IN ORCHARD 3D SENSING FOR CROP IDENTIFICATION AND LOCALIZATION IN A VIRTUAL ENVIRONMENT

A thesis presented in partial fulfilment of the requirements for the degree of

Master of Engineering

in

School of Engineering

at Massey University, Palmerston North,

New Zealand.

Brett Maurer

2024

Abstract

Due to recent events of the global pandemic, as well as decreasing labour available in the agricultural sector in New Zealand, there has been an overall labour shortage in this multi-million dollar industry. With the advancement and accessibility of new technologies, this is an industry in which these can be applied effectively to increase efficiency and potentially be done remotely. This can be a key benefit for the industry, as it would help avoid labour shortages with the ability to undertake some orchard work remotely, as well as providing job opportunities to those who may not normally be able to undertake this sort of physical work. Another resulting benefit in having the ability to potentially work remotely means that the need for travel to and around the orchard is decreased, which would also have sustainability benefits. One such technology that has been looked at, with developments being made over the recent years is robotic harvesting. A key element in this process is the remote sensing and vision systems used to identify, and subsequently localise, the targeted crop to be harvested.

Currently, most of the research that has been done in this area has been using expensive, high precision 3D scanning tools. However, with the advancement in this area of technology, cheaper and more practical alternatives may be suitable for the application.

The primary focus of this project was to evaluate and quantify the accuracy and therefore suitability of multiple remote sensing systems to be used in an orchard environment. For this project, consumer grade RGBD(Red, Green, Blue, Depth)camera systems were used to identify and localise apples, with the resulting depth and image data being interacted with in a virtual reality environment in the form of point clouds

as well as mesh bodies, in order to identify and localise the apples in 3D space in a virtual environment.

Multiple metrics were evaluated for these systems including depth accuracy, point density, external factor effects such as sunlight, positional accuracy in 3D space and error in the localization of objects. After initial investigation, it was found that it would be possible to use these devices for this application, however refinements would need to be made. The depth data for both forms of 3D sensing was very accurate and effective in an indoor environment, with high depth precision and point density, however there was more variation when data was collected outdoors, mainly due to interference from external factors such as sunlight. As expected, point density decreased with increasing distance, resulting in an optimal operating range of 1-2m being established. The resultant point clouds collected were also used to interact in both a 2D and 3D environment, using CloudCompare and virtual reality, respectively, in order to quantify the error in manual localisation of objects in 3D space. The error in the manual localisation of objects scanned in the test orchard setup, in this case apples, was also quantified by comparing results from a purely virtual environment using virtual 3D scanned apples to the point cloud data collected from the orchard setup. An interesting finding from this was that most of the variation in results came from the error in manual localisation, rather than inaccurate depth data.

Overall, this project was successful in analysing and quantifying the characteristics of multiple remote sensing methods, finding that the concept would work for the application based on the results collected, but would benefit with future development and refinement in order to be commercially viable. Another key area that would need subsequent research and development is the data transfer between the environment being scanned, and the environment in which this data is interacted with, as to be practical for real-world use, this would need to be efficient and automated. With further development, filters could be applied to the data to more efficiently target the crop, with a suitable algorithm being applied to automatically identify and localize the crop, which could lead to increased efficiency, quality and accuracy of the ideal application

of robotic harvesting.

Acknowledgements

I would like to thank my supervisor Dr Gabe Redding for his support and advice throughout this project, providing valuable insight and experience without which this project would not be possible.

A special thanks also needs to be given to Dr Russell Wilson and Prof. Andrew Shilton, for providing support and encouragement throughout and especially in the final stages of the project.

I would also like to acknowledge Rob Elstone and Hortworx for providing the opportunity to work with in the development of this project.

Thanks must also be given to Prof. Andrew East and Prof. Johan Potgieter as well as the rest of the staff and students at MAF Digital Lab for their help, support and providing access to resources throughout this project

A very special final thanks must be given to my family, Kevin, Lorna, and Rhys, as well as my partner Jess, for their help, support and love throughout this project.

Table of Contents

List of Tables

List of Figures

Introduction

Agriculture and horticulture are key elements of the New Zealand economy, with the value of agricultural exports amounting to over 53 billion New Zealand dollars in the year of 2022 [1], with horticultural exports contributing 7.7 billion New Zealand dollars of this sum [1]. Due to multiple factors, one of which in recent years was COVID-19, there is a labour shortage in this employment sector [2]. With this problem facing the industry, alternative solutions need to be considered in order to overcome these issues and ensure stability for the future of this industry.

From the development of previous industries, we can take the lessons learnt and adapt the methods that they used to overcome similar issues that faced them. For example, the manufacturing industry has invested in the development of automation systems and robotics, including human robot collaboration, which greatly increase productivity, efficiency, accuracy whilst also improving upon worker safety [3]. With the improvement of these factors, the subsequent effect is an overall increase in profits.

In recent years developments have been made developing and incorporating both existing and emerging technologies into both the horticultural and agricultural sectors. Some examples of these applications can be seen in automated milking systems, automated feeding systems, including automated irrigation, as well as developments in crop surveying and robotic harvesting [4] just to name a few. Using developing technology from this sector as well as combining it with fast progressing technologies from other sectors such as manufacturing, specific tasks can be more readily approached with the goal of finding an effective solution through the application of these combinations of technology.

One such area that is an obvious opportunity to apply new methods and technology in order to increase efficiency, production yield, and accuracy is crop harvesting. Traditionally, most crops are set up and grown in a way that is conducive to be harvested by mechanical machinery all across the agricultural industry, ranging from crops like grain, to more fragile produce such as blueberries [5]. As these methods have been in use for years, the way that these crops are planted has stayed the same for decades, with the already existing machinery developed to work effectively in these environment. However, there are some types of crops that have traditionally not been grown in a way that is suitable for any sort of effective automated harvesting, be it mechanical or robotic. One such crop is apples. Traditionally, apples have been grown in orchards with relatively large spacings between trees and in general with regular tree shapes, with branches growing 360 degrees from the trunk of the tree in a conic shape [6]. Recently, effort has been put into the research and development of new crop growing systems in the apple industry, specifically in New Zealand with the development of the Future Orchard Production System orchards(FOPS) [7]. With these orchards, land area is maximised, greatly increasing yield per hectare whilst also being very suitable for the future application of automated systems, ranging from phenotyping to crop monitoring and automated harvesting. The premise of these FOPS orchards is similar to that of a vineyard, with the optimal spacing between rows being found to be 2.0m between rows, with a typical height of 3.0 m [6]. These dimensions allow optimal land usage whilst also ensuring that the apples receive the correct amount of sunlight and nutrients in order for them to grow efficiently and maintain their quality. An example of a FOPS orchard can be seen below in Figure 0.0.1. As well as providing optimal yield, this orchard system lends itself to the implementation of automated processes, as it is much more structured and regular than a traditional apple orchard setup.

With these orchards being more suitable for automated systems, advancements can be made in these areas. Instead of approaching the whole solution of automated harvesting, it needs to be broken down into multiple parts with each part being built upon in order to work in conjunction with one another with the end goal being automated

Figure 0.0.1: Example FOPS Orchard

harvesting. This can be broken down into three main categories, being:

- Vision Systems
- Robotic Harvesting System
- GPS and Mapping System

For the scope of this project, the main focus will be on the vision systems, as this is a vital element and stepping stone for the other considered components. While this is the primary focus of this research, the other categories also need to be kept in mind when considering this, as all the systems need to be able to work in conjunction with one another.

Over the years many types of vision systems have been developed, ranging from regular RGB (Red, Green, Blue) photography to Lidar (Light Detection and Ranging) and laser scanning systems. Some major hurdles in using these systems previously have been cost, accessibility and processing speed, with scans often needing significant post processing [8].

For this research, it was decided to use consumer RGBD (Red, Green, Blue, Depth) cameras, which are relatively low cost and operate using stereo vision RGB cameras in order to generate point cloud data. The main areas of focus for the research are to assess the suitability of the use of this type of camera in an in-orchard scenario, with

the end goal being the application of automated harvesting. While the application of automated harvesting is outside the scope of this project, components of this do need to be considered when assessing the viability of the vision systems, as elements such as accuracy relative to the size and position of apples are critical metrics.

The objective of this research is to assess the potential of RGBD cameras for integration into an automated or semi-automated harvesting system, including the assessment of the potential application of a virtual picking environment. Achieving these objectives will require assessment of the following;

- Accuracy of depth data from an RGBD camera
- Effects of an outdoor environment on an RGBD camera
- Effect of distance on point cloud density from RGBD camera data
- Error of point cloud interaction in a regular virtual environment
- Error of point cloud interaction in a virtual reality environment

Using the data collected and assessed from this research, this technology can then be applied in further automated harvesting applications knowing the accuracy and suitability of the vision systems that could potentially be used.

Review of Literature

Based on the scope of this project and the objectives established, the following topics need to be reviewed in order to establish the current state of the art in these areas, as well as to identify gaps in the research.

1.0 Types of 3D Sensing Techniques

In order to assess the most viable 3D sensing options for the research, the functionality of each type of 3D sensing needs to be identified, researched and evaluated in order to make an informed decision of the most suitable sensor for differing applications and conditions.

1.1 LiDAR and Time of Flight

Lidar, which stands for Light Detection and Ranging[9] is a form of 3D sensing that uses light in the form of a pulsed laser to measure ranges based on the period of time it takes for the reflected light to return to the receiver[10]. The first well known application of the technology was during the Apollo 15 mission, when a laser altimeter was used to scan the surface of the moon[11]. Since it's development, traditional uses of Lidar have been in surveying, geology, archaeology, seismology, forestry, often being used to create accurate 3D maps or environment reconstructions. In more recent times, Lidar has been used in vehicle applications such as object and vehicle sensing, which can be utilized in the application of autonoumous navigation[12]. The options for applications has expanded, as the accuracy and accessibility of Lidar technology has increased greatly

over the last few years.

Time of flight (ToF) depth sensing technology works by measuring the round trip time of a light signal provided by a laser projector[13]. The way that time of flight systems operate are similar to Lidar, however Lidar uses pulsed lasers to generate a point cloud, as opposed to a time of flight sensor which has a constant light source, with the reflected light being observed[14]. The phase shift between the illumination and reflection is measured and can then be translated into a distance measurement[14]. Time of flight sensors are used in applications such as advanced automotive functions, such as pedestrian safety and crash detection, as well as many autonomous vehicle sensing applications[15].

1.2 Structured Light

The principal of a structured light 3D scanner functions on a projection of light patterns and a corresponding camera that captures these images[16]. The light patterns are projected onto the target object, with the projection lines appearing distorted from any other angle other than that of the origin. This distortion is then captured by the camera and is subsequently used for geometric construction of the surface[17]. As well as being able to provide an accurate 3D representation of an object, because the structured light projection is analysed via a camera, this camera can take RGB (Red, Green, Blue) images that are easy to calibrate and align to the model. In turn, this means that structured light 3D scanners can apply accurate high definition RGB colours to the scanned model. Due to the nature of structured light 3D scanning, it lends itself to small scale static scanning, as any dynamic movement of the object, as well as interfering lighting conditions, would affect the quality of the scan. Typically structured light scanning is used for scanning objects accurately for simulations, scanning objects for 3D printing, as well as scanning objects to create a 3D model to be applied in a virtual environment, such as in video games[18].

1.3 Stereo Vision

Stereoscopic 3D sensing works using the same principles as stereoscopic vision, the most well known example being binocular vision, or stereopsis. Stereopsis is what gives an image depth [19]. The way that this is achieved is that each eye has a slightly different lateral position, meaning that the image seen by each eye differs slightly. The disparity between these two images is then processed, in the case of human binocular vision by the visual cortex in the brain, the result yielding depth perception [20]. This concept has been taken and applied to artificial technology to achieve similar results in the form of stereo cameras [21]. Stereo cameras use two lenses and take images from both, with the disparity between them giving the impression of depth [22]. More recently, this technology has been applied to RGB-D cameras, also refererred to as depth cameras. These sensors use both the depth information obtained through stereo vision as well as the RGB image information to create RGB-D images(Red, Green, Blue, Depth). Essentially, each RGB pixel also has a depth measurement associated with it. This data can then be processed to create a point cloud or similar 3D visualisation. This form of stereo vision is known as passive stereo, as opposed to active ranging methods such as active stereo. Active stereo incorporates the same stereo camera setup as passive stereo, however it also implements projection of structured light[23]. This provides a solution to the inherent problem of stereo vision in detecting smooth surfaces[23]. The same result can also be achieved using only one camera by matching the projected features on the surface to the features extracted from the acquired image through triangulation[24]. With active stereo vision, the process of matching the corresponding images becomes simpler as the pattern is more regular. Figure 1.0.1 below shows the matching criteria of active stereo vision.

With modern technology, consumer grade stereo vision systems have become more available, making it a viable option for uses such as industrial automation and 3D machine vision, an industry which is rapidly advancing and constantly developing.

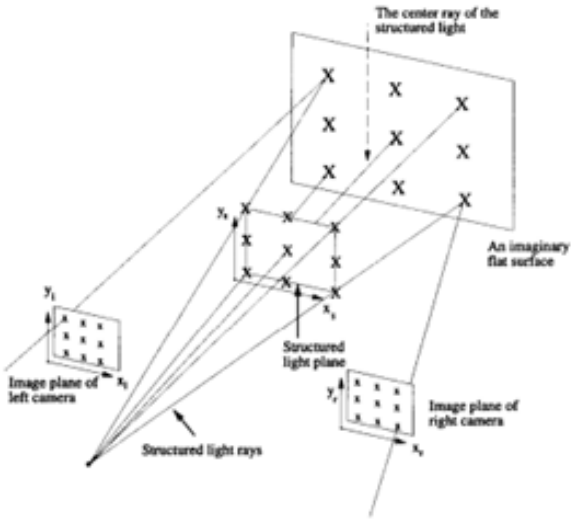


Figure 1.0.1: Diagram demonstrating the concept of active stereo vision systems[21].

2.0 Point Cloud Data Interaction

As part of the scope of research for this project, the interaction with the resultant data from the experimental testing also needs to be considered. In most methods of 3D sensing, as described above in Section 1.0, point clouds are the main form of the raw data collected. Described below are some of the previous examples and research conducted in the interaction with point cloud data, as well as potential methods that could be utilized in this application. With the availability of multiple 3D data visualization softwares, all options need to be considered and assessed in order to make an informed decision about the most suitable ones for the application of this project.

2.1 Meshlab

Meshlab is an open source software that is used for creating and processing 3D meshes [25]. As demonstrated in [26], a study was conducted on assessing the use of Meshlab as a complete tool for the integration of photos and colour with high resolution 3D geometry data. The specific area of interest in this study was to to be able to apply accurate colour information on the 3D meshes generated, as the target objects for this study were cultural heritage objects, meaning that having accurate colour information

is the best approach in order to describe the visual appearance of the objects.

Multiple methods and features of Meshlab were used to manipulate and edit the 3D scanned meshes in order to provide accurate colour information. One such method was weighted blending. Using a set of calibrated images, the colour information can be projected onto a 3D surface. However, the colour from different images will likely never be exactly the same, due to a variety of factors such as lighting inconsistencies and unstable camera settings [26]. This could be overcome by simply blending all the variations of colour at each point, however it was found that a better solution that involved blending based on quality. This method in Meshlab is able to calculate colour for any point over a 3D surface, by estimating the quality using a weighted mean. This quality is based on a variety of metrics, which include distance, viewing angle, and proximity of the pixel to critical points on the photo. Using these metrics, a quality score for each point is determined, that is subsequently used in the colour blending process. Due to the nature of the process, this method is well suited to datasets that use a large number of images, and also complex datasets and 3D models.

An alternative method that was researched [26], was photo stitching. This method is similar to the weighted blending described above, except instead of using a blended colour obtained from multiple images, it uses the most suitable single corresponding triangle of colour from a single image. Similar selection parameters as used in the weighted blending method were used. Once the most suitable colour from the corresponding photo is determined, this triangle of colour is then applied to the corresponding triangle on the 3D model. This process is undertaken for every surface of the 3D model. Further processing of lighting inconsistencies is then carried out by smoothing out edge inconsistencies between the triangular photo segment due to lighting inconsistencies. The advantage of this process is that it produces sharp colour mapping, as there is no blending between colours, however it is more applicable to applications where less photos are needed, in conjunction with a less complex 3D model. This method is not as suitable for complex models as the input dataset would need to be very large, and the texture would be very fragmented. As a result, Meshlab may struggle to process

models consisting of over 2 million triangles. It was found that it was a very valuable software in the accurate reproduction of cultural heritage objects, and would be very suitable for other similar applications. In this study [26], it is stated that in the future an integrated multi purpose tool that allows the user to accurately reconstruct colours onto a mesh surface should be developed, and would be possible sue to the open source nature of Meshlab.

2.2 CloudCompare

Another widely used software in the 3D visualisation and point cloud processing software[27]. As well as being able to process point cloud data, it can also compute and process triangular meshes, as well as calibrated images [28]. Based on [29], CloudCompare can effectively be used for the processing of raw point cloud data in order for it to be used in a functional form. These basic but powerful functions include cropping of point clouds in multiple planes, as well as assessing characteristics and quality metrics of point cloud data[30]. Due to the usability, functionality across multiple operating systems such as Windows, Linux and macOS[31], and open source nature of the software, it makes it a very attractive package for the use of processing point clouds for a variety of applications and industries[27]. As seen in [32], CloudCompare was used in conjunction with other software in order to obtain usable point cloud data for the application of 3D earth working. The ultimate goal of precise 3D mapping of earth work and construction sites is to be able to facilitate the operation of unmanned equipment[32]. Multiple techniques were used to gather the point cloud data, including the use of both Lidar as well as photogrammetry via the use of unmanned aerial vehicles(UAVs). This data was then processed using CloudCompare before being evaluated using an analytical comparison in order to verify the accuracy of the Lidar-Photogrammetry hybrid data, with the conclusion being made that this form of data would be more accurate than either on their own. CloudCompare was also utilised in a study[33] investigating the acquisition of point cloud data via the use of UAVs. In this study, a UAV was used in conjunction with a laser scanning range finder and RGB camera to scan a target

object, specifically a bridge in this study. In a flight time of 10 minutes, approximately 20,000 scan lines were recorded. As part of the evaluation, CloudCompare was used to calculate the distance between scans from a terrestrial laser scanner and the laser scanning range finder used on the UAV. The results from this can be seen below in Figure 2.0.1. Overall it was found that the use of CloudCompare to assess the variation between scans provided a very good estimate of the accuracy potential. Using similar methodology, this concept could be applied and modified to be used in a wide variety of point cloud data processing and comparisons.

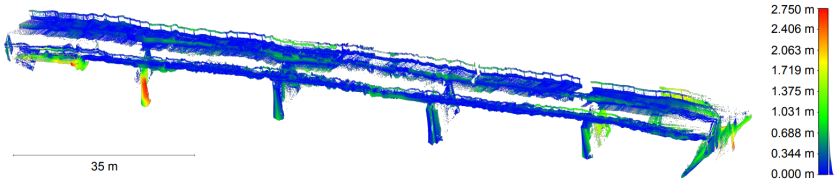


Figure 2.0.1: Colour coded distances between LSRF and TLS point clouds calculated using CloudCompare[33].

Overall it was found that the use of CloudCompare to assess the variation between scans provided a very good estimate of the accuracy potential. Using similar methodology, this concept could be applied and modified to be used in a wide variety of point cloud data processing and comparisons.

