for large data sets. In *3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on*, pages 160–168, 1999.

[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, Nov 2011.

[52] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling*, 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] F. Sadlo, T. Weyrich, R. Peikert, and M. Gross. A practical structured light acquisition system for point-based geometry and texture. In *Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics*, SPBG'05, pages 89–98, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[55] J. Saez, A. Hogue, F. Escolano, and M. Jenkin. Underwater 3d slam through entropy minimization. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3562–3567, May 2006.

[56] A. R. Sahand Seifi, Oscar Meruvia-Pastor. Derees supplementary materials. `http://www.cs.mun.ca/~omeruvia/research/research.html`. Accessed: 2014-08-20.

[57] O. M.-P. Sahand Seifi, Afsaneh Rafighi. Real-time registration of highly variant colour+depth image pairs (poster). `http://www.cs.mcgill.ca/~kry/gi2014/GI2014PosterProceedings.pdf`, 2014. Poster presented at Graphics Interface 2014 conference, May 7-9, Montreal, Quebec, Canada.

[58] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, Apr. 2002.

[59] 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 '14, pages 136–141, New York, NY, USA, 2014. ACM.

[60] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[61] H. Sunyoto, W. Van der Mark, and D. Gavrila. A comparative study of fast dense stereo vision algorithms. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 319–324, June 2004.

[62] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[63] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan. Scanning 3d full human bodies using kinects. *Visualization and Computer Graphics, IEEE Transactions on*, 18(4):643–650, April 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*, SIGGRAPH '94, pages 311–318, New York, NY, USA, 1994. ACM.

[65] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177–280, July 2008.

[66] R. Vasudevan, G. Kurillo, E. Lobaton, T. Bernardin, O. Kreylos, R. Bajcsy, and K. Nahrstedt. High-quality visualization for geographically distributed 3-d teleimmersive applications. *Multimedia, IEEE Transactions on*, 13(3):573–584, June 2011.

[67] K. Wang and Q. Yu. Accurate 3d object measurement and inspection using structured light systems. In *Proceedings of the 12th International Conference on Computer Systems and Technologies*, CompSysTech '11, pages 221–227, New York, NY, USA, 2011. ACM.

[68] S. Weik. Registration of 3-d partial surface models using luminance and depth information. In *3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in*, pages 93–100, May 1997.

[69] L. Xia, C.-C. Chen, and J. Aggarwal. Human detection using depth information by kinect. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 15–22, June 2011.

[70] B. Zeisl, K. Koser, and M. Pollefeys. Automatic registration of rgb-d scans via salient directions. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2808–2815. IEEE, 2013.

[71] Q.-Y. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Trans. Graph.*, 32(4):112:1–112:8, July 2013.

[72] B. Zitov and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977 – 1000, 2003.

[73] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 371–378, New York, NY, USA, 2001. ACM.