3.0 3D Sensing Techniques in Agricultural Applications

Since the introduction and development of these 3D sensing technologies, trials and studies have also been conducted in the agricultural space in order to assess and evaluate the suitability of such systems for applications in this area. One such research project [8], was conducted by using Lidar and RTK-GNSS together with the goal of mapping orchards. Using Lidar is not dependant on intrinsic restraints such as variable lighting conditions in order to identify fruit. Experimental testing was undertaken using a mobile terrestrial laser scanner(MTLS) consisting of a Lidar sensor synchronised with an RTK-GNSS satellite navigation receiver. The theory that was used in

[8],was that apples have a different reflectance value compared to their surroundings, such as branches and leaves. The Lidar system generates a point cloud with (x, y, z, R) values, giving each point a 3D coordinate as well as a reflectance value, R. Once the raw point cloud data was collected, it was then processed to generate a usable model. A rigid transformation was performed by applying both a rotation and translation matrix to each point, meaning the point cloud generated would be with absolute coordinates. This resulting point cloud was then manually labelled in order to give the ground truth location of the apples. Once the point cloud was generated, an apple detection algorithm could be implemented. The first step of this algorithm was point cloud segmentation, which was achieved by removing points from the point cloud which had reflectance values that did not correspond to the reflectance threshold, RTH, of apples. Outlier removal was then applied to the remaining points in order to delete noisy points that may otherwise connect clusters from different apples. Even after this point removal, some groups of apples touching resulted in clusters of more than one apple. These clusters were then split into sub clusters, each containing one apple. This was done by predicting the number of apples in each cluster, with the cluster then being split using the K-means algorithm. To predict the K number for each cluster, the author tested three different methods. The first approach was inspired by a template matching technique [34] . This technique projects the point clouds of each cluster onto a 2D plane. After processing using a Gaussian filter, the predicted K number corresponds to the maximum number of apple centres found, each centre being a local maxima. To ensure the best prediction, each cluster was projected onto four planes from different perspectives. The second method used to predict the K number applies a decision tree based on cluster features, such as volume(V), cluster points(P), mean reflectance of cluster points($\overline{R}$), and a geometric parameter computed as the product of the normalised eigenvalues. The decision tree then uses these features to predict the number of apples in the cluster. The third method used uses a combination of the previous two methods outlined, by first distinguishing between single and multi-apple clusters using the second method, and then applying the first template matching

technique to the clusters containing more than one apple to determine the K number. Overall it was established that using this method of Lidar reflectance in conjunction with the methods used to predict the K value were very effective, with the decision tree method being the most effective with a localization success of 87.5%, identification success of 82.4% and an F1 score of 85.8% using the test dataset[8].

Another study [35] , used RGBD rather than Lidar as the chosen method of 3D sensing. Unlike with the previous study using Lidar [8], there is no reflectance value of the different objects within the field of view, such as the target crop and surrounding leaves and branches. This makes fruit detection and localisation challenging tasks due to factors such as varying illumination conditions, partial occlusion of the fruit, and the colour variation of the fruit. An algorithm based on RGB-D images was developed to detect and locate citrus fruit in outdoor environments, with the intended use to be for robotic harvesting. The hardware used in this study consisted of a Kinect V2 sensor to capture the RGB-D images. The Kinect V2 uses ToF technology for depth sensing, with a reported accuracy of 2mm [36]. In this study, 506 RGB-D images were collected, with the sensor being placed approximately 600mm from the target tree. 80% of these images were randomly selected to make up the training set, with the remainder making up the test set. The methodology used after initial data acquisition consisted of depth filtering, Bayers-classifier-based image segmentation, and density clustering. These clusters were then evaluated by an Support Vector Machine(SVM) classifier, before the 3D position and size of the fruit could be extracted. To validate the detection, localisation and sizing performance, quantitative experiments were also performed. From these experiments, the proposed algorithm achieved an F1 score of 0.9197, which meets the accuracy and robustness requirement of a fruit harvesting robot[37]. As stated in [35], the proposed algorithm achieved good results, the detection pipeline can still be improved, with the possibility of implementing factors such as curvature detection and also investigating the impact of strong sunlight on the data of the ToF sensor, as this can lower the depth precision according to [8]. From the results being compared to a ground truth sample, the positioning errors of the x,y,z coordinates were 7.0 $\pm$ 2.5mm, -4.0 $\pm$ 3.0mm and

13.0 ± 3.0mm, which the author states would meet the accuracy requirements of a fruit harvesting robot[38].

Another application of using 3D vision systems in an agricultural environment can be seen in [39]. In this study, a Microsoft Azure Kinect RGB-D camera is used in the application of measuring tree stem diameter. This information can then be used for sustainable planning and management of forests. The Azure Kinect is an RGB-D sensor that uses Time of Flight depth sensing technology, as well as a 12MP RGB camera and Inertial Measurment Unit (IMU) that can be used for odometry [40]. As seen in [41], previous versions of the Kinect sensor were used outdoors with varying results, mainly due to the influence that direct sunlight can have on the IR sensors[8]. Because of the advancements in the Azure Kinect, such as greater spatial resolution, reduced power consumption, depth sensor global shutter spanning, binning pixels to extend range as well as more control over settings for capture mean that the Azure Kinect is more likely to be a viable solution for outdoor 3D sensing[41], which is what this research axamines. For the data collection for the tree stem measurements, 51 trees from 9 species were sampled. Ground truth measurements of each tree were also taken, using a diameter tape at a height of 1.3m. The Azure Kinect was then mounted on a tripod with a level plane at a height of 1.3m to collect 3D point cloud data of the tree stems. The sensor was positioned 1m away, with a 4 second video recording being taken. The sensor was then moved away from the tree in 1m increments, to a maximum of 5m, with the same recording procedure being undertaken at each increment. Recordings were also collected using different depth sensor Field of View(FOV) settings depending on the distance from the tree. Data was collected for all FOV settings at the 1,2 and 3m increments, with the unbinned Near Field of View(NFOV), 2x2 binned NFOV and 2x2 binned Wide Field of View(WFOV) being capture at 4m, and the 2x2 binned NFOV at 5m. These FOV settings were used based on results from previous trials where the stems were captured using different FOV settings at varying distances. The ambient light was also recorded during the trial, with the measurement being taken from next to the Kinect immediately before recording. Using the video captured from the depth

and RGB streams, the initial point clouds could be extracted. These point clouds were then cleaned, removing points from low density areas. The stem models could then be created using a circle fitting approach. A 20cm slice was extracted from each stem point cloud, between the heights of 1.2m and 1.4m. These slices were then segmented into 2cm width strips. A best fit circle was then calculated based on a 2D projection of these points, using the least squares method described in [42] .The diameter could then be estimated as the mean value of the diameters of each fitted circle to the nearest 10mm. Using these results, linear regression models were then derived using the Root-Mean-Square Error(RMSE), relative Root Mean Squared Error(rRMSE)and model bias were used to assess the performance of the Azure Kinect sensor. The results from the authors trials can be seen below in Figure 3.0.1.

		Sensor Distance from Stem				
Field of View	Error	1 m	2 m	3 m	4 m	5 m
NFOV 2 × 2 Binned	RMSE (cm)	9.50	8.43	12.56	18.94	18.47
	rRMSE	0.29	0.26	0.37	0.58	0.74
	Bias (cm)	2.71	2.05	−4.59	−16.83	−27.05
	Trees Failed	0	0	1	15	41
NFOV Unbinned	RMSE (cm)	10.13	12.56	13.85	18.93	*
	rRMSE	0.31	0.39	0.43	0.77	*
	Bias (cm)	1.57	−4.34	−13.02	−32.94	*
	Trees Failed	0	0	22	39	*
WFOV 2 × 2 Binned	RMSE (cm)	11.56	14.02	16.86	14.85	*
	rRMSE	0.36	0.41	0.58	0.7	*
	Bias (cm)	0.69	−6.15	−16.94	−34.94	*
	Trees Failed	0	2	21	30	*
WFOV Unbinned	RMSE (cm)	13.67	16.88	6.83	*	*
	rRMSE	0.44	0.67	0.31	*	*
	Bias (cm)	−5.49	−25.07	−23.25	*	*
	Trees Failed	1	25	44	*	*

Figure 3.0.1: Figure showing results from trial using Azure Kinect[39].

From the results, the binned capture method had a lower error in stem diameter measurement when compared to the equivalent unbinned setting, with the lowest error being observed using the binned NFOV setting at 2m. As well as having the lowest error, using this setting also resulted in more point clouds being able to be generated from the captured data, as can be seen in the 'Trees Failed' row in Figure 3.0.1. This can be seen with 1 tree failure at 3m with the NFOV Binned setting, compared to the next lowest of 21 Tree Failures using the WFOV Binned settings. Figure 3.0.2 below shows the mean point density using the different settings over multiple distances.

Using the data from the ambient light measurements and normalized error, the

	Sensor Distance from Stem				
Field of View	1 m	2 m	3 m	4 m	5 m
NFOV 2 × 2 Binned	27.48	10.11	5.26	3.06	2.02
NFOV Unbinned	26.52	8.62	3.03	1.74	*
WFOV 2 × 2 Binned	27.07	8.58	4.14	2.55	*
WFOV Unbinned	22.08	3.40	1.73	*	*

Figure 3.0.2: Figure showing density results from trial using Azure Kinect[39].

Coefficient of Determination(R^2) was calculated to assess the affect that light intensity may have on the sensor, and how it may impact the results obtained from the sensor. It was found that at close ranges, between 1 and 2m, there was a negligible correlation between residual error and ambient light using the binned NFOV. This error increased with an $R2$ of 0.21 at 3m using the same mode. The unbinned NFOV settings at 1m also had a negligible correlation between light and error. From the results of this paper, the best settings and conditions for capture using the Azure Kinect is the 2x2 binned NFOV mode at distance of 2m away from the object. Unlike it's predecessors, and other RGB-D sensors, ambient light was found to have a minimal effect on the depth data apart from lowering the point cloud density.

To further explore the use of RGB-D cameras in agricultural applications, another study explores the use of an RGB-D camera used for 3D vision system being applied to identify and localise broccoli heads in field was examined[43]. The main objective was to investigate the feasibility of using consumer RGB-D cameras mounted on a tractor to identify the crop, in this case broccoli, and subsequently be able to localise and size the broccoli based on the data gathered. The method used for evaluation included collecting sensory datasets from two different countries with complementary growing seasons, from the UK where broccoli is a summer crop, and Spain where it is a winter crop. For data collection a Kinect V2 was used, the same sensor described in [8]. The experimentally verified depth accuracy of this sensor is less than 2mm in the central region of the sensor[44]. The sensor was used inside an enclose to block direct sunlight in order to minimise noise in the collected data, with artificial lighting also being used to help regularise the colour images from the sensor and make it possible for data to be captured in any lighting conditions. The platform with the sensor and enclosure was

then fitted to the back of a tractor using a 3 point linkage as seen below in Figure 3.0.3. Once the initial data had been captured, the author used a global recognition method for detecting the broccoli heads. This is applied by processing the depth data and applying the following steps of the detection pipeline. These steps consist of 3D point cloud pre-processing, feature extraction, classification and temporal filtering. The first step that is applied is the point cloud pre-processing. Outlier removal is performed to eliminate any noisy data which could lead to errors in subsequent data processing. This is done by computing the distance of each point to its k neighbours and finds the mean and standard deviation of these distances. Any points that fall outside a certain distance threshold which is defined by the sum of the global mean and standard deviation. The next step in pre-processing is ground filtering, which is segmenting the ground using a depth threshold. In this case, a filter was applied over a range between 0.5-1.0m, meaning that anything outside this range was excluded. This distance was based on the height of the sensor from the ground during testing. After this filtering, cluster segmentation could be applied. A distance based clustering algorithm to cluster points based on the Euclidean distance between point pairs was then applied to the data. Once cluster segmentation is complete, features can then be extracted from the depth data using feature descriptors. Using this data, size estimation as well as location relative to each neighbouring head of broccoli could be calculated. Overall, it was found that this study confirmed the potential for low cost 3D imaging for commercial crop harvest.

Figure 3.0.3: Experimental setup using tractor with enclosure and RGBD camera[43].

4.0 Virtual Environments

With the advancement of these 3D scanning techniques mentioned above, comes the need for a pipeline in order for a user to interact with the 3D sensing data when considering the use in the application of robotic harvesting. With the constant advancement of 3D graphics engines, being able to utilize these tools in an application with point clouds would be optimal.

4.1 Unity

Unity is a game engine that is used for creating interactive media, typically in the form of video games [45]. It has been widely used across a variety of industries, from the aforementioned game development, to running complex simulations for research [46]. Included in the wide range of applications of this game engine is the visualisation and interaction with point cloud data.

4.1.1 Point Clouds in Unity

In a study [47] investigating the use of terrestrial Lidar and UAV photogrammetry, Unity was used to create an interactive visualization of an architectural heritage building. In this study, a hybrid point cloud was created using both the terrestrial laser

point cloud data and the UAV point cloud data using Iterative Closest Point(ICP) algorithms. Using this combined point cloud and primitive 3D objects, a 3D model of the scanned environment was created. Using the 3D .obj model, it could then be imported into Unity [45]. Using this method, it was found that using this method was a good way of providing an interactive environment, however it seemed that it was not as straightforward as being able to import a raw point cloud file into the game engine.

In further studies [48], an interactive pipeline was developed for the use of rendering point clouds. Using C# the Unity Pointcloud Interactive Core rendering pipeline could be implemented. Using this newly developed pipeline, it creates the ability to process larger datasets, and implements a dynamic batching algorithm in order to maintain good visual quality even as the size of the models may increase [48]. Using this pipeline provided a more efficient process of processing the point clouds by increasing the GPU (Graphics Processing Unit) performance using a delayed rendering algorithm, essentially rendering the point cloud in segments. This study shows that with the knowledge of the algorithm's used and a working understanding of implementing a C# component in Unity, efficient point cloud rendering can be achieved.

4.2 Unreal Engine

Similar to Unity described above in Section 4.1, Unreal Engine(UE) is a 3D computer game engine, written in C++, supporting a wide range of desktop, mobile, console and virtual reality platforms[49]. Since it's release in 1998, constant development has been achieved, making it currently one of the best options for interactive 3D development.

4.2.1 Point Clouds in Unreal Engine

As demonstrated in [50], Unreal Engine can also be utilised for point cloud visualisation and interaction. The goal of this study was to provide a method to digitise 3D scans of trees. It is also stated that in this method of visualisation, there is great potential for the use of VR due to the inherent 3D nature of point clouds and subsequent digitised models [51]. The methodology proposed in this study was to use a point cloud and

identify the tree by parts, primarily trunk and foliage. Once these have been identified, separate processing can occur in order to digitise the trees. Figure 4.0.1 below shows the proposed workflow process.

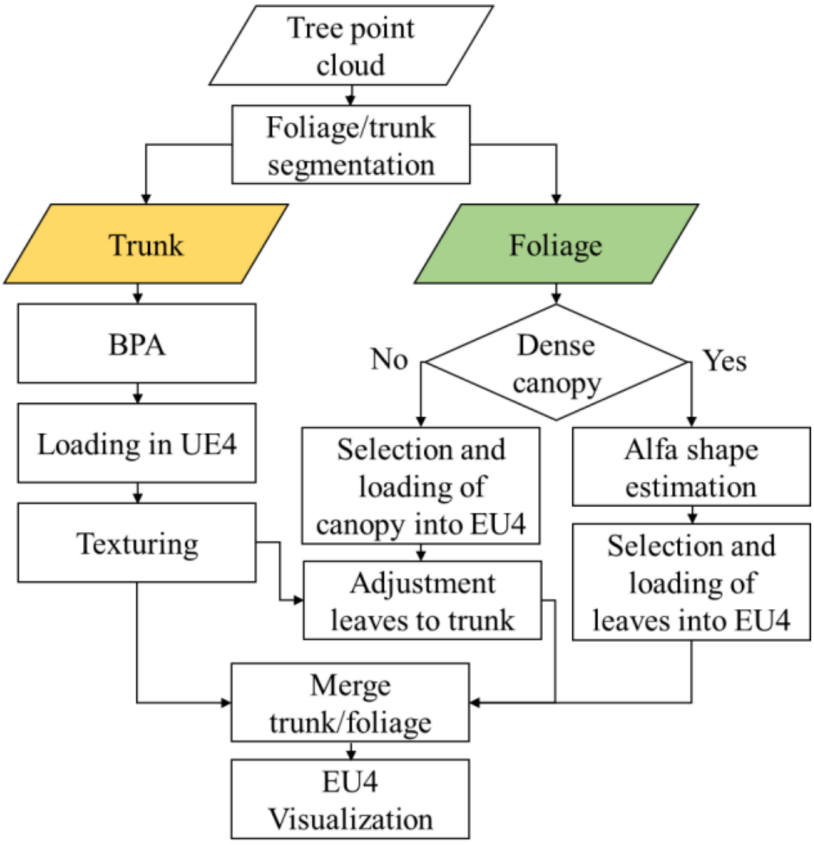


Figure 4.0.1: Workflow of proposed digitisation method[51].

In order to segment the tree features, multiple features were assessed and calculated. Included in these features were linearity, planarity, sphericity, first and second principal components, curvature, eigentropy, omnivariance and verticality. As well as these processes, an outlier removal filter was applied to remove isolated points, in order to provide more accurate data. These processes were undertaken in CloudCompare. Once segmentation had been completed, digitisation of each component could be completed. The resulting meshed models were then imported into Unreal Engine and merged to create a fully digitised tree. Figure 4.0.2 below shows the comparison between the point cloud image and the digitized model of a tree using this method.

Figure 4.0.2: Point cloud tree and digitized tree[51].

Overall it was found that using Unreal Engine to digitise point cloud data was highly effective, however it was highly dependant on the pre-existing libraries of leaves and canopies that could be implemented [51].

In further studies [52], Unreal Engine has been used in a variety of 3D visualisation applications. In this case, it was utilised as a platform to stream point cloud data to render to be used for VR based telepresence, in this specific case in an operating room. The advantages of creating this immersive and high quality virtual environment are that it allows distant specialist doctors to assist local doctors effectively, one step removed from being present[52]. Although telepresence has been used extensively in this field previously[53], traditionally it has been in video form [52]. In this research, an Azure Kinect was used as the method of 3D sensing. In conjunction with this RGBD camera, Unreal Engine was chosen as the visualisation platform in order to provide high quality graphics and robust network components [52]. As well as these factors, Unreal Engine has built in collision properties that can be applied to point clouds, whilst also supporting a wide range of VR headsets[49][52]. One factor that was found in the initial research was that the Unreal Engine network components are not suitable for low latency interaction, which is very important for live data streaming and interaction.

As a result, a custom server was developed, with the point cloud data being streamed to this server, before being integrated into an Unreal Engine virtual environment. This developed pipeline utilised C++ and CUDA[54], with the system diagram being shown below in Figure 4.0.3.

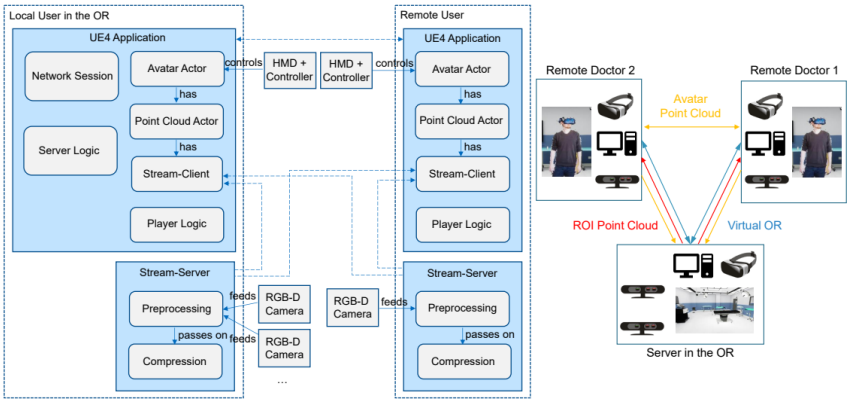


Figure 4.0.3: System architecture and setup for VR telepresence[52].

Using the Azure Kinect, the colour and depth images obtained were compressed and transmitted to the client server, where they were decompressed. It was decided to use this method rather than compressing the point cloud data itself, as this process allows the use of more efficient compression algorithms[55]. Using this method also allows multiple point clouds from different cameras to be registered relative to each other to provide a more complete 3D environment. Typically, Unreal Engine does not support native point cloud rendering, however a point cloud plugin, 'Lidar Point Cloud'[56] is available that can handle this rendering and collision building process. With theis processed live feed point cloud data now available in Unreal Engine, it cloud be used in conjunction witha VR headset in order to provide an immersive and interactive experience for the user. Using the streaming methods described above, a low latency was achieved, with the motion to photon latency of 120 to 150ms, of which approximately half of can be attributed to the inherent Azure Kinect latency. Due to the nature of the compression algorithm's, this method also does not require the use of multiple high end PC's in order to efficiently run, which is another great advantage. Based on user testing, it was found that 83% of test participants affirmed that there

were moderate to high benefits using this system compared to traditional video based telepresence[52]. It was also observed in this study that a limiting factor was the RGB-d camera's resolution, which was identified as an area of potential improvement reliant upon the advancement of new hardware. Overall this study presented the potential benefits of VR point cloud telepresence, with the potential for Augmented Reality(AR) to be utilized in further work.

4.3 Applications of Virtual Reality

As seen in the above study[52], there is potential for VR to be used across a wide range of industries for multiple applications, including in the agricultural and horticultural sector. In another study[57], VR was used as a visualisation method for a digital twin of a greenhouse environment, however this study only utilised RGBD imagery rather than 3D data. Further applications of VR were utilised in [58], where a VR environment was utilised in the training of staff for pruning trees. It is stated that training staff in the science of tree pruning is challenging due to the time period between pruning and regrowth[58]. In this study, a virtual environment was created in which to train people in tree pruning, where the virtual tree's can be controlled to have accelerated regrowth. It was found that this methodology could be useful for a training application however the technology limitations at the time prevented accurate models being produced and visualised in an effective manner. As seen in [59], a virtual environment was set up in order to control a tractor in a virtual environment,. The goal of this study was to make an accurate virtual representation of a tractor, with the same control inputs as a real tractor in order to test different aspects of the tractor in varying conditions. The resultant environment was one that mimicked the characteristics of a real tractor, meaning that for further research this could be applied to a real tractor controlled in this virtual environment. In this test only a virtual environment was used, with no 3D scanned element being implemented.

As seen in the studies mentioned above, VR has been utilised accross a wide range of areas, however there is still scope to utilize innovative methods to combine the use of

VR with a 3D scanned environment in order to achieve an output in a real environment, in this case robotic harvesting.

Methodology

5.0 Experimental Setup

5.1 Orchard Experimental Setup

In order to test and gather data and validate the precision and accuracy of the depth sensing devices being tested, an experimental setup was designed and constructed. The first consideration for this test setup was the environment, with the main two choices being indoor and outdoor. Indoor testing would provide a more controlled testing environment, however it would lack key factors, such as sunlight, which could be a major contributing factor in the results obtained [60]. Due to this, it was decided to do testing both a controlled environment as well as in an environment which would replicate real conditions.

A space in the Apple Innovation Orchard at the Massey University Orchard was obtained and used for this experimental setup. The next step towards constructing this test rig was to decide upon the best targets for the 3D capture testing. As the test was meant to replicate a simplified in-orchard application, the objects needed to be similar to real fruit, in this case apples. The option of using real apples was considered, however was found unsuitable due to the nature of picked fruit decaying quickly, and with the data being collected over an extended period of time, it was decided that this was not a viable option. Another option that was considered was to use simple spheres, which have roughly the same shape as apples. The best option for these spheres would be for them to be 3D printed, so the right sizes and mounting points could be achieved. The final consideration, that combined the first two options, was settled upon being

the best option. This option was to use 3D printed apples. In order to achieve this, a variety of apples were acquired, in this case an assortment of different size and shape Envy apples. The first step towards creating 3D printed apples was to create a 3D scan of them. Using the resources available at the time, it was decided to use the Einscan 3D scanner to scan the apples. An example of one of the resulting scans can be seen below in Figure 5.0.1.

These scans were then meshed and exported as both .obj and .stl files, both 3D object filetypes. The resulting stereolithography files (.STL) were then used to print the scanned apples out of polylactic acid(PLA) on a Creality Ender 3 3D printer. An example of one of the printed apples can be seen below in Figure 5.0.1.

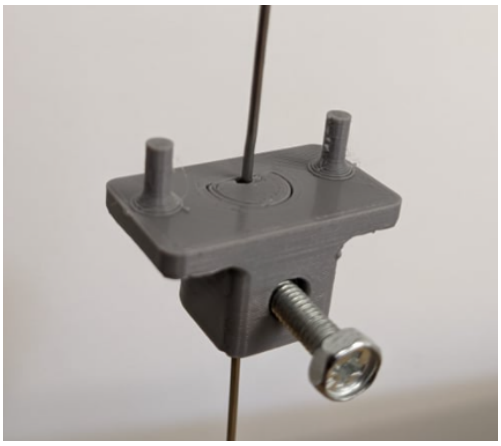
(a) Textured 3D Scanned Envy apple using the Einscan 3D scanner

(b) 3D Printed apple using PLA filament

Figure 5.0.1: 3D scanned apple and the resulting 3D printed apple

These printed apples were then painted with a Plastidip paint to help protect them from exposure to outdoor elements and also replicate the colour of real apples, in this case dark red. When setting up the experiment as seen in Figure 5.0.3, some of the 3D apples were left unpainted in order to see if the colour of the target object affected the depth image. Using previously existing stringer wires set up in the orchard, which were approximately 0.5m apart, wires were run vertically between these stringers. The 3D printed apples were then mounted on these vertical wires on mounts designed to

slide up and down these wired, and then be locked into the desired position. Figure 5.0.2 below shows these mounts, which were also designed to hold the apples in place around the centre axis, to reduce the amount of movement in the apples as much as possible. Figures 5.0.3 below show the 3D printed apples on the vertical wires using the mounting system, in place between the wire stringers.



(a) 3D Printed mount

(b) 3D printed mount in position with 3D printed apple

Figure 5.0.2: 3D printed mount to allow the precise positioning of the 3D printed apples

(a) Front view of experimental setup

(b) Side view of experimental setup

Figure 5.0.3: 3D printed apples set up at precise locations in the orchard using the 3D printed mounts

5.2 Depth Testing Experimental Setup

For other depth camera characteristic testing, both an indoor and outdoor setup was used. This setup was for more basic foundational testing, so was a simplified setup that met the needs for the experiments undertaken. A medium density fibreboard (MDF) 1.0m x 1.0m in size was used as the target surface, being chosen as it was easily transportable and could be used for both the indoor and outdoor experiments to keep the target surface consistent. Once this board was secured against the wall, lines were then accurately marked on the floor using a tape measure at varying distances from the target surface, starting at 0.50m and progressing in intervals of 0.25m to a maximum distance of 2.00m.

6.0 Investigation of Depth Accuracy

The first aspect of 3D vision systems that was investigated was the depth precision and accuracy. This experiment was undertaken using the Intel Realsense D435i[61], described above in Section 1.3, as for the scope of this project it was decided that it was the most suitable, due to it's low cost, versatility, and well documented support. This RGBD camera was used in conjunction with the experimental setup described above in Section 5.1 . The Realsense camera was set up at an initial distance of 0.5m away from the MDF target surface, which was accurately measured with a tape measure. The Realsense camera was perpendicular to the ground plane, and parallel to the target surface. This was achieved by using a square edge against the front of the Realsense camera, as depicted below in Figure 6.0.1.

6.1 Data Collection

Using the Intel Realsense Viewer SDK, the point cloud could be saved and then be exported for processing and analysis. This was first done at the initial measurement of 0.5m, with each subsequent exported point cloud progressing 0.25m further from the target surface, to a maximum of 2.0m. This test procedure was then repeated a further

Figure 6.0.1: Intel Realsense D435i with square edge to ensure perpendicular angle to base surface

two times, giving a total of three replicate data sets. This test was carried out in both indoor and outdoor conditions using the same setup, as based on [62], outdoor sunlight could affect the functionality and accuracy of the RGB-D camera.

With the raw data collected for each distance, the data was then processed using Cloud Compare, a 3D point cloud processing software. As discussed and reviewed above in Section 13.1.1, Cloud Compare was chosen due to it having all the features needed for point cloud analysis, with the ability to create meshes from point clouds and also import and export multiple 3D point cloud file types, which would be needed for other aspects of this project. The main compatible filetypes needed for the research include .PLY and .E57 files.

For the experiment conducted in Section 6.0, the exporting file format used was .PLY, as that is the native export filetype that the Intel Realsense Viewer uses.

6.2 Data Processing

Each scan was then imported to CloudCompare individually. Figure 6.0.2 below shows an example of the raw point cloud imported directly into CloudCompare before any

processing has taken place. The process of importing the point clouds was then repeated for the three scans at each distance, with a total of 21 scans being imported. As the scans were all taken in the same session, the alignment of the scans was relative to eachother, meaning that by default they were parallel to eachother, with only small spatial adjustments needing to be made for depth analysis. The next step in the point cloud processing was to crop the clouds so that just the region of interest was left. This cropping was done so that the selected region was selected on the flat plane of the first scan, 0.5m, and was projected through the other 20 scans, leaving identical remaining regions of interest for analysis. The result of this cropping on the entire collection of point clouds can be seen below in Figure 6.0.3.

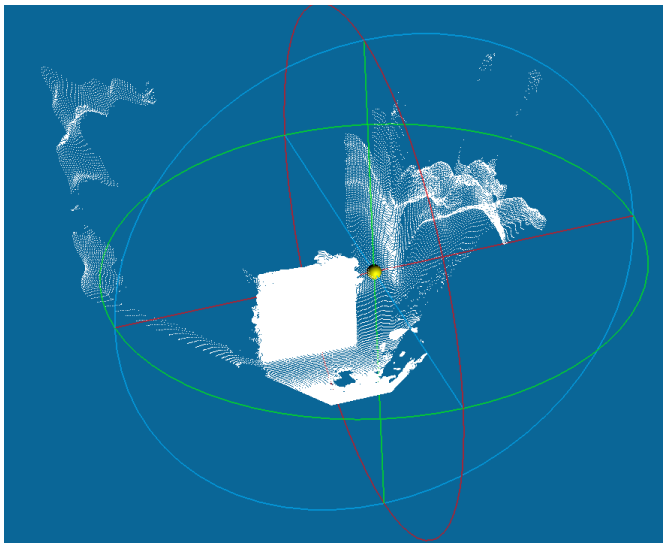


Figure 6.0.2: Raw point Cloud .PLY file of the target surface imported into Cloud-Compare

The next step in processing these point clouds was to insert a reference plane. This reference plane was inserted at the origin of the environment, coordinates 0,0,0, which correlates to the camera position during point cloud capture. The plane was positioned parallel to the point clouds. Since the camera was precisely set up perpendicular to the ground, and therefore parallel to the target surface, there was minimal disparity between the angle of the plane and angle of the point cloud on this axis. However, on the longitudinal axis, there were some discrepancies between scans at differing distances,

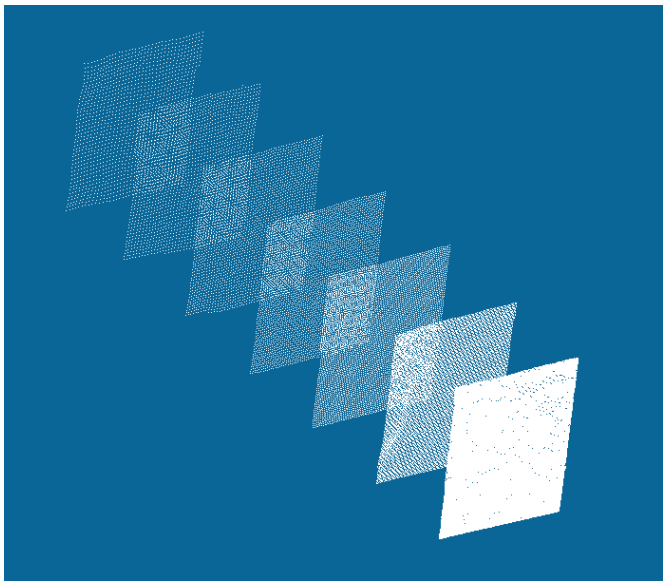
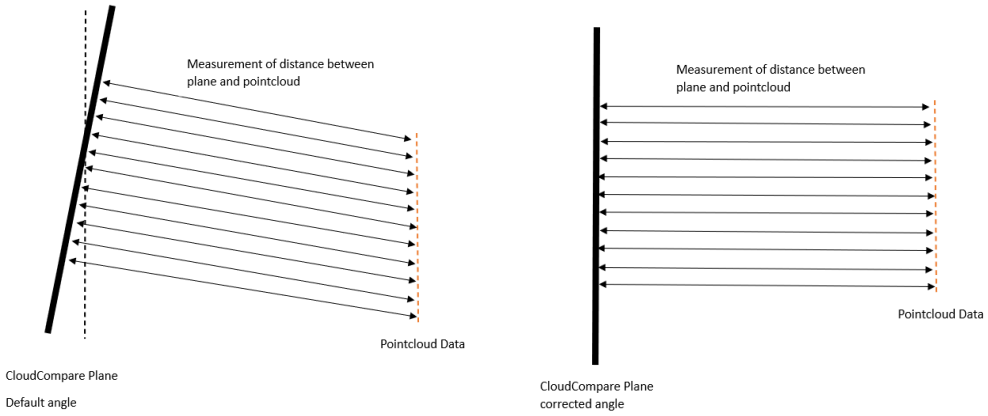


Figure 6.0.3: Point clouds collected at each distance processed and cropped

creating a slight difference between the angle of the plane relative to the angle of the point cloud. After checking all the point clouds, it was found that the maximum difference was only 2 degrees. Although this may seem like an insignificant amount of difference, the effect that this can have on the calculations of average point distance and standard deviation is significant.

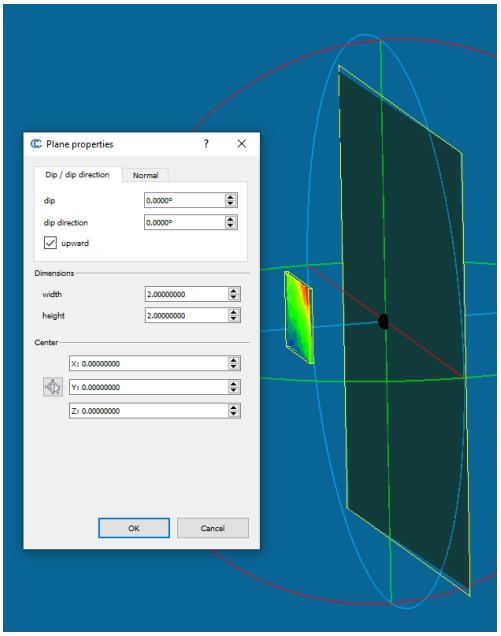
In order to reduce this error as much as possible, the plane angle relative to the point cloud angle was optimised. The first step in this process was to compute the distance between the primitive object, in this case a plane, and the point cloud also known as the P2C distance. In doing this, the perpendicular distance between the plane and each point is computed. Figure 6.0.4 below demonstrates this. If the plane is not parallel to the point cloud, the distance between some points will increase and decrease, therefore affecting the mean and standard deviation data. Once the P2C distance was calculated, the plane angle could then be optimised. This was done by incrementally increasing and decreasing the angle of the plane about the X axis. Figure 6.0.5 shows this process. Essentially this process was used to minimise the standard deviation value that was calculated from the P2C distance, with the smallest standard deviation corresponding to the most parallel plane angle. This was done in increments

of 1 degree, as this was the smallest increment that this value was able to be modified by. This process was then repeated for each point cloud. As seen in Figure 6.0.5 below, it can be seen that the plane angle was optimal at 2 degrees as this gives the lowest standard deviation and most accurate mean value compared to the other plane positions. Once the plane angle for every point cloud had been optimised, the data for mean distance and standard deviation could be collected. This was done by creating a distribution of the points relative to the plane. This was repeated for all point clouds at each distance.

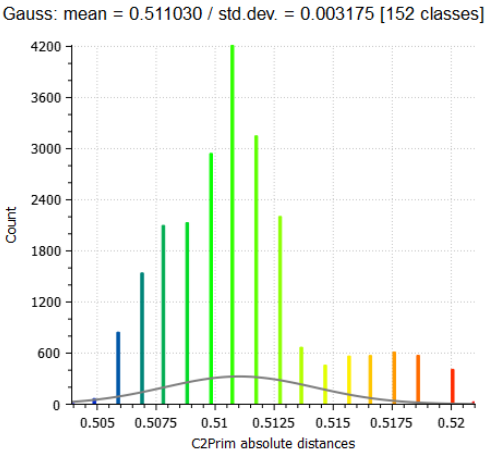


(a) Default plane angle measurements (b) Corrected plane angle measurements

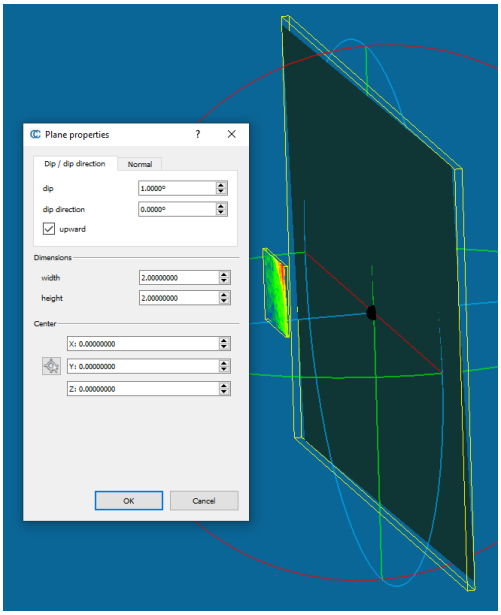
Figure 6.0.4: Diagram demonstrating the measurement between point cloud and plane



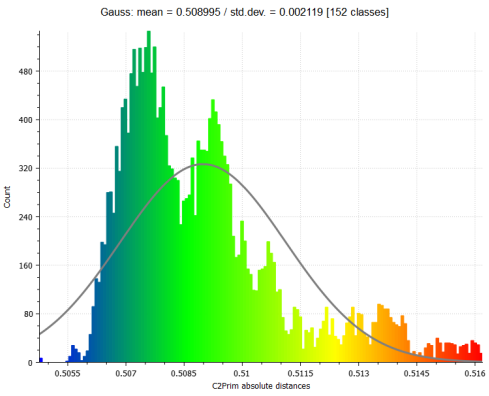
(a) Initial plane position at 0 degrees



(b) Distribution of points relative to the plane at 0 degrees

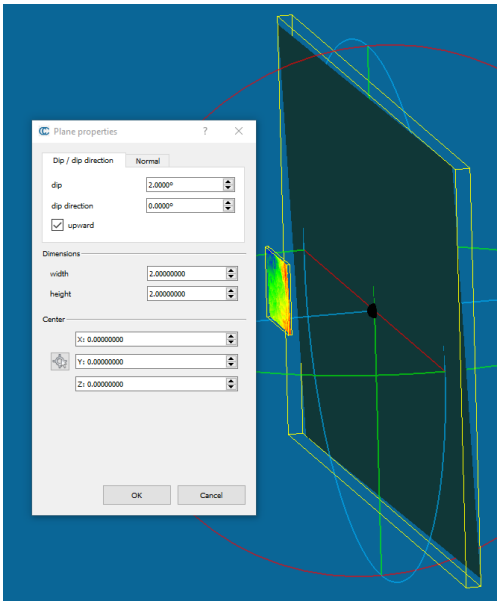


(c) Plane adjusted by 1 degree

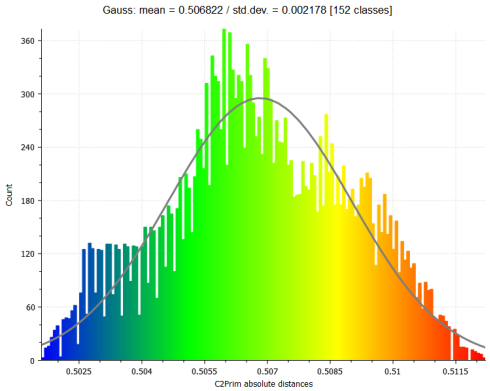


(d) Distribution of points relative to the plane at 1 degree

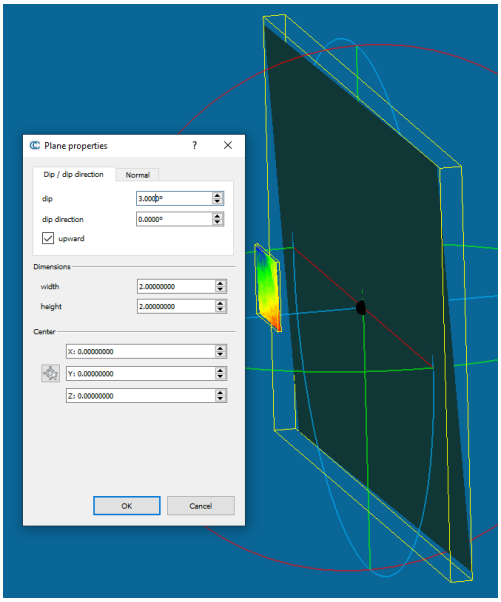
Figure 6.0.5: Plane Angle Optimisation



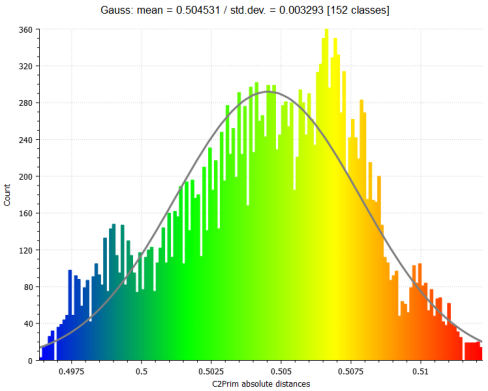
(e) Plane adjusted by 2 degrees



(f) Distribution of points relative to the plane at 2 degrees



(g) Plane adjusted by 3 degrees



(h) Distribution of points relative to the plane at 3 degrees

Figure 6.0.5: Plane Angle Optimisation

Once the data had been optimised relative the the plane, it could then be processed in order to extract and calculate key information. The way that the raw data was presented was with a distribution of small depth ranges with 'N' occurrences. For example, for the 0.75m dataset, between 0.739m and 0.740m, there were 107 data points. Using this data, the total distance was calculated by multiplying each distance by the number of data points. The mean was then calculated by dividing the total distance by the number of data points. Using Equation (6.0.1) below, the variance for each point could then be calculated.

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1} \tag{6.0.1}$$

Where x_i is the distance value, $\overline{x}$ is the mean value, and n the number of value in the data set. Once the variance for each point in the data set was calculated, the total variance could be calculated by the sum of the variance of each point. Using this value, the average variance could be found by dividing by the total number of data points. As shown by Equation 6.0.2 below, taking the square root of this number gives the standard deviation of the data set.

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{6.0.2}$$

This process was carried out for each data set at each distance, giving the standard deviation at each distance. Using this value, the standard deviations could be calculated and compared. With the standard deviation now also calculated, a Gaussian distribution could also be fitted to the model in order to help better visualise the distribution. The distribution of each data set at each distance can be seen in Section 10.0.

7.0 2D Position Accuracy

The next component of the 3D vision system that was evaluated was the object localisation accuracy. This experiment was also conducted using the Intel Realsense D435i, in conjunction with the in-orchard experimental setup described above in Section 5.0 The main goal of this data collection was to establish and verify the precision and accuracy of the spatial data acquired by the 3D sensor, in this case the D435i. The data collected was used to analyse the accuracy of the 2D coordinates, being X and Y, of objects in the depth image, with the depth, Z being assessed independently in Section 6.0 above.

7.1 Data Collection

The form of data collection gathered with the D435i were again point clouds as above in Section 6.0 , which were obtained using the Intel Realsense SDK. Based on results gathered from the experiment undertaken in Section 6.0, it was decided to set up the camera at a distance of 1.5m for data collection. This was due to the fact that the quality of the point cloud was still acceptable at this distance based on the data collected above in Section 6.0 combined with the field of view of the camera, it was able to fit the complete experimental setup in one frame, meaning that multiple images would not be needed to be overlaid and joined, which would introduce another source of error and uncertainty. On the settings for the D435i, the depth range was also set from 1.0m to 2.5m, eliminating the collection of background points and just focusing on the points in the target area. The D435i camera was positioned 1.5m away and at a height of 1.0m, with the front of the camera again being perpendicular to the floor plane. The raw preview image as viewed in the Intel Realsense Viewer can be seen below in Figure 7.0.1. This view was then exported as a single point cloud as a .PLY file. This process was then repeated 10 times to have a data set of point clouds to gather positional data from.

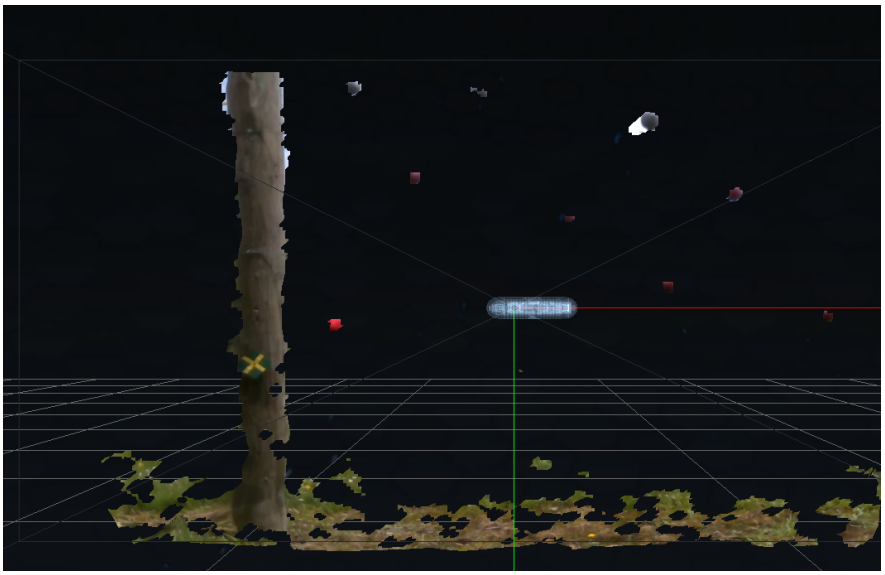
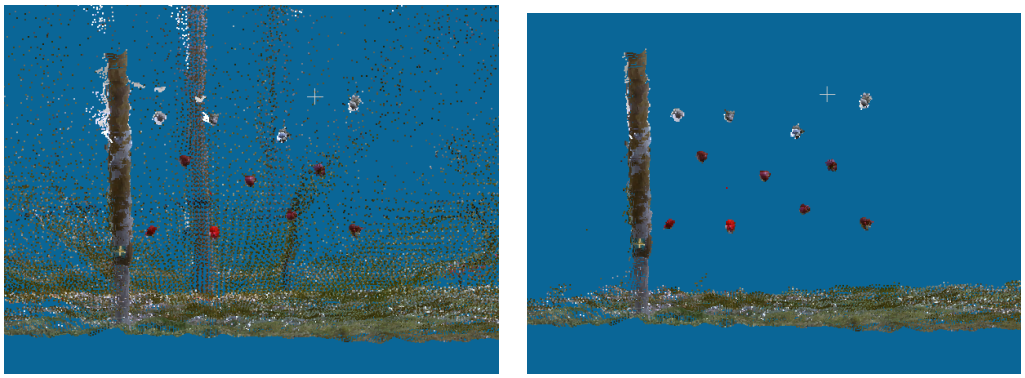


Figure 7.0.1: Raw image of the outdoor experimental setup using the Realsense SDK Viewer

7.2 Data Processing

The collected point clouds were then imported into CloudCompare. Each point cloud was imported individually. The raw point cloud can be seen below in Figure 7.0.2. Once each cloud was imported, stray points were then deleted in order to clean the point cloud. This was by using a feature in CloudCompare that deletes points further than a selected distance from neighboring points. In this case, a value of 0.1m was used to eliminate any coarse stray points. An example of the processed cloud can be seen in Figure 7.0.2.

Once all 10 clouds were processed, data collection from the clouds could then be undertaken. The ground truth measurement between each 3D apple had been established by manual measurements between marked centre points. This manual measurement was undertaken using a tape measure, as it was decided to be the most efficient and reliable way to get accurate measurements with the resources available at the time. Although there is still some error in these measurements, it can be quantified and is taken into account in the analysis of the results of this experiment. Using the 'measure' feature in CloudCompare, the distance between any two points can be measured.



(a) Raw point cloud of the experimental setup in CloudCompare

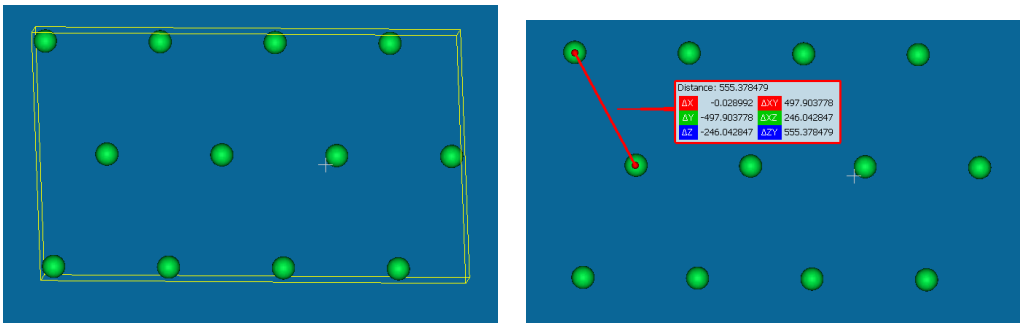
(b) Processed point cloud of the experimental setup in CloudCompare

Figure 7.0.2: Point cloud data showing the experimental setup in CloudCompare

This was used to measure the distances between the 3D printed apples, which were then compared to the ground truth measurements taken. This was undertaken on all 10 scans, the results of which can be found in Section 11.0. At the same time as this experiment was undertaken, a similar test was also carried out to compare and quantify the error in the human input of using the 'measure' feature in CloudCompare. This was done by setting up 12 spheres at known locations on a single depth plane. The setup of this experiment can be seen below in Figure 7.0.3. The same process that was used on the point clouds was replicated for this test, with the centre to centre distances of the spheres being found using the measure function in Cloudcompare. Figure 7.0.3 shows this process. This experiment was also undertaken 10 times, with the 2D positions for each sphere being recorded each time. Using these results, the error from this process could then be factored into and compared to the results obtained from the measured 2D point cloud distances. These results can also be found in Section 11.0.

8.0 Point Cloud Density

The next step in the evaluation of 3D vision systems, in this case using an RGB-D camera, was to assess the point cloud density as a function of distance. The density of the point cloud of the target object plays an important part in the quality and perception of the object. Fundamentally, the point cloud density decreases as distance increases [63].



(a) Experimental setup consisting of 12 spheres at set distances in CloudCompare

(b) Using the measure function to determine centre-to-centre distances in CloudCompare

Figure 7.0.3: Experimental setup using 12 spheres at set locations in CloudCompare in order to compare the user error of the 'measure' function.

As distance increases, the area captured by the camera Field of View also increases. This experiment was conducted to quantify the quality of a point cloud over varying distances, in order to assess optimal operating ranges in an orchard environment.

8.1 Data Collection

This experiment was conducted using a board that was 0.3m x 0.3 m as the target surface, and was undertaken in an orchard environment during the day, in sunlight, as well as in an indoor environment. The target surface was set up perpendicular to the ground, therefore parallel to the Realsense camera. The experiment was again conducted at multiple distance intervals, starting at 0.5m and increasing in steps of 0.25m to a maximum distance of 2.0m. Although the size of the target board is mostly immaterial, the larger the test surface the more data points collected. The size of 0.3m x 0.3m was chosen as it was the largest size of test surface that comfortably fitted in the image frame at the closest measurement point, 0.5m. As described above in section 6.0 , the Intel Realsense D435i was used to capture the point cloud frames at each distance. A single frame at each distance was exported as a PLY file using the Intel Realsense Viewer, with this process being repeated three times in order to have three replicate datasets.

8.2 Data Processing

Once this data was collected, it was then processed and analysed in CloudCompare. As in Section 6.0 above, each point cloud was individually imported into CloudCompare. Figure 8.0.1 below shows an example of a raw point cloud image as seen in CloudCompare before any processing. Each point cloud was then cropped in order to isolate the target surface, with all points outside of this area being selected and deleted. This was repeated for all 3 point clouds at each distance. An example of the processed point cloud can be seen below in Figure 8.0.2 . Using the in-built function in CloudCompare, the total number of points for each point cloud could be found. As each cloud only consists of the target surface since processing, this number corresponds to the point density of the 0.3m x 0.3m surface. This number was then converted to a density using SI units of points/m^2 in order to be compared to other data collected in this experiment as well as external data. The resultant density data can be seen in Section 12.0 with the relationship between distance and point cloud density displayed in Figure 12.0.1.

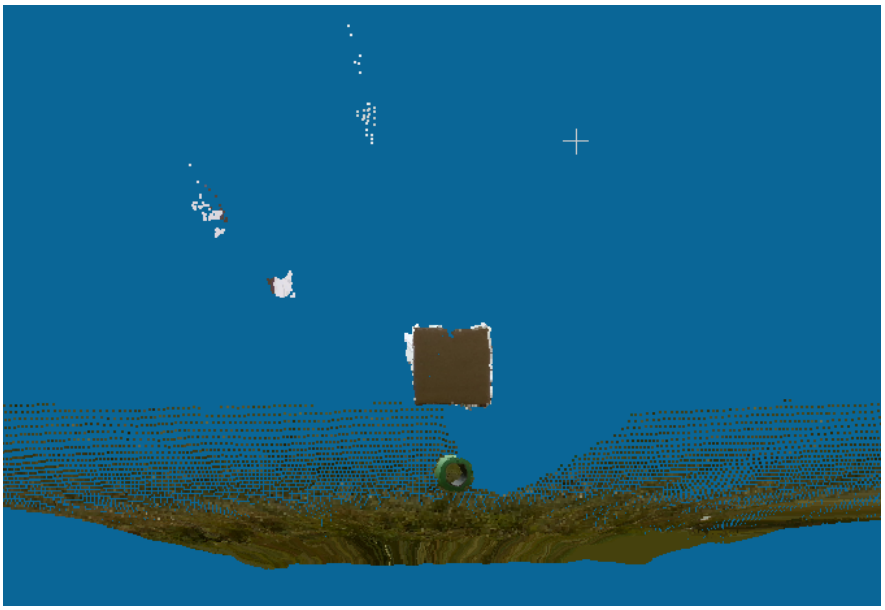


Figure 8.0.1: Unprocessed point cloud image in CloudCompare of the target 0.3m x 0.3m surface at a distance of 0.5m

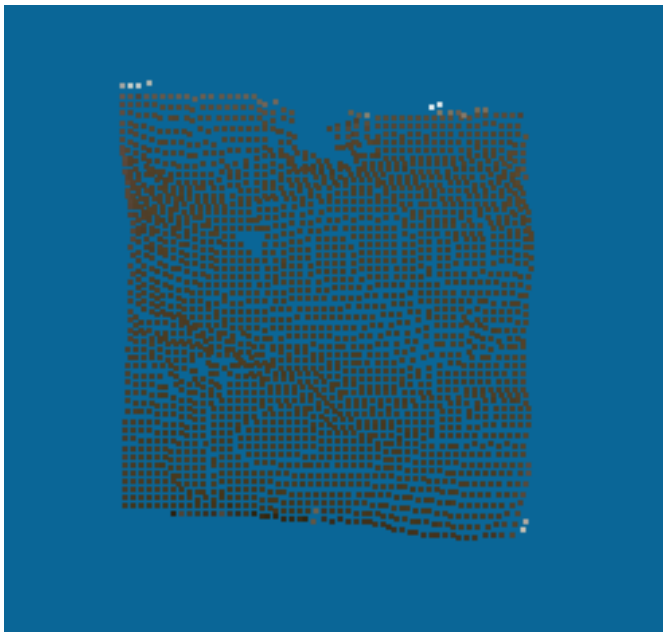


Figure 8.0.2: Processed point cloud image in CloudCompare of the target 0.3m x 0.3m surface at a distance of 0.5m, with target surface the only remaining feature

9.0 Point Cloud Interaction

9.1 Unreal Engine 3D Interaction

Once the point cloud data in Sections 6.0 and 8.0 had been collected, processed and analysed, the next step in the process was to interact with the point clouds. The main objective of this was to use the results from the data collected in Sections 6.0, 7.0, 8.0, to be able to efficiently collect new scans that would have optimal point cloud characteristics, such as density and depth accuracy, and know the accuracy and precision of the collected scans was within a viable threshold. These subsequent scans could then be used to identify and select the apples in a 3D environment and provide accurate 3D coordinates.

9.2 Unreal Engine 3D Interaction

As mentioned above in Sections 6.0-8.0 , CloudCompare was first used to import and process the raw point clouds. The next step was to decide on another software that provided an intuitive interaction with a 3D environment, and was able to support point

cloud import and interaction, utilizing features such as collisions. The two main options based on [64] and discussed above in Section 1.3 were Unity and Unreal Engine.

Based on the evaluation in Section 1.3 it was decided that as a relatively new game engine user and for the purposes that it was required for, Unreal Engine would be optimal. As described by [64], it is an open and advanced real-time 3D creation platform which gives creators across industries the freedom and control to deliver cutting-edge content, interactive experiences, and immersive virtual worlds. Using Unreal Engine, a template environment was first set up. This basic environment can be seen below in Figure 9.0.1. The next step was to enable the UE4 Point Cloud Plugin, which allowed import and interaction of the following filetypes seen and described in Table 9.0.1.

Depending on the filetype needed, point clouds could be exported from CloudCompare as .las, .laz or .e57 files. For this project, it was decided to use .e57 files in Unreal Engine, as this was found to be the most efficient filetype for this application [66] . With the plugin installed and the .ply files being exported as .e57 files, they could now be imported into the Unreal Engine environment. Figure 9.0.2 below shows an image of a processed pointcloud from Section 7.0 in the Unreal Engine environment.

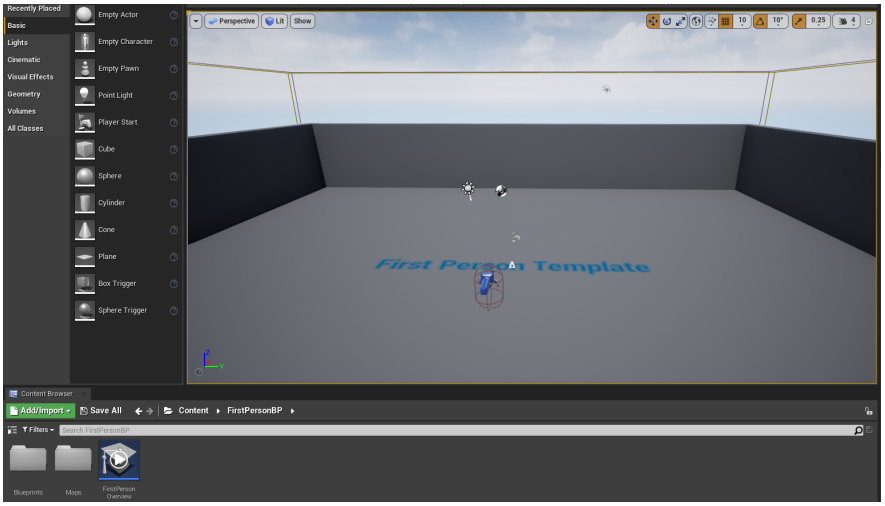


Figure 9.0.1: Blank first person Unreal Engine environment

Now that the point clouds could be imported and visualised, the next step was to enable a form of interaction with the target point cloud. The first step in this process

Unreal Engine 4 Pointcloud Plugin Filetypes	
Filetype	Description
.xyz, .pts, .txt	General types of ASCII point cloud file formats that can contain either: • Point coordinates (X Y Z for each point, expressed in meters)[65]. • Point coordinates plus colors, (X Y Z R G B for each point)[65].
.las, .laz	LAS is a public file format for the interchange of three-dimensional point cloud data between users. Although developed primarily for the exchange of LiDAR point cloud data, this format supports the exchange of any three-dimensional X, Y, Z tuplet. This binary file format is an alternative to proprietary systems, as well as to generic ASCII file interchange systems, and is widely used. LAZ files are compressed LAS files. They are much smaller, but their import is also proportionally slower. Unreal Engine supports 8-bit, 12-bit, and 16-bit LAS / LAZ files[65].
.e57	E57 is a compact, open-source file format that stores and exchanges 3D imaging data such as point clouds, images, and metadata[65].

Table 9.0.1: Comparison between compatible filetypes used for point clouds in Unreal Engine

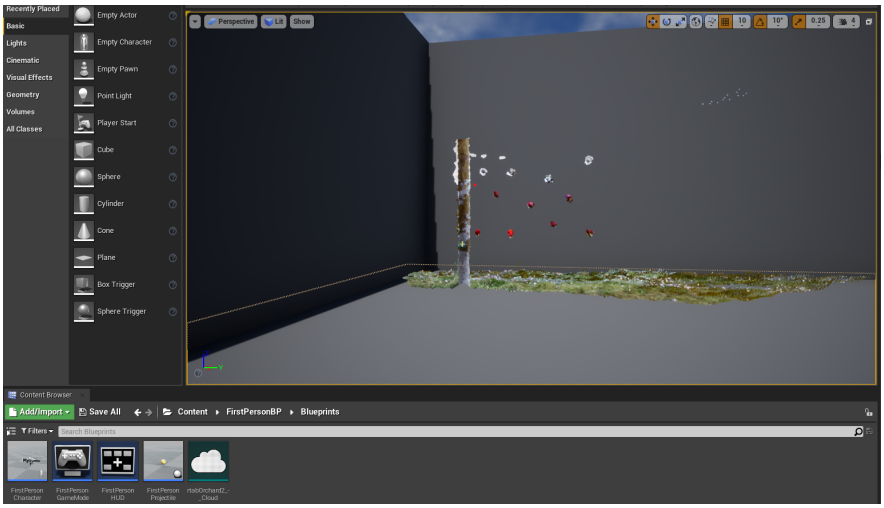


Figure 9.0.2: Unreal Engine environment with point cloud imported using the point cloud plugin

was to build collisions on the point cloud model, which was done using the point cloud plugin. To optimise these collisions, the largest gap between neighboring point clouds was found, and the collision size for each point was made to this size. This ensured that collisions would occur as close to each individual point as possible, whilst not allowing collisions between points to be missed, resulting in the most accurate interaction and point selection environment. Once collisions were set up, the next step was to be able to select a point in the environment, and eventually be able to output the 3D global coordinate of this point. An example of the blueprint structure used to implement this process can be seen below in Figure 9.0.3 .

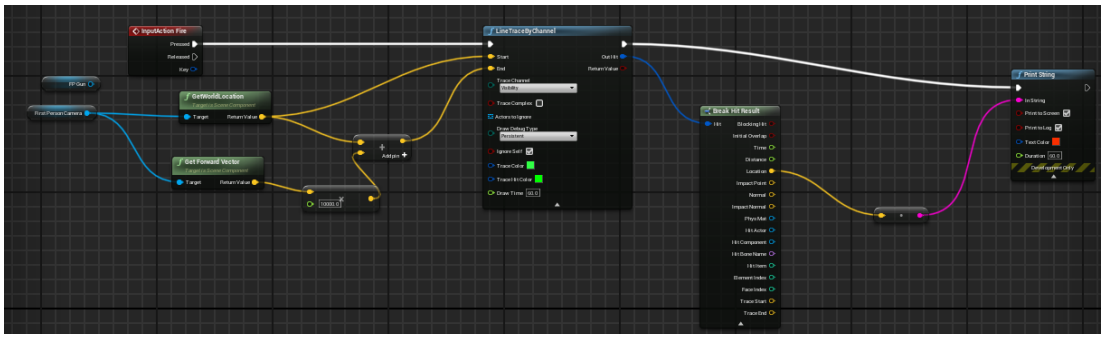


Figure 9.0.3: Sample blueprint showing the process of implementing a line trace and returning the collision location coordinates

This blueprint essentially takes an input, in this case 'mouse left click', acquires the

world location and combines this with the forward vector, resulting in the location of the collision, which is the intersection between points. A line trace is also implemented, projecting a line from the first person model to the target point. This trace is set to be coloured green, however upon target collision, the trace changes to red. Figure 9.0.4 below demonstrates this line trace and hit point. Once the point has been selected by clicking, the location is also output to the screen, as well as being logged in the output log file. Figure 9.0.5 below shows the line traces, hit points and coordinate outputs on the experimental orchard setup point cloud obtained in Section 7.0.

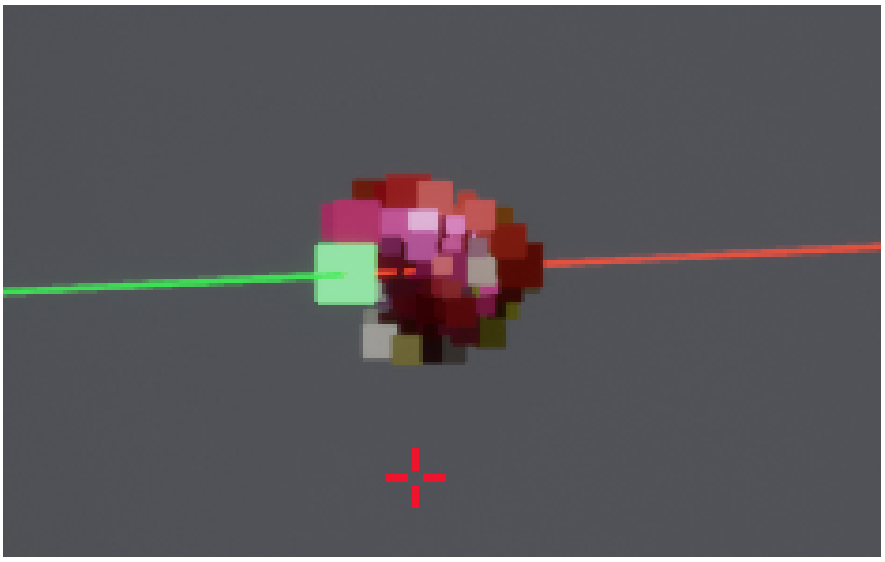


Figure 9.0.4: Image showing the initial line trace in green, the collision point, and the subsequent line trace after the collision, which is red

Now that the foundation had been built for acquiring the 3D position of objects in UE4, this could be used to obtain data for analysis and comparison to the data obtained in Section 7.0 , comparing the spatial location results obtained. Using a new point cloud of the experimental setup used in Section 7.0 , the positions of each apple were logged by clicking on their centre of mass, giving the global coordinates. These global coordinates were then translates to local coordinates relative to each apple by calculating the difference. This experiment was undertaken 10 times to ensure that the results were valid and any outliers identifiable. Table 13.0.1 in Section 13.1 shows the results of this experiment. In addition to this experiment, in order to quantify the

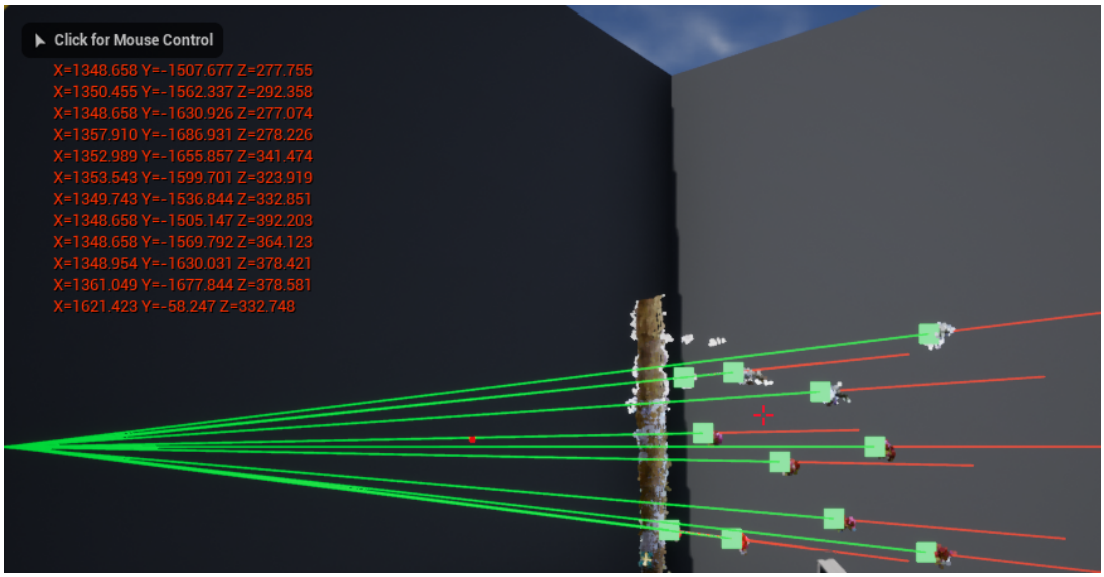


Figure 9.0.5: Image showing the line trace, collision point, and resulting coordinates of the target objects

error of the point cloud data in this environment, the same experiment was undertaken with the point cloud apples substituted for the virtual 3D scanned apples set at known locations, in order to compare the positional accuracy between the two sets of data collected. The results for this data and the comparison with the point cloud data can be seen in Section 13.1 With the base environment now set up in UE4, the next step was to create an environment suitable for interaction in Virtual Reality. As discussed above in Section 1.3, VR interaction is another highly valuable form of interaction, providing a very immersive experience for the user.

9.3 Unreal Engine Virtual Reality Interaction

To set up the VR environment in UE4 was a similar process to setting up the normal virtual environment. The first step was to set up a new blank VR template in UE4. With this blank template now set up, the next step was to enable the UE4 point cloud plugin. With the plugin enabled, point cloud files could then be imported. For initial testing, the same point cloud file as used in Section 7.0 was imported, which can be seen below in Figure 9.0.6. As with the testing in the regular non-VR environment, a blueprint needed to be developed to interact, extract and display the positional data

of the target points, this time using the motion controllers in VR. Figure 9.0.7 below shows the blueprint set up on the VR pawn model to allow this.

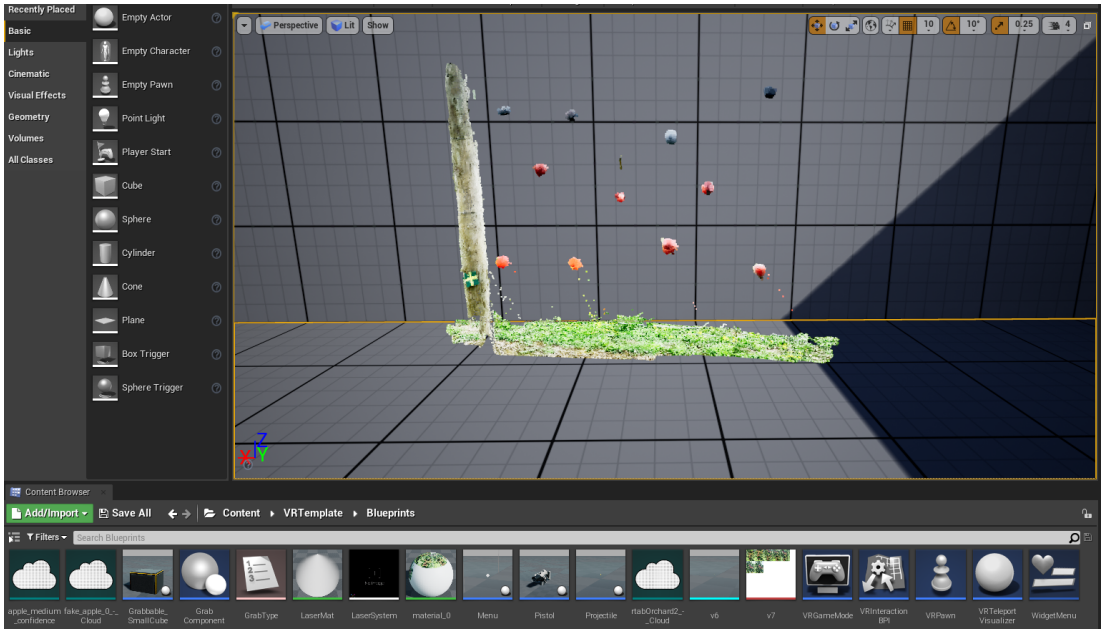


Figure 9.0.6: Processed point cloud of experimental setup obtained in Section 7.0 imported into the UE4 VR environment

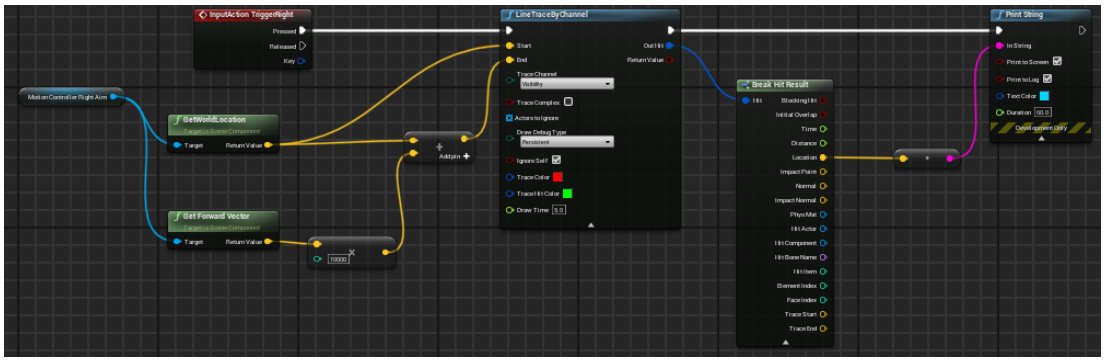


Figure 9.0.7: Sample blueprint showing the process of implementing a line trace and returning the collision location coordinates in a VR environment

Although the general process for constructing the blueprint in VR is similar to Section 9.2 above, there were multiple differences that needed specific VR functions in order for the blueprint to work effectively. The first of these differences that needed to be addressed was to create a reference aim, as in VR there is no mouse cursor to use as an aiming reference. This was done by creating a line trace from the VR controller.

The first step to create this trace was to set up an event tick, which results in the event getting called every frame. In this case we want the event to be the line trace, so this event tick is linked to the input of the line trace function.

The next step of implementing this was to find the current position of the controller, and therefore the starting point of the line trace. This was done by calling the world location and the forward vector, then using the vector addition function to get the sum of these vectors, resulting in the updated position. This was then input into the line trace by channel function, with the trace channel set to visible. The result was a straight visible line emitting from the end of the controller. Figure 9.0.8 below shows the resulting line trace, creating an aiming reference for the user.

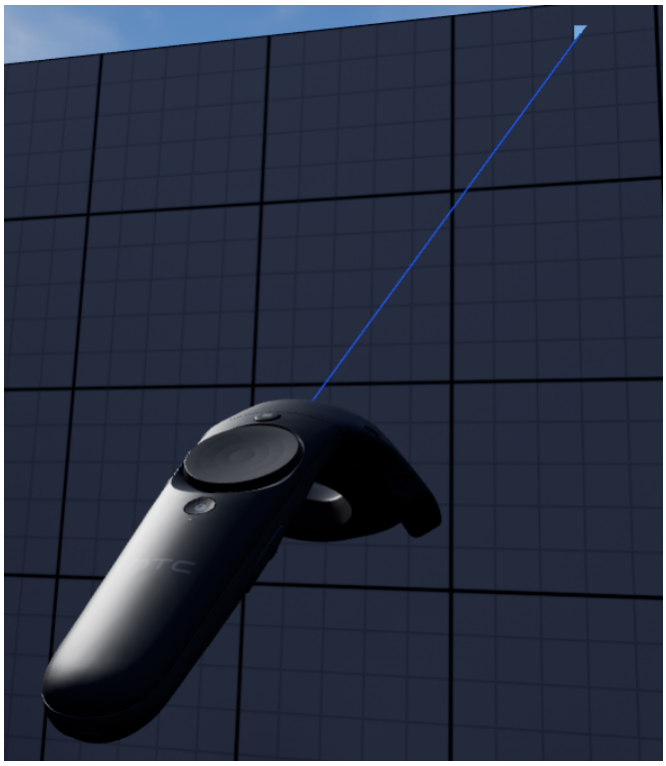


Figure 9.0.8: VR motion controller with constant line trace for aiming

Now that the aiming trace had been generated, a collision could be created that would return the spatial data information. As in Section 9.2, a line trace using the same method was used to log the spatial coordinates of the collision, with the main difference being that the motion controller was used as the object reference. The same

transform to acquire the starting location of the trace, using the current position of the motion controller that was used for the aiming trace was implemented, however there was no event tick enabled for the collision trace. The resulting line trace can be seen below in Figure 9.0.9.

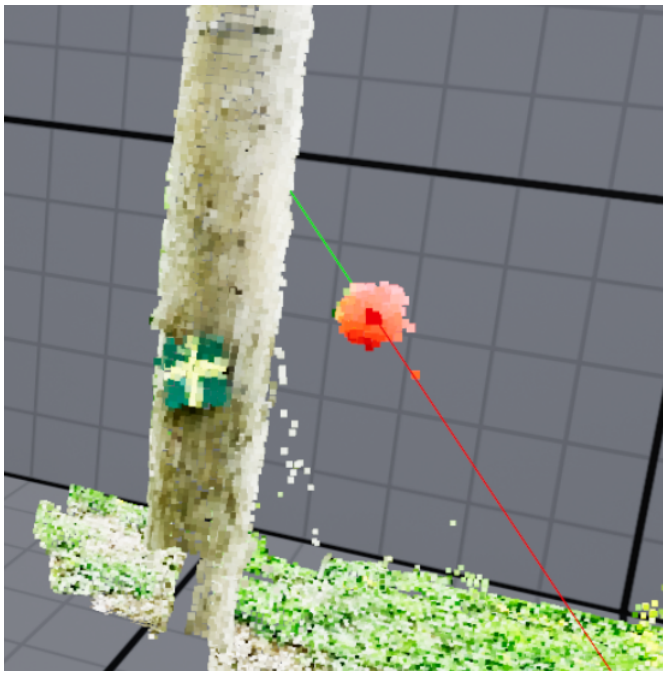


Figure 9.0.9: Resulting line trace after collision with target in VR

With the VR environment now set up, the same experiment that was undertaken in Section 7.0 could be performed. The spatial coordinates for each apple point selection were recorded, with the experiment being undertaken 10 times. The results from this experiment can be seen in Table 13.0.5, Section 13.2. As an advancement on assessing the interaction with point clouds, the following stage was to conduct an experiment to assess and quantify the factor that the human input has on the results, which would also help validate the accuracy of the point cloud scans by being able to quantify one of the error sources in the measurements. As in Section 9.2, the same setup as used in the 3D virtual environment was adapted to be used in the virtual reality environment, using the 3D scanned apples as the target objects in order to eliminate any potential sources of error and inaccuracies from the point cloud scan. Creating a blank UE4 project, 12 of these apples were set up at known global coordinates, providing ground

truth locations for the centres of the apples. This setup in the virtual environment can be seen below in Figure 9.0.10 . This process was undertaken 10 times to form a valid dataset. The results from this experiment can be seen in Section 13.2.

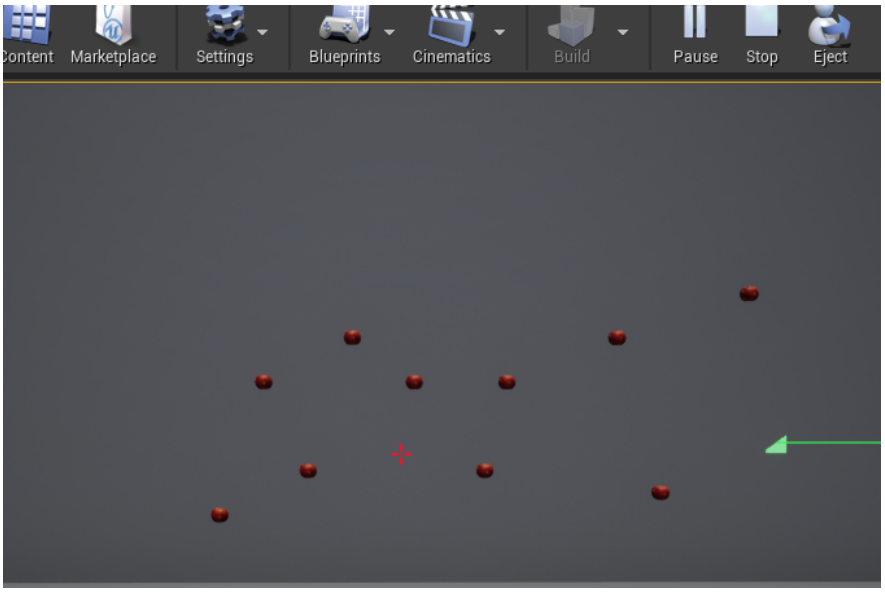


Figure 9.0.10: 3D scanned apples set up in the VR environment in order to compare results to point cloud interaction data

Results

10.0 Depth Accuracy

Data was collected in both indoor and outdoor environments over the distance range of 0.5m-2.0m at intervals of 0.25m, as described above in Section 6.0. Each scan was repeated three times at each distance. This data collected gave the raw results of each data point's depth position, in this case distance z. In order to usefully interpret this data, it needed to be processed to better visualize the distribution and calculate the mean, variance and standard deviation of each data set in order to quantify the accuracy of the camera over the measurement range. Using the process described above in Section 6.0 these attributes were calculated and processed, with the resultant data for both indoor and outdoor measurements being displayed below in Figures 10.0.1 and 10.0.2 respectively.

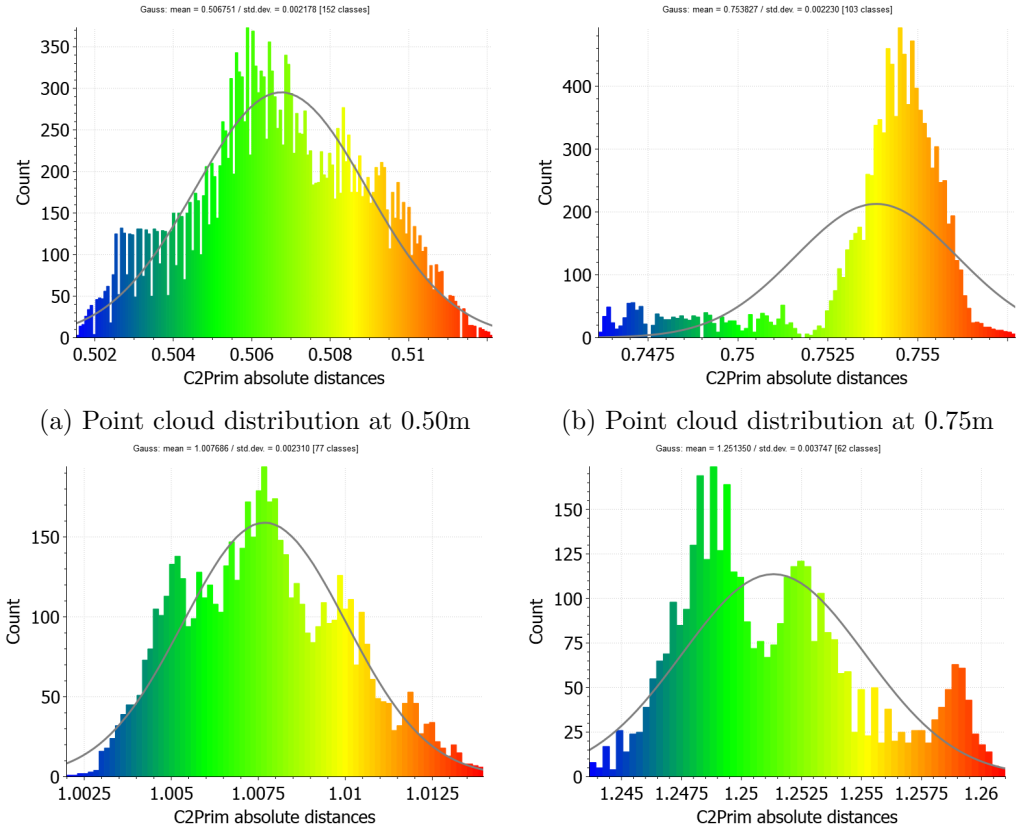
Figure 10.0.3 below shows the comparison between the standard deviations calculated for the indoor and outdoor data.

As well as looking at the standard deviations for both indoor and outdoor data, the mean distance values of each dataset were also analysed. The average measured distance of the three replicate data sets for each distance measurement was calculated, with the residual then being calculated for the indoor and outdoor data. The results from this can be seen below in Figure 10.0.4.

As expected, it can be seen that in general, as the distance for each dataset increases, so does the value of the standard deviation, meaning that the depth data is more accurate and precise at closer distances due to the lower variation in point distances.

From this data, it can be seen that the standard deviation of the indoor data sets is much less than the standard deviation of the outdoor data sets, with the standard deviation of the indoor data at 2.0m being 0.004m, compared to 0.078m for the outdoor data. As mentioned before in Section 5.0, this could be attributed to a multiple factors. As [62] found, sunlight had an impact on the accuracy or active IR depth cameras due to the interference of the IR pattern and the sunlight, which decreases the precision of the depth accuracy due to the IR grid not being fully captured by the IR stereo vision of the RGBD camera. As well as the sunlight potentially disrupting the projected IR pattern, the sunlight can also directly affect the IR sensor that is part of the RGBD camera hardware, again leading to less accurate data.

For further research in this area, it would be beneficial to try both different RGBD



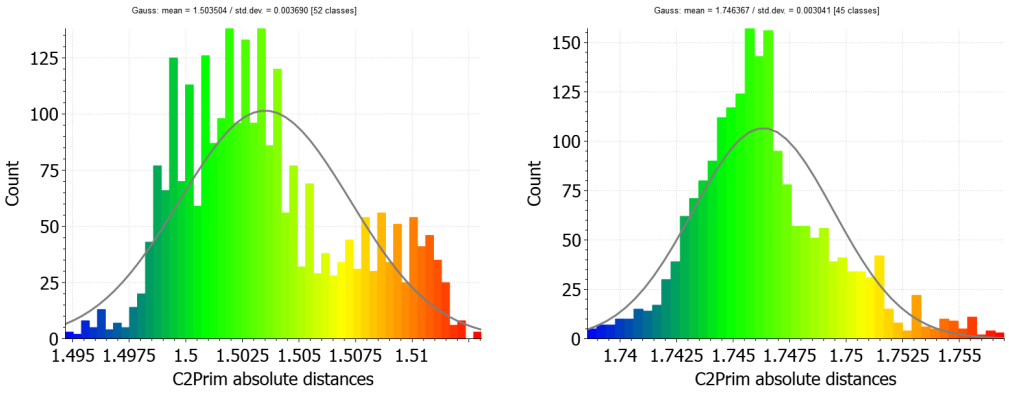
(a) Point cloud distribution at 0.50m

(b) Point cloud distribution at 0.75m

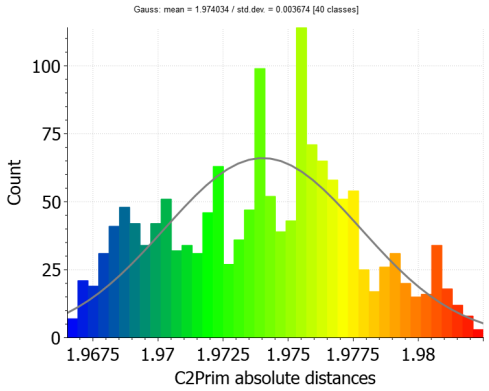
(c) Point cloud distribution at 1.00m

(d) Point cloud distribution at 1.25m

Figure 10.0.1: Distribution of point cloud at each data measurement in an indoor environment



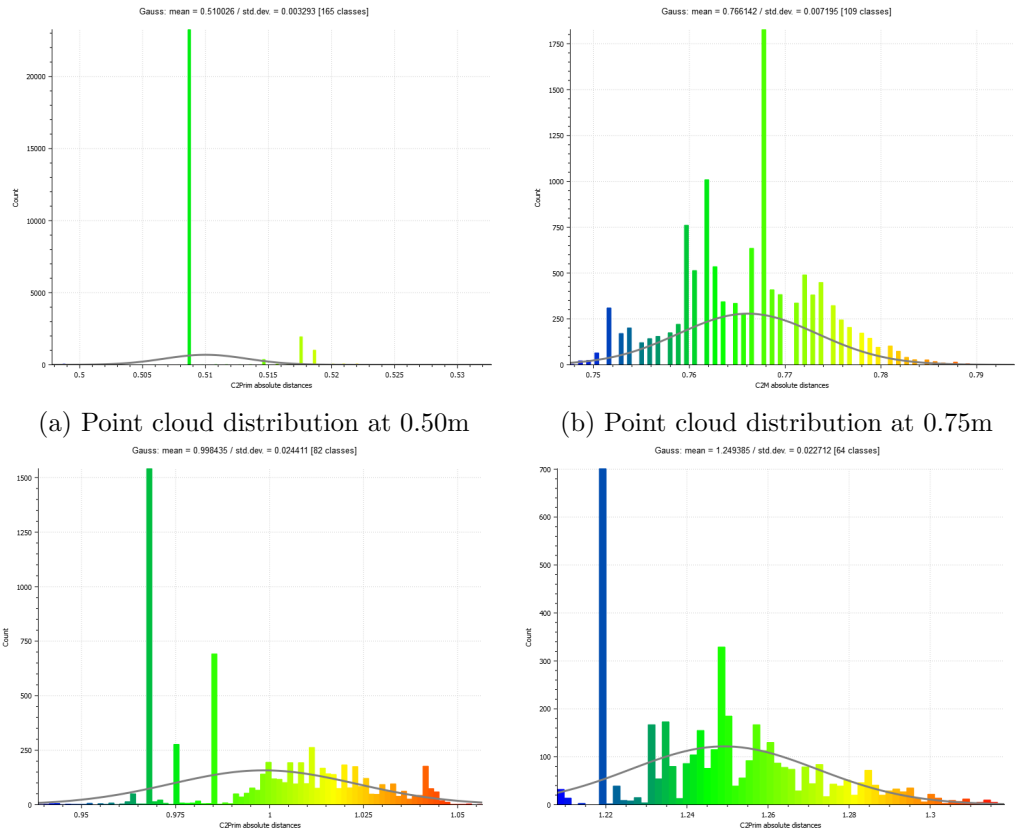
(e) Point cloud distribution at 1.50m (f) Point cloud distribution at 1.75m



(g) Point cloud distribution at 2.00m

Figure 10.0.1: Distribution of point cloud at each data measurement in an indoor environment

camera hardware, as well as other forms of 3D sensing in order to compare the point variation and standard deviation. In the scope of this research, we were limited to the Realsense RGBD camera, which was a suitable choice



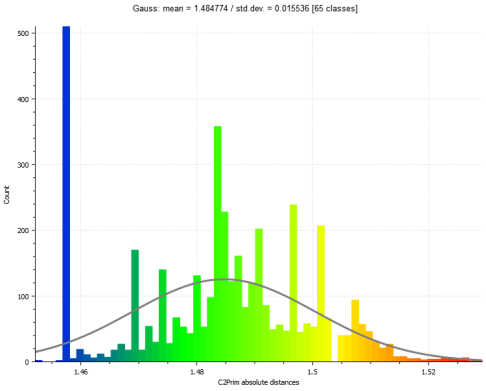
(a) Point cloud distribution at 0.50m

(b) Point cloud distribution at 0.75m

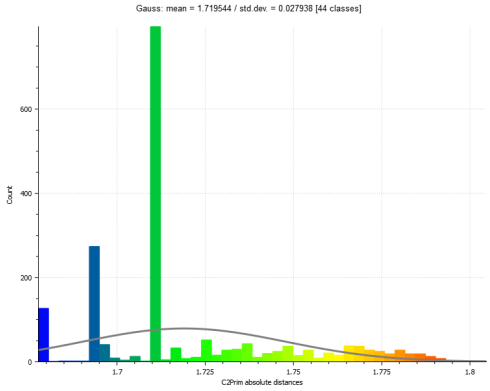
(c) Point cloud distribution at 1.00m

(d) Point cloud distribution at 1.25m

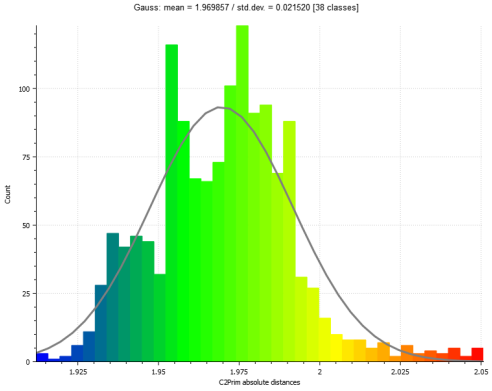
Figure 10.0.2: Distribution of point cloud at each data measurement in an outdoor environment



(e) Point cloud distribution at 1.50m



(f) Point cloud distribution at 1.75m



(g) Point cloud distribution at 2.00m

Figure 10.0.2: Distribution of point cloud at each data measurement in an outdoor environment

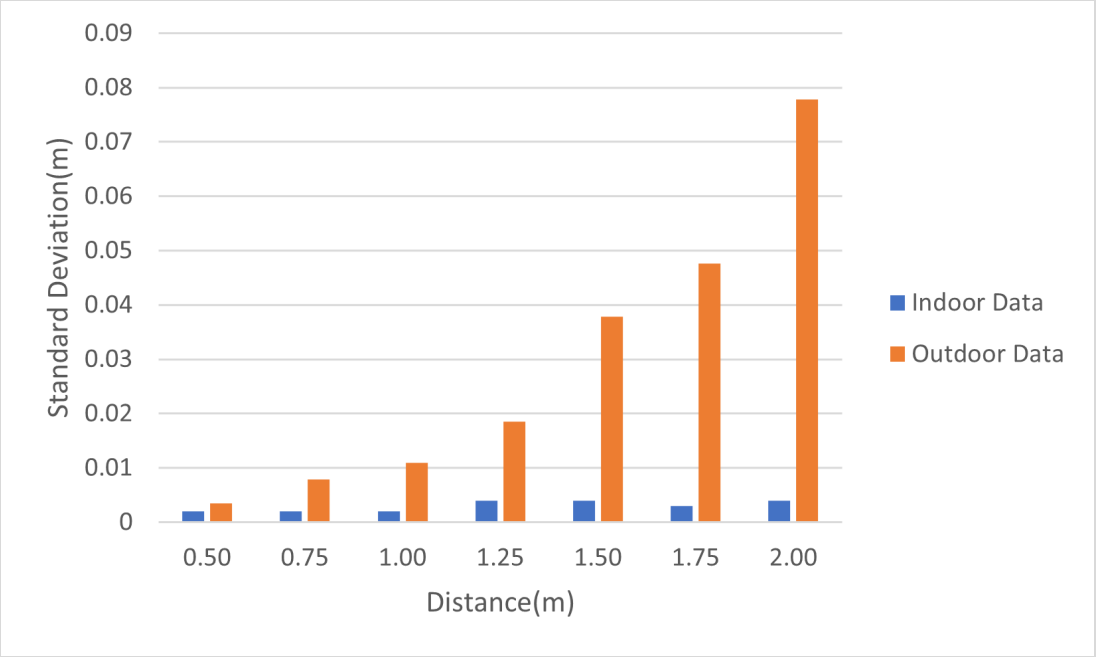


Figure 10.0.3: Comparison of standard deviation in indoor and outdoor conditions

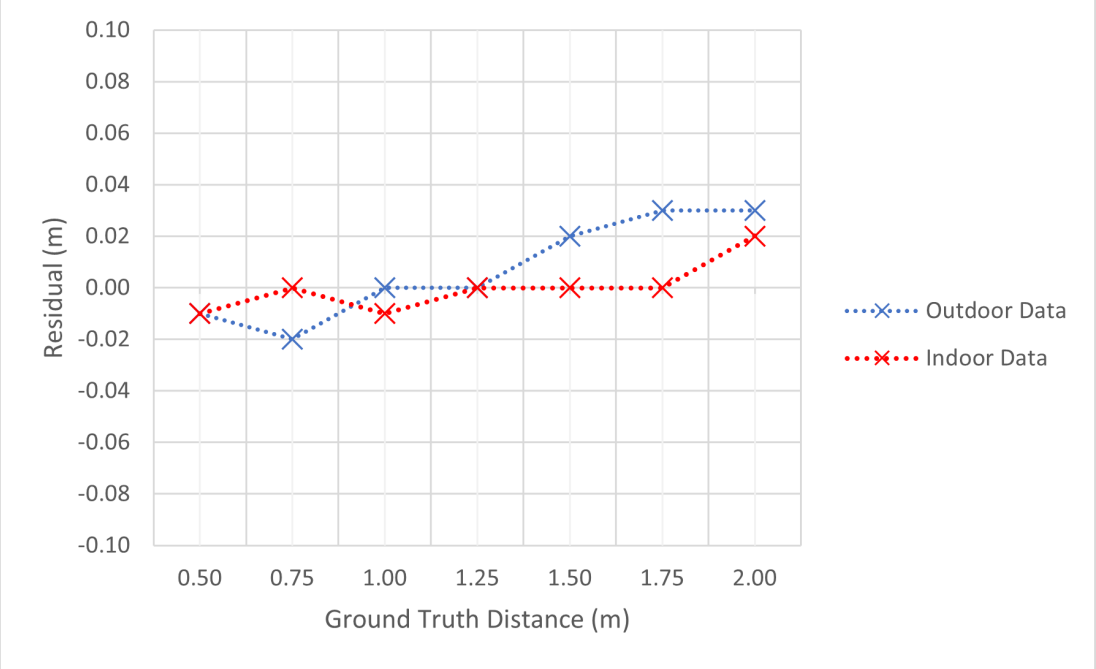


Figure 10.0.4: Comparison of residuals for indoor and outdoor distance measurements

11.0 2D Position Accuracy

As described above in Section 7.0, data using the RGBD camera was collected in order to validate the 2D positional accuracy. Effectively, this is the accuracy of the points in the x,y plane. The reason that this was separated from the 3rd dimension results, which was the z plane data mentioned above, was due to the nature of the apple shapes and testing environment, which would inherently give variation in the datapoint z values, making it more difficult to determine the accuracy and quantify the error of this measurement. Using the experimental setup described in Section 5.0, data was collected at a distance of 1.5m, as based on the data collected and analysed in Section 6.0, the point cloud density was high enough to be viable, the field of view of the camera captured the entire test setup, as well as having an acceptable standard deviation in depth data as determined above. With the data collected, it was imported into CloudCompare and processed to exclude any stray points by removing any data points with a distance greater than 0.05m between any other point. Using the measure function in CloudCompare, the centre to centre distances between each apple could be calculated.

Table 11.0.1 below shows the summary of the measurements of the experimental setup in CloudCompare, as well as the ground truth data that was collected by manual measurement of the centre to centre distances of the test apples. The measurements were conducted 3 times each, with Table 11.0.1 showing the averages of these distances.

This testing showed the error and accuracy of the data collected using the RGBD camera, however it also includes the error that is contributed to the human input in measuring the data points the CloudCompare virtual environment. In order to isolate this error to better quantify just the error in the data, similar data collection was done in CloudCompare but the variable of the data error was eliminated. This was done by generating an array of spheres of similar dimensions the the apples used in the point cloud. The spheres in this array were set at specific x,y coordinates, with the same measure function in CloudCompare then being used to calculate the centre to centre

Apple Number	**Actual Distance(mm)**	**Cloudcompare Measure Distance(mm)**	**Absolute Difference(mm)**
1	570	574	4
2	700	679	21
3	570	576	6
4	600	595	5
5	700	710	10
6	600	588	12
7	600	609	9
8	480	474	6
9	630	650	20
10	680	650	30
11	660	664	4
		Average Error	11.54mm

Table 11.0.1: Centre to centre apple point cloud distances using CloudCompare

distances between spheres. Table 11.0.2 below shows the results of this testing. Again, this experiment was conducted 3 times, with the average measurements being shown in Table 11.0.2.

With this data, the human input error could be quantified and taken out of the point cloud measurement data error by subtracting the average sphere error from the average point cloud measurement error, which is shown below in Table 11.0.3.

From these observations, it can be seen that the error in the point cloud data is larger than the error introduced via the input of human interaction. However, even though the inherent data error makes up the majority of the total error, it is still well within the error range for a robotic manipulator to perform a successful grasping action when given these spatial coordinates[38]. Using this method of manual localization shows that the source of error to be investigated in future research would be the accuracy of the data acquisition, as at this stage this error outweighs the human error contribution. One such was would be to potentially investigate different methods of 3D sensing, as well as investigating different hardware options for each method. In the scope of this research this was limited to the Realsense RGBD camera, however further data on other hardware options would be an important consideration in future work.

Sphere Number	**Actual Distance(mm)**	**Cloudcompare Measure Distance(mm)**	**Absolute Difference(mm)**
1	560	565	5
2	560	565	5
3	560	565	5
4	560	563	3
5	560	562	2
6	560	558	2
7	560	568	2
8	560	563	3
9	560	566	6
10	560	561	1
11	560	560	0
12	560	562	2
		Average Error	3mm

Table 11.0.2: Centre to centre virtual sphere distances using CloudCompare

Average Point Cloud Error(mm)	**Average Sphere Error(mm)**	**Average Point Cloud Data Error(mm)**	**Percentage Human Error (%)**
10.94	2.97	7.97	27.17

Table 11.0.3: Comparison of error contributions in localisation

12.0 Point Cloud Density

As described above in Section 8.0, testing was carried out to determine the point cloud density of data captured from the RGBD camera over varying distances. Due to the way that this experiment was designed and set up, it was also possible to undertake testing in both an indoor and outdoor environment in order to compare the results and identify any impact that outdoor factors may have had. The testing was conducted from a distance of 0.5m, to a maximum distance of 2.0m at 0.25m increments. The results from the outdoor environment testing can be seen below in Table 12.0.1.

Distance (m)	**Raw Density 1**	**Raw Density 2**	**Raw Density 3**	**Average Raw Density**	**Average Density (points/m^2)**
0.50	6466	7348	7389	7068	78530
0.75	4250	4352	4090	4231	47007
1.00	2487	2513	2816	2605	28948
1.25	1685	1701	1573	1653	18367
1.50	1231	1164	1263	1219	13548
1.75	907	904	905	905	10059
2.00	676	723	720	706	7848

Table 12.0.1: Point cloud density at measured data points in outdoor conditions

Table 12.0.2 below shows the results from undertaking the same experiment under indoor conditions.

Distance (m)	**Raw Density 1**	**Raw Density 2**	**Raw Density 3**	**Average Raw Density**	**Average Density (points/m^2)**
0.50	23092	23091	23087	23090	184131
0.75	10592	10561	10557	10570	84290
1.00	5917	5922	5904	5914	47164
1.25	3819	3819	3819	3819	30455
1.50	2658	2660	2659	2659	21204
1.75	1985	1989	1999	1991	15877
2.00	1523	1527	1528	1526	12169

Table 12.0.2: Point cloud density at measured data points in indoor conditions

Figure 12.0.1 below illustrates this relationship between density and distance in

both and indoor and outdoor environment.

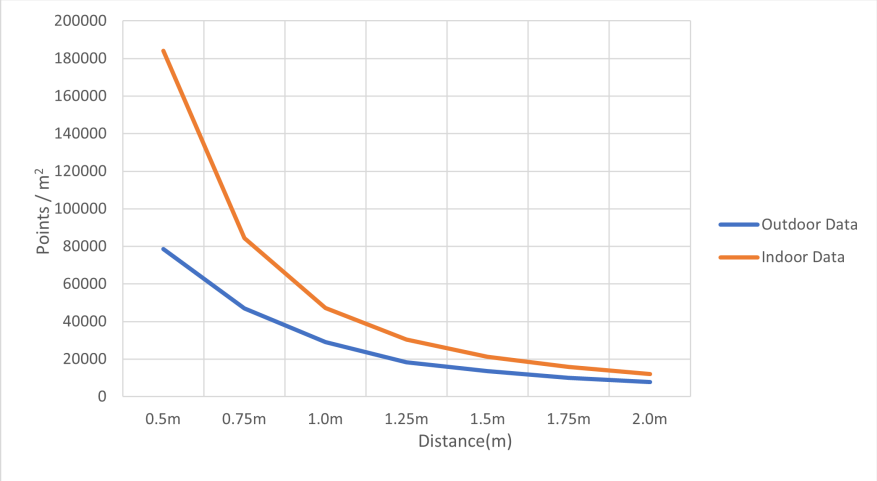


Figure 12.0.1: Point cloud density as a function of distance in an outdoor and indoor environment environment

As expected, it can be seen that as the distance increases, subsequently the point density decreases, and this is the ongoing trend, however it is not a linear relationship. In order to compare this trend, the hypothetical trend for point density based on the Realsense D435i FOV parameters could be calculated. With both the horizontal and vertical FOV angles known, this could be used with the distance from target measurement to calculate the hypothetical pixel coverage over a set area at varying distances, specifically the same range of distances used in the experiment above. Using some basic trigonometric functions, Equations 12.0.1 and 12.0.2 below were developed and used to calculate the horizontal and vertical FOV distance respectively.

$$hFOV = 2(tan(\frac{\beta}{2})d) \tag{12.0.1}$$

$$vFOV = 2(tan(\frac{\alpha}{2})d) \tag{12.0.2}$$

Where β is the horizontal FOV angle, α is the vertical FOV angle and d is the distance to target. With these two measurements now known, the proportion of coverage of an object of a known size relative to the FOV measurements could be calculated.

Effectively, the target object remains the same size, with the FOV area increasing as d increases. Equation 12.0.3 below was used to calculate this percentage of coverage independent to resolution, which can then be used in conjunction with the known depth resolution to calculate the number of points/m^2 over the distance range. The native resolution that was used with the Realsense camera was 848 x 480, which is the value that was subsequently used to calculate the theoretical point density. Figure 12.0.2 below shows the comparison of the indoor and outdoor density data and the theoretical density data.

$$PercentCoverage = \frac{(d_h \times d_v)}{(vFOV \times hFOV)} \tag{12.0.3}$$

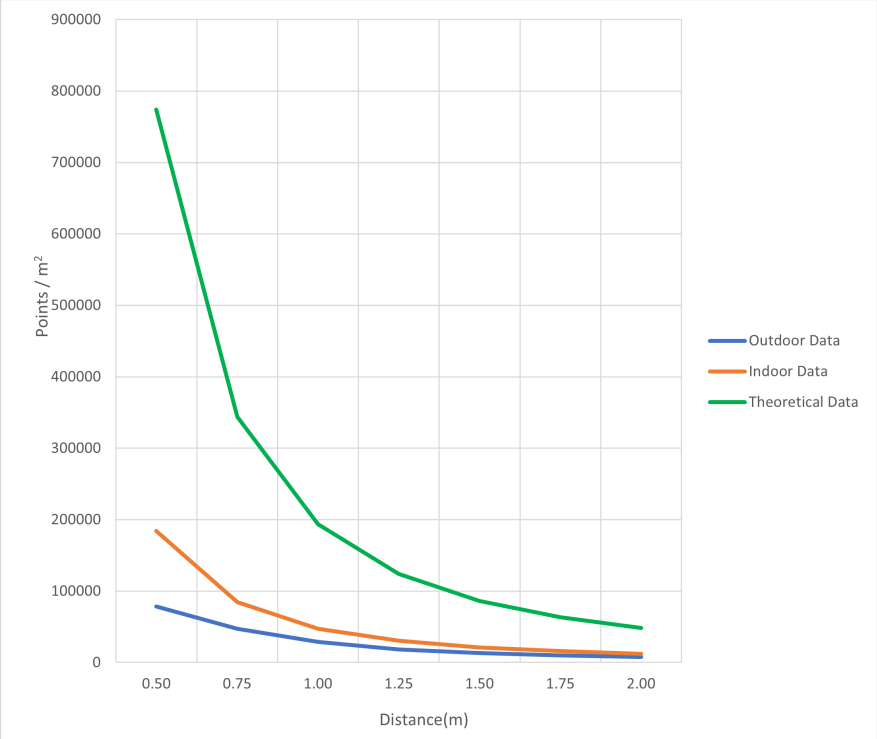


Figure 12.0.2: Point cloud density as a function of distance in an outdoor and indoor environment environment compared to the theoretical calculated density

In order to verify this data, the relationship between the theoretical data and the collected data needed to be analysed. As can be seen by the trend of the data in Figure 12.0.2, it would be expected that all sets of data follow power law. In order to confirm

this relationship, it would be expected that a log-log plot of the indoor and outdoor data would produce a linear result. Figure 12.0.3 below shows this data.

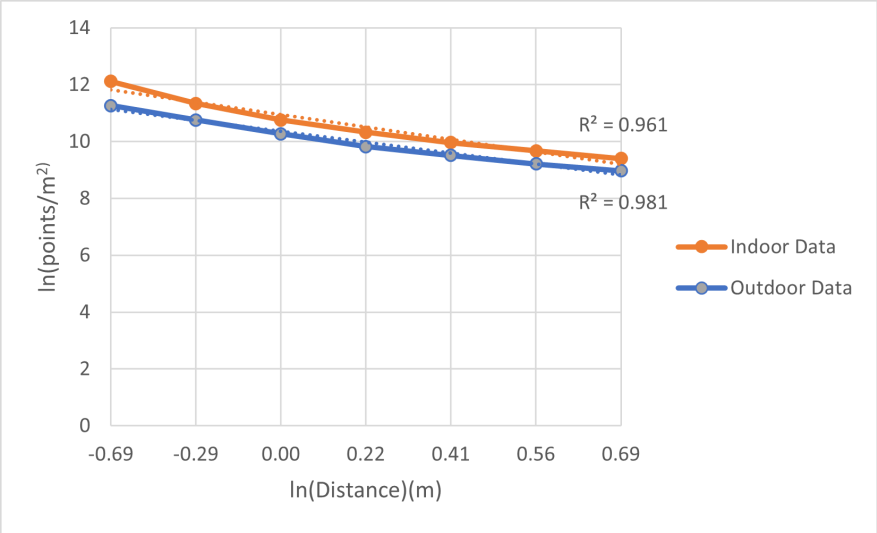


Figure 12.0.3: Log-log graph of experimental point cloud data collected in an indoor and outdoor environment

With this data plotted, the R^2 value for each set of data could be calculated. These R^2 values, being 0.961 and 0.981 for the indoor and outdoor data respectively, proves that the data collected does follow power law, as these values are very close to the R^2 maximum of 1 when using a linear trendline.

Although the data collected does follow the same trend as the theoretical data, both the indoor and outdoor data density values are well below the theoretical maximum. As stated above in Section 1.3, the main factor in this difference can be attributed to the interference from sunlight[8]. Another potential factor for this decrease in density, as stated in [67], is the surface that is being scanned. Typically reflectance of the target surface plays a part in Lidar scanning[62], however the type and colour of the surface can also play a part in the density and quality of an RGBD point cloud. For example, most RGBD cameras have a difficult time in accurately providing depth information on dark or reflective surfaces[68], as well as surfaces that are uniform in colour. A combination of these factors can be attributed to the disparity between the theoretical density and the observed density, particularly the external factors such as sunlight which

is demonstrated in Figure 12.0.1 as the only variable that changed was the location of experimentation. For future research, multiple independent factors could be assessed, such as target surface roughness, target material and colour in order to further verify and quantify the contributing factors on depth data density.

13.0 Unreal Engine Virtual Environment

13.1 Virtual Environment Results

As described above in Section 9.2, after analysis was done in CloudCompare, Unreal Engine was also used to interact with the captured data. The following experiment was very similar to the experiment in Section 7.0, however it was a necessary step in order to verify that the data in Unreal Engine was producing accurate results, as this was the virtual environment that would be used with the Virtual Reality interaction.

13.1.1 Unreal Engine Virtual Environment Point Cloud Interaction

Using the same point cloud data set as used in Section 11.0, it was imported into Unreal Engine. Once the point cloud was imported, the Unreal Engine blueprint that was created to output X,Y,Z coordinates based on mouse click position was used to find the centre to centre distances between the apples. Unlike the 'measure' function in CloudCompare which calculates the distance between two points in 3D space, the data from Unreal Engine is output as 3D coordinates. Due to some variation in the depth distances when picking datapoints, this also needs to be considered in a 3D environment. In order to get the most accurate distance, Equation 13.0.1 below was used to calculate the distance between points in three dimensions, using the X, Y, Z coordinates of each selected point as the inputs. Table 13.0.1 below shows the results from this experiment.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{13.0.1}$$

The average error between the actual ground truth data and the measured distance in a virtual environment in Unreal Engine was found to be 18.8mm, which can be compared to the 11.0mm average error found in the experiment conducted in CloudCompare in Section 11.0 using 3D apple models. Although there is quite a large difference between these two values it is still within the tolerance described by [38] for a robotic manipulator to grasp the object, which in this case would be approximately 50% of the

Apple Number	**Actual Distance (mm)**	**Measured Distance (mm)**	**Absolute Difference (mm)**
1	570.0	580.4	10.4
2	700.0	699.5	0.5
3	700.0	702.6	2.6
4	570.0	593.1	23.1
5	600.0	630.1	30.1
6	630.0	613.5	16.5
7	480.0	452.9	27.1
8	480.0	446.0	34.0
9	630.0	649.1	19.1
10	680.0	659.2	20.8
11	660.0	637.9	22.1
		Average Error	18.8 mm

Table 13.0.1: Measured point cloud centre to centre distances in Unreal Engine

average diameter of the target apples, which based on [69] is 72mm. This means that the error range of a robotic manipulator in this application would be approximately 36mm, with the errors from both above experiments being well within this error range.

Apple Number	Actual Distance (mm)	Measured Distance (mm)	Absolute Difference (mm)
1	860.2	858.0	2.2
2	1050.0	1030.0	20.0
3	1346.3	1329.7	16.6
4	1581.1	1574.9	6.2
5	2926.1	2885.2	41.0
6	1118.0	1122.6	4.6
7	2000.0	2010.0	10.0
8	2015.6	2014.3	1.3
9	1820.0	1805.0	15.0
10	1581.1	1559.0	22.1
		Average Error	13.9 mm

Table 13.0.2: Measured virtual apple centre to centre distances in Unreal Engine

Average Point Cloud Error(mm)	Average 3D Apple Error(mm)	Average Point Cloud Data Error(mm)	Percentage Human Error(%)
18.8	13.9	4.9	74.0

Table 13.0.3: Human error input in Unreal Engine measurements

13.1.2 Unreal Engine Virtual Environment Virtual Apple Interaction

As with the experiment in Section 11.0, this data had both human error and error from the 3D point cloud itself. In order to isolate this data error, testing was conducted with accurate 3D apple models placed at known coordinates within the virtual environment. The same method as described above was used to compute the distances between mouse click points in the virtual environment. Table 13.0.2 below shows the results of this.

From this experiment, it can be seen that the average error decreases using the 3D scanned apples set up at specific coordinates in the virtual environment. Table 13.0.6 below shows the difference between the two tests.

This shows that for testing in the Unreal Engine environment that the majority of the error in fact comes from the human error, rather than error in the point cloud data

itself. This is useful information as it reinforces the fact that the data collected using the Realsense RGBD camera is accurate, whilst also showing the factor that human error can have in results when making manual measurements in a virtual environment. Knowing that this data is accurate and useful in this environment also means that it can be used in a Virtual reality environment which can be used as a valid comparison in point cloud interaction.

13.2 Unreal Engine Virtual Reality Results

13.2.1 Unreal Engine Virtual Reality Point Cloud Interaction

With the testing in the Unreal Engine environment conducted above in Section 13.1, the next experiment conducted was testing in a virtual reality environment in Unreal Engine in order to compare the accuracy and error of the interaction with the point cloud data. Using the setup described above in Section 7.0 and Section 13.1, a virtual reality environment was created using the same concept as in Section 9.3, with the program outputting 3 dimensional coordinates at the selected points, however instead of clicking on a computer screen, the virtual reality controller was used to select the target points in the virtual reality environment. The first experiment undertaken in virtual reality was done using the same point cloud scan as used above in Section 13.1. Again, these results were then compared to the ground truth measurements. The results from this experiment can be seen below in Table 13.0.4.

From this testing, it can be seen that the average error in the virtual reality environment is much larger than the error in the normal virtual environment, being approximately 42mm compared to the 19mm average error in the regular Unreal Engine virtual environment. This total error is outside the range of approximately 36mm as mentioned in Section 13.1 based on [69]. As in Section 13.1 above, in order to quantify and compare the proportion of human error and error from the data itself, the same experiment was conducted using the 3D scanned apples positioned at known coordinates. The results of this experiment can be seen below in Table 13.0.5.

Apple Number	Actual Distance (mm)	Measured Distance (mm)	Absolute Difference (mm)
1	570.0	551.6	18.4
2	700.0	730.1	30.1
3	700.0	704.4	4.4
4	570.0	537.5	32.5
5	600.0	558.7	41.3
6	630.0	539.4	90.6
7	480.0	421.9	58.1
8	480.0	380.7	99.3
9	630.0	631.3	1.3
10	680.0	608.1	71.9
11	660.0	640.2	19.8
		Average Error	42.5mm

Table 13.0.4: Virtual Reality Unreal Engine centre to centre point cloud measurements

13.2.2 Unreal Engine Virtual Reality Virtual Apple Interaction

From this data, it can be seen that the average error using a controlled created environment is 29.7mm, which is also larger than the error of 14.0mm using the same experimental setup in the previous experiment conducted in Section 13.1. This shows that the percentage of human error that can be attributed to the total error is approximately 70 %, which is higher than the percentage of human error in a virtual environment as seen in Section 11.0 and similar to the 75 % human error contribution in Section 13.1.1. Although the human error in the virtual and VR environment is larger than in other tested environments, the average point cloud data error remains similar throughout, which helps verify that Unreal Engine is a suitable platform for the data visualisation, whilst identifying the error that can be attributed to human input as an area for further improvement.

From a user perspective, the virtual reality environment is very intuitive, giving the user a three dimensional experience which makes selecting the points easier, however it seems that there is in fact more variability in the positions of the points selected compared to the non-VR environment. One reason that could contribute to this is being able to select any point of the point cloud apple more easily, given the ability

Apple Number	Actual Distance (mm)	Measured Distance (mm)	Absolute Difference (mm)
1	860.2	842.0	18.2
2	1050.0	1020.0	30
3	1346.3	1292.4	53.9
4	1581.1	1563.2	17.9
5	2926.1	2875.8	50.4
6	1118.0	1127.0	9.0
7	2000.0	1960.0	40.0
8	2015.6	1994.5	21.1
9	1820.0	1794.8	25.2
10	1581.1	1549.5	31.6
		Average Error	29.7mm

Table 13.0.5: Virtual Reality Unreal Engine centre to centre virtual apple measurements

Average Point Cloud Total Error(mm)	Average 3D Apple Error(mm)	Average Point Cloud Data Error(mm)	Percentage Human Error (%)
42.5	29.7	12.8	69.9

Table 13.0.6: Human error input in Unreal Engine VR measurements

to effectively see 360 degrees around the apples, with the exception of missing points on the rear side of the apple. This in turn creates more opportunities for points that are not central to the apple to be selected. For example, the user could be viewing the point cloud apple from a 45 degree angle from the side, and select the middle of the apple from that perspective. However, when looking at the apple from a parallel plane, this point is actually on the side of the apple, 45 degrees from the central plane, which in turn increases the point to point distance, which would contribute to a larger error when compared to the ground truth measurements.

Another potential contributing factor in the human contribution of the error in this experiment is from the sensitivity of the line trace from the VR controller. As with any line from a fixed point, with a fixed input movement at the origin, this movement is multiplied the longer the line is. Essentially, for the same angular input movement at the origin, the linear movement of the line trace increases the further away the controller is from the target object, which is based on the basic physics of rotational angle calculation[70]. In practice, this means that the further away the user is from the target object, there is a higher chance for potential error due to the larger movement of the selection point with the same input movement, leading to an increased chance in inaccurate point selection. Figure 13.0.1 below demonstrates this, showing that with the same angular input, the distance from the origin of the collision point increases.

One potential way to improve this factor would be to decrease the sensitivity of the motion controller input, so that for the same input movement, the line trace would move a smaller linear distance than previously, giving the user more control over point selection at further distances. Another more involved potential solution would be for a filter of some description to be applied to the point cloud. One potentially suitable algorithm would be a neural network detection algorithm that would be trained for apple detection, however in practice this would be hard to apply to a point cloud, and would be a good are for further research to be conducted. Another algorithm that could be used is colour based blob detection, which would group points together based on colour and other desired parameters such as density. Using these algorithms, or

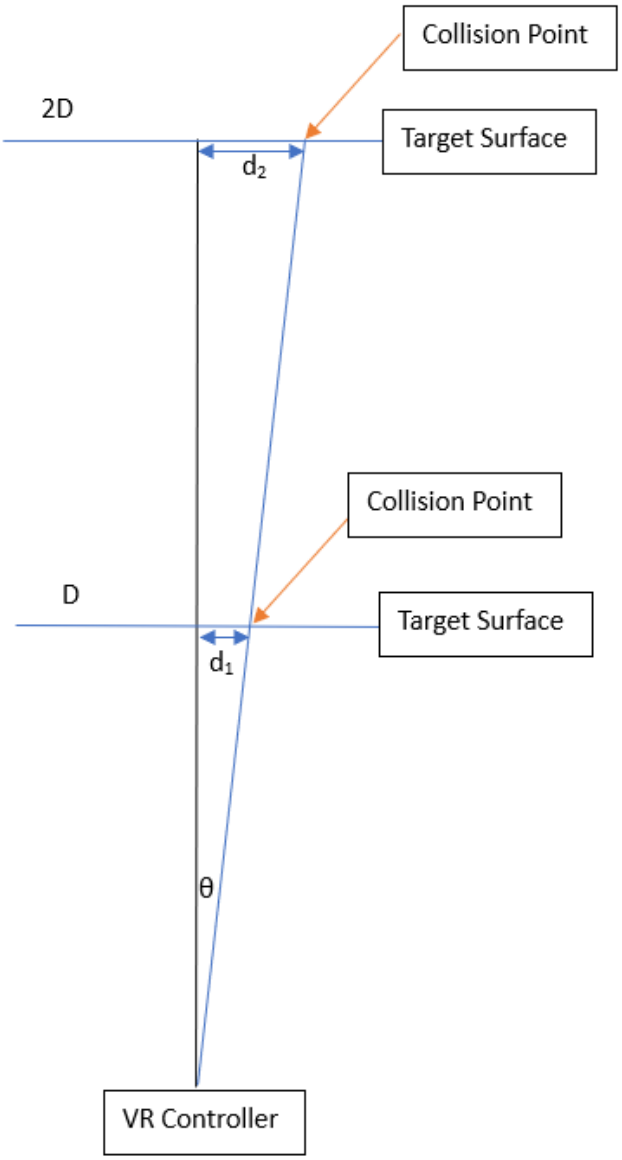


Figure 13.0.1: Diagram demonstrating the increase in collision location relative to distance.

similar, a centroid for the detected 3D shape could be calculated. This centroid would have coordinates which the line trace could be programmed to 'snap' to when it was within a certain distance, meaning that there would be much less chance for human error, as the localization error would come from the centroid calculation rather than human input. Applying an algorithm to the point cloud for apple identification would also be a step towards autonomous harvesting, as this would be a key area of research

for the further advancement of this technology.

Conclusions

Based on testing with both the Realsense RGBD camera, a practical maximum range of 2.0m was found based on the point cloud density of the initial data collected. Although the maximum operating range extends beyond this, one of the reasons that this was decided upon as the maximum data collection distance was that at distances further than this the point density becomes too low to be usable and be able to identify the objects accurately. A secondary consideration for this distance was also based on the environment of a FOPS orchard in which this setup may be further researched and applied, as it has limitations of row width typically being between 2 and 3 metres [7]. The data to support this, as well as the relationship between distance and point density was found in Section 12.0. This shows that the relationship between the data collected and the theoretical data matches very closely, with the main disparity between the data being in the total point density. This lower point density can be attributed to a variety of factors, the main one being interference from sunlight, with the outdoor data having a significantly lower density than the indoor data. To prove that both the theoretical and collected data followed power law, the natural log of the points/m^2 was used to calculate the R^2 value, as the expected result would be to produce straight lines, which is evidenced by the resultant R^2 values.

Following the testing to identify an optimal test range for the camera, further testing of depth data components was undertaken over a range of 0.5m to 2.0m at 0.25m increments. It was found that the accuracy of the points remained within the limits over this range, however there was less variation in data collected at closer ranges, which was observed during the testing and analysis performed in Section 10.0. This

testing shows that the standard deviation increases significantly as distance increases in the outdoor environment, however this disparity is much less when examining the indoor data. As well as the standard deviation of the data being used as a metric for the accuracy of the point clouds, the residuals of the distance measurement were compared, with the outdoor data having a maximum residual of 0.03m, and the indoor data having a maximum residual of 0.02m. From these metrics it can be seen that the spread of datapoints in the outdoor environment is much larger than the indoor data, however the mean values of both datasets are comparable and within a practical range for the intended application of crop localization.

As seen and discussed in Section 4.3, there are few previous studies based on the interaction with point clouds, particularly in a virtual reality environment. As a result, how point clouds would function and appear in a practical virtual reality environment application was largely unknown. Based on the data collected through experimentation, it was found that it is possible for the point clouds to be saved as a compatible file type and imported into the virtual environment. Once the point clouds were built in the virtual environment, they could be interacted with in both a virtual environment and a virtual reality environment, in this case being used to localise the scanned apples in 3D space.

It was found that in both a virtual environment and virtual reality environment, the point selection was intuitive and easy to use, with the virtual reality environment providing a very immersive experience for the user. In order to quantify the accuracy of the data gathered, the results from testing in multiple virtual environments, including virtual reality, were compared. Using the data gathered, the accuracy of the localisation results was measured, with the results being compared to ground truth data in order for the error to be quantified. Based on the results acquired, it was found that most of the error came from human error in manually selecting localisation points, rather than error in the depth data itself. It was found to be the case in both 2D and 3D environments, with the virtual reality environment having the highest overall error and human contribution error. One reason for this is due to the perspective of where the

points may have been selected in both environments, as with apples being a spherical shape it was easy to be slightly off-centre when selecting a point. Although this did add error to the results, it can be concluded that this is still within a usable range for applications such as robotic harvesting, based on literature referencing the error range of robotic arms and manipulators being approximately half the diameter of an apple, in this case making the effective error range approximately 36mm[38]. Based on the results from the experiments conducted, the methods used would be a viable option for the application of crop identification. Further advances using this technology could also be made, with the potential for centroid calculating algorithms combined with the manual point selection to be implemented and utilised, essentially giving a corrected centre position of each apple which would increase the accuracy and subsequently decrease the human error component. Another option for further development would be to automate the crop identification and point selection process using a trained neural network, however the step of human interaction and involvement in the development of this technology cannot be overlooked, as it provides a valuable space for the research and growth of this technology.

Although it was expected for some methods of remote sensing, it could be seen from the results that some methods are indeed affected by external factors such as sunlight. This was especially noticeable using the Realsense camera which uses active stereo vision, emitting an array of infrared beams in order to improve depth data accuracy. Although the data was still usable, there was a significant difference between the indoor and outdoor data collected, with more variation in the depth data as well as more noise in the outdoor point clouds as the sunlight interfered with this active stereo vision system. After analysing the data, it could be seen that the standard deviation of the outdoor data increased as the distance increased, however with the indoor data there was only a slight variation in the standard deviation data across all measurements. Although it was outside the scope of this project, for future research, it would be valuable to compare different types of remote sensing techniques to the RGBD camera results, in order to better analyse the effect of the external factors

on the point cloud data. Based on the testing conducted with the Realsense RGBD camera, it was found that this method of data collection would work and operate within the error range for applications such as robotic harvesting, however advancements and refinements would need to be made in order to be used in a commercial application. Compared to more advanced and expensive laser scanning systems that have typically been used for studies in this field, it makes the possibilities for future development much more approachable, due to the accessibility, comparative low cost and portability of these consumer grade 3D vision systems. With the results being within the acceptable range for the application of harvesting, this technology can be utilised and adapted further for full robotic harvesting, using virtual harvesting as an intermediate step towards autonomous systems. Based on the data collected, further research on the communications between the data collection and data interaction would be needed in order to test the robustness of the system in a real environment.

Utilising these 3D point clouds in a virtual environment was found to be very intuitive, however more human error was introduced in the VR environment compared to the normal virtual environment. As mentioned in Section 9.2, there would be multiple areas to investigate methods to increase accuracy in further research. Another consideration that needs to be taken into account when conduction future research is the development for processing pipeline between the 3D scanning, to the virtual environment in Unreal Engine. Although it was outside the scope of this project, it is a vital part of the overall implementation of this technology. Based on the literature reviewed, the best approach would be to set up a client side server, where RGB and depth images are received, rather than raw point cloud data. This data can then be processed client side to produce usable point cloud data in a virtual environment, while reducing processing and latency, making it a viable pipeline for live streaming depth data.

With the accuracy and precision of the vision systems having been tested, improvements could easily be made with superior hardware that is constantly being developed. The next major step for this research would be to test the accuracy and correlation

between the coordinates obtained in the virtual environment, and test them with a robotic arm in a controlled environment. This would prove the fundamental concept of virtual harvesting and provide opportunity for further development. Overall this area has great opportunity for further development, as it would be a valuable contribution to the horticultural industry, whilst also providing alternative employment for an extended demographic of potential employees around the world.

References

[1] figure.nz, "Export value of the new zealand horticulture industry by sub-sector," 2022. [Online]. Available: https://figure.nz/chart/VUDY8fb30SauH2oc.

[2] D. Duncan, "The impacts of covid-19 on aotearoa/new zealand's working people: A report 12 months on," *Noticias CIELO*, no. 3, p. 14, 2021.

[3] L. Gualtieri, E. Rauch, and R. Vidoni, "Emerging research fields in safety and ergonomics in industrial collaborative robotics: A systematic literature review," *Robotics and Computer-Integrated Manufacturing*, vol. 67, p. 101 998, 2021, ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2020.101998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S073658452030209X.

[4] C. W. Bac, E. J. van Henten, J. Hemming, and Y. Edan, "Harvesting robots for high-value crops: State-of-the-art review and challenges ahead," *Journal of Field Robotics*, vol. 31, no. 6, pp. 888–911, 2014. DOI: https://doi.org/10.1002/rob.21525. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21525. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21525.

[5] B. Waddle, "Crop growing practices," *Cotton*, vol. 24, pp. 233–263, 1984.

[6] J. Lordan, P. Francescatto, L. I. Dominguez, and T. L. Robinson, "Long-term effects of tree density and tree shape on apple orchard performance, a 20 year study—part 1, agronomic analysis," *Scientia Horticulturae*, vol. 238, pp. 303–317, 2018, ISSN: 0304-4238. DOI: https://doi.org/10.1016/j.scienta.2018.04.

033. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304423818302826.

[7] J. Wilson, "Understanding the pomology of the planar cordon tree architecture in apple: A thesis presented in partial fulfilment of the requirements for the degree of master of science in horticulture at massey university, palmerston north, new zealand," Ph.D. dissertation, Massey University, 2020.

[8] J. Gené-Mola, E. Gregorio, J. Guevara, *et al.*, "Fruit detection in an apple orchard using a mobile terrestrial laser scanner," *Biosystems engineering*, vol. 187, pp. 171–184, 2019.

[9] R. Collis, "Lidar," *Applied optics*, vol. 9, no. 8, pp. 1782–1788, 1970.

[10] U. Wandinger, "Introduction to lidar," in *Lidar: range-resolved optical remote sensing of the atmosphere*, Springer, 2005, pp. 1–18.

[11] P. Daukantas, "Lidar in space: From apollo to the 21 st century," *Optics & Photonics News*, vol. 20, no. 6, pp. 30–35, 2009.

[12] T. Raj, F. Hanim Hashim, A. Baseri Huddin, M. F. Ibrahim, and A. Hussain, "A survey on lidar scanning mechanisms," *Electronics*, vol. 9, no. 5, p. 741, 2020.

[13] L. Li *et al.*, "Time-of-flight camera—an introduction," *Technical white paper*, no. SLOA190B, 2014.

[14] M. Hansard, S. Lee, O. Choi, and R. P. Horaud, *Time-of-flight cameras: principles, methods and applications*. Springer Science & Business Media, 2012.

[15] Y. Cui, S. Schuon, D. Chan, S. Thrun, and C. Theobalt, "3d shape scanning with a time-of-flight camera," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 2010, pp. 1173–1180.

[16] E. R. Eiríksson, J. Wilm, D. B. Pedersen, and H. Aanæs, "Precision and accuracy parameters in structured light 3-d scanning," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, pp. 7–15, 2016.

[17] T. Bell, B. Li, and S. Zhang, "Structured light techniques and applications," *Wiley Encyclopedia of Electrical and Electronics Engineering*, pp. 1–24, 1999.

[18] R. M. Ruiz, M. T. M. Torres, and P. S. Allegue, "Comparative analysis between the main 3d scanning techniques: Photogrammetry, terrestrial laser scanner, and structured light scanner in religious imagery: The case of the holy christ of the blood," *ACM Journal on Computing and Cultural Heritage (JOCCH)*, vol. 15, no. 1, pp. 1–23, 2021.

[19] G. F. Poggio and T. Poggio, "The analysis of stereopsis," *Annual review of neuroscience*, vol. 7, no. 1, pp. 379–412, 1984.

[20] W. W. Sprague, E. A. Cooper, I. Tošić, and M. S. Banks, "Stereopsis is adaptive for the natural environment," *Science advances*, vol. 1, no. 4, e1400254, 2015.

[21] M. Kytö, M. Nuutinen, and P. Oittinen, "Method for measuring stereo camera depth accuracy based on stereoscopic vision," in *Three-dimensional imaging, interaction, and measurement*, SPIE, vol. 7864, 2011, pp. 168–176.

[22] D. J. Montgomery, C. K. Jones, J. N. Stewart, and A. Smith, "Stereoscopic camera design," in *Stereoscopic Displays and Virtual Reality Systems IX*, SPIE, vol. 4660, 2002, pp. 26–37.

[23] A. Hogue and M. Jenkin, "Active stereo vision," in *Computer Vision: A Reference Guide*, Springer, 2021, pp. 27–32.

[24] Y. Wei, Z. Dong, and C. Wu, "Depth measurement using single camera with fixed camera parameters," *IET Computer Vision*, vol. 6, no. 1, pp. 29–39, 2012.

[25] P. Cignoni, G. Ranzuglia, M. Callieri, *et al.*, "Meshlab," 2011.

[26] M. Callieri, G. Ranzuglia, M. Dellepiane, P. Cignoni, and R. Scopigno, "Meshlab as a complete open tool for the integration of photos and colour with high-resolution 3d geometry data," *Comput Appl Quant Methods Archaeol*, pp. 406–16, 2012.

[27] D. Girardeau-Montaut, "Cloudcompare," *France: EDF R&D Telecom ParisTech*, vol. 11, p. 5, 2016.

[28] T. J. Dewez, D. Girardeau-Montaut, C. Allanic, and J. Rohmer, "Facets: A cloudcompare plugin to extract geological planes from unstructured 3d point clouds," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 41, pp. 799–804, 2016.

[29] J. Gené-Mola, R. Sanz-Cortiella, J. R. Rosell-Polo, A. Escola, and E. Gregorio, "In-field apple size estimation using photogrammetry-derived 3d point clouds: Comparison of 4 different methods considering fruit occlusions," *Computers and electronics in agriculture*, vol. 188, p. 106 343, 2021.

[30] A. Escola, J. M. Peña, F. López-Granados, *et al.*, "Mobile terrestrial laser scanner vs. uav photogrammetry to estimate woody crop canopy parameters–part 1: Methodology and comparison in vineyards," *Computers and Electronics in Agriculture*, vol. 212, p. 108 109, 2023.

[31] D. Girardeau-Montaut. [Online]. Available: `https://www.danielgm.net/cc/`.

[32] D. Moon, S. Chung, S. Kwon, J. Seo, and J. Shin, "Comparison and utilization of point cloud generated from photogrammetry and laser scanning: 3d world model for smart heavy equipment planning," *Automation in Construction*, vol. 98, pp. 322–331, 2019.

[33] D. Mader, R. Blaskow, P. Westfeld, and H.-G. Maas, "Uav-based acquisition of 3d point cloud–a comparison of a low-cost laser scanner and sfm-tools," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, pp. 335–341, 2015.

[34] R. Brunelli, *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

[35] G. Lin, Y. Tang, X. Zou, J. Li, and J. Xiong, "In-field citrus detection and localisation based on rgb-d image analysis," *Biosystems Engineering*, vol. 186, pp. 34–44, 2019.

[36] O. Wasenmüller and D. Stricker, "Comparison of kinect v1 and v2 depth images in terms of accuracy and precision," in *Computer Vision–ACCV 2016 Workshops: ACCV 2016 International Workshops, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part II 13*, Springer, 2017, pp. 34–45.

[37] Y. Onishi, T. Yoshida, H. Kurita, T. Fukao, H. Arihara, and A. Iwai, "An automated fruit harvesting robot by using deep learning," *Robomech Journal*, vol. 6, no. 1, pp. 1–8, 2019.

[38] V. Babin and C. Gosselin, "Mechanisms for robotic grasping and manipulation," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 573–593, 2021.

[39] J. McGlade, L. Wallace, B. Hally, A. White, K. Reinke, and S. Jones, "An early exploration of the use of the microsoft azure kinect for estimation of urban tree diameter at breast height," *Remote Sensing Letters*, vol. 11, no. 11, pp. 963–972, 2020.

[40] M. Tölgyessy, M. Dekan, L. Chovanec, and P. Hubinsk, "Evaluation of the azure kinect and its comparison to kinect v1 and kinect v2," *Sensors*, vol. 21, no. 2, p. 413, 2021.

[41] X. Liang, A. Kukko, I. Balenović, *et al.*, "Close-range remote sensing of forests: The state of the art, challenges, and opportunities for systems and data acquisitions," *IEEE Geoscience and Remote Sensing Magazine*, vol. 10, no. 3, pp. 32–71, 2022.

[42] N. Chernov and C. Lesort, "Least squares fitting of circles," *Journal of Mathematical Imaging and Vision*, vol. 23, pp. 239–252, 2005.

[43] K. Kusumam, T. Krajnık, S. Pearson, T. Duckett, and G. Cielniak, "3d-vision based detection, localization, and sizing of broccoli heads in the field," *Journal of Field Robotics*, vol. 34, no. 8, pp. 1505–1518, 2017.

[44] H. Yang, X. Wang, and G. Sun, "Three-dimensional morphological measurement method for a fruit tree canopy based on kinect sensor self-calibration," *Agronomy*, vol. 9, no. 11, p. 741, 2019.

[45] J. K. Haas, "A history of the unity game engine," *Diss. Worcester Polytechnic Institute*, vol. 483, no. 2014, p. 484, 2014.

[46] M. Pasternak, N. Kahani, M. Bagherzadeh, J. Dingel, and J. R. Cordy, "Simgen: A tool for generating simulations and visualizations of embedded systems on the unity game engine," in *Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 2018, pp. 42–46.

[47] R. Andaru, B. Cahyono, G. Riyadi, G. Ramadhan, S. Tuntas, *et al.*, "The combination of terrestrial lidar and uav photogrammetry for interactive architectural heritage visualization using unity 3d game engine," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 39–44, 2019.

[48] Y. Wu, H. Vo, J. Gong, and Z. Zhu, "Unitypic: Unity point-cloud interactive core," *Parallel graphics and visualisation*, 2021.

[49] A. Sanders, *An introduction to Unreal engine 4*. CRC Press, 2016.

[50] S. González-Domınguez, J. Balado, A. Novo, and P. Arias, "Tree digitisation from point clouds with unreal engine," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 10, pp. 555–560, 2023.

[51] N. L. Faust and W. E. Roper, "Geo-spatial and temporal image and data exploitation iii," *Geo-Spatial and Temporal Image and Data Exploitation III*, vol. 5097, 2003.

[52] R. Fischer, A. Mühlenbrock, F. Kulapichitr, V. N. Uslar, D. Weyhe, and G. Zachmann, "Evaluation of point cloud streaming and rendering for vr-based telepresence in the or," in *International Conference on Virtual Reality and Mixed Reality*, Springer, 2022, pp. 89–110.

[53] D. M. Hilty, K. Randhawa, M. M. Maheu, *et al.*, "A review of telepresence, virtual reality, and augmented reality applied to clinical care," *Journal of Technology in Behavioral Science*, vol. 5, pp. 178–205, 2020.

[54] J. Sanders and E. Kandrot, *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.

[55] C. Schröder, M. Sharma, J. Teuber, R. Weller, and G. Zachmann, "Dyncam: A reactive multithreaded pipeline library for 3d telepresence in vr," in *Proceedings of the Virtual Reality International Conference-Laval Virtual*, 2018, pp. 1–8.

[56] Z. Fialová, R. Klepárnık, and J. Sedláček, "Visualization of woody vegetation changes in 3d point clouds," *Journal of Digital Landscape Architecture*, pp. 301–309, 2022.

[57] N. Ariesen-Verschuur, C. Verdouw, and B. Tekinerdogan, "Digital twins in greenhouse horticulture: A review," *Computers and Electronics in Agriculture*, vol. 199, p. 107 183, 2022.

[58] D. W. Carruth, C. Hudson, A. A. Fox, and S. Deb, "User interface for an immersive virtual reality greenhouse for training precision agriculture," in *International Conference on Human-Computer Interaction*, Springer, 2020, pp. 35–46.

[59] Y. Zang, Z. Zhu, Z. Song, and E. Mao, "Virtual reality and the application in virtual experiment for agricultural equipment," in *Computer and Computing Technologies in Agriculture IV: 4th IFIP TC 12 Conference, CCTA 2010, Nanchang, China, October 22-25, 2010, Selected Papers, Part III 4*, Springer, 2011, pp. 257–268.

[60] Q. Wang and Q. Zhang, "Three-dimensional reconstruction of a dormant tree using rgb-d cameras," in *2013 Kansas City, Missouri, July 21-July 24, 2013*, American Society of Agricultural and Biological Engineers, 2013, p. 1.

[61] S. Rijal, S. Pokhrel, M. Om, and V. P. Ojha, "Comparing depth estimation of azure kinect and realsense d435i cameras," *Available at SSRN 4597442*, 2023.

[62] J. Gené-Mola, J. Llorens, J. R. Rosell-Polo, *et al.*, "Assessing the performance of rgb-d sensors for 3d fruit crop canopy characterization under different operating and lighting conditions," *Sensors*, vol. 20, no. 24, p. 7072, 2020.

[63] F. Mémoli and G. Sapiro, "Comparing point clouds," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 2004, pp. 32–40.

[64] A. Šmıd, "Comparison of unity and unreal engine," *Czech Technical University in Prague*, pp. 41–61, 2017.

[65] *Unreal engine — unreal engine 5.0 documentation.* [Online]. Available: `https://docs.unrealengine.com/5.0/en-US/lidar-point-cloud-plugin-for-unreal-engine/`.

[66] D. Martin-Fuentes and P. M. Cabezos-Bernal, "Universal point cloud viewer based on unreal engine," *DISEGNARECON*, vol. 16, no. 30, pp. 13–1, 2023.

[67] S. Soudarissanane, R. Lindenbergh, M. Menenti, and P. Teunissen, "Scanning geometry: Influencing factor on the quality of terrestrial laser scanning points," *ISPRS journal of photogrammetry and remote sensing*, vol. 66, no. 4, pp. 389–399, 2011.

[68] M. Trosin, I. Dekemati, and I. Szabó, "Measuring soil surface roughness with the realsense d435i," *Acta Polytechnica Hungarica*, vol. 18, no. 6, pp. 141–155, 2021.

[69] Y. Chen, J. Zhang, W. Li, Y. Ren, and Y. Tan, "Grading method of apple by maximum cross-sectional diameter based on computer vision," *Transactions of the Chinese Society of Agricultural Engineering*, vol. 28, no. 2, pp. 284–288, 2012.

[70] M. Carmeli, "Rotational relativity theory," *International Journal of Theoretical Physics*, vol. 25, pp. 89–94, 1986.