MULTI-VIEW DEPTH RECONSTRUCTION

Wenrui Zhang u5647399

Supervised By

DR. Chuong Nguyen and Mr. Matt Adcock

November 14, 2016

College of Engineering and Computer Science

Australian National University

This thesis does not contain any content that has been submitted for the award of any other degree or diploma in any university. To the best of the author's knowledge, it contains no material previously published or written by another person, except for where explicit reference is made in the text.

Thanks for my supervisor Chuong. I would never make it without his help.

ABSTRACT

This project developed an algorithm to use a 2*2 camera array to do Multi-View Depth Reconstruction to generate 3D point clouds of the images captured by the camera array. The performance of this Multi-View Depth Reconstruction algorithm was compared with the performance of existing Stereo Reconstruction method in the same experiment environment. The comparing results show that the accuracy of using camera array was improved. However, this Multi-View Depth Reconstruction algorithm was not robust when the object contained a large area of pixels with the same color. Some details in the original image were lost because only a small part of pixels from the original image was used for Multi-View Depth Reconstruction. The running time of this algorithm was too long for an image so it could not be used for the real-time applications. Unifying the color of the images captured by the four cameras, applying weights on the correlation results from different camera pairs, and trying camera array with a larger distance between cameras can all be the further work of this project.

CONTENTS

LIST OF FIGURES

LIST OF EQUATIONS

INTRODUCTION

Using two cameras to capture the same scene, we can get the depth information of the scene. It is actually a very mature technic named Stereo Reconstruction. We may easily start to think about if there is any way to use more than 2 cameras to improve the accuracy of depth reconstruction. This is the topic of this project.

The camera module accessory for the Raspberry Pi allows users to take pictures in full HD (Raspberry Pi Foundation, 2016). In this project, I used 4 Raspberry Pi cameras to make up a 2*2 camera array. I used two of them to do Stereo Reconstruction and then used all the 4 cameras to do a Multi-View Depth Reconstruction. The results of depth reconstruction are delivered as 3D point clouds. By comparing the 3D point clouds generated by Stereo Reconstruction and Multi-View Depth Reconstruction, we can see if the accuracy is improved when using more cameras. We can also see if there are any new problems occur when using camera array instead of stereo cameras.

The Stereo Reconstruction and Multi-View Depth Reconstruction were realized under the same experience environment. The type of the four Raspberry Pi cameras was the same. They all captured images with 480*720 resolution.

The programs in this project were coded by Python 2.7. OpenCV library was used for process images and cameras. The full name of OenCV is Open Source Computer Vision. It was designed for computational efficiency in computer vision and it is free for both academic and commercial use (Itseez, 2016). Spyder software was used for creating a convenient coding interface. Spyder is a powerful interactive development environment for Python with interactive testing, advanced editing, debugging and introspection features (The Spyder Project Contributors, 2016).

At the beginning of this project, the final goal was achieving light-field reconstruction in real time. Steps like Camera Calibration, Stereo Calibration, Stereo Rectification were all preparation work for light-field reconstruction. However, I found that the theories in Stereo Reconstruction could not be easily used for the 2*2 camera array. I needed to develop a new algorithm for Multi-View Depth Reconstruction using the four cameras. Developing this algorithm was a time-consuming work. As a result, the project was not able to enter into the light-field reconstruction stage because

of the time limitation. Comparing the performance of this new algorithm with the performance of the existing Stereo Reconstruction became the new direction of this project.

The codes created in this project can be found at https://github.com/paopaosusu/Stereo-and-Multi-View-Depth-Reconstruction

LITERATURE REVIEW

With the introduction of pinhole cameras in the late 20th century, cameras became a common occurrence in our everyday life. However, the price of its cheapness was significant distortion (opencv dev team, 2016). Camera Calibration is a process of estimating the parameters of a pinhole camera model. The parameters include intrinsic parameters and Extrinsic parameters. The intrinsic parameters encompass focal length, images sensor format, and principal point (Hartley & Zisserman, 2003). OpenCV team provides the algorithm for Camera Calibration based on Zhang model which is a Camera Calibration method that uses traditional calibration techniques and self-calibration techniques (Zhang, 2000). The algorithms of Stereo Calibration and Stereo Rectification are also provided by OpenCV team. Hirschmuller's algorithm is used by OpenCV team for matching. This algorithm uses a pixelwise, mutual information based matching cost for compensation radiometric differences of input images (Hirschmuller, 2008). The Epipolar Geometry was the basic theory for Stereo Calibration and Stereo Rectification. Epipolar Geometry helps to use two cameras to find the depth information. Our eyes work in a similar way where we use two cameras and this is call stereo vision (Mordvintsev & K, 2013). The Multi-View Depth Reconstruction method in this project uses the concept of dichotomy method. Dichotomy method is also called Bisection method which is a rood-finding method that bisects and interval repeatedly and selects subintervals in which a root must lie. It is simple and robust buy relatively slow (Burden & Faires, 1985).

METHODS

To compare the result of Stereo Reconstruction and Multi-View Depth Reconstruction, I did the two types of reconstruction in the same environment. Both two reconstructions included the steps of Camera Calibration and Stereo Calibration.

CAMERA CALIBRATION

CONCEPT INTRODUCTION

The images captured by a pinhole camera are introduced with distortion. Before doing depth reconstruction, the images must be undistorted to get a good result. The main goal of Camera Calibration is getting the intrinsic parameters and distortion coefficients of a camera which can be used for undistorting the images captured by this camera.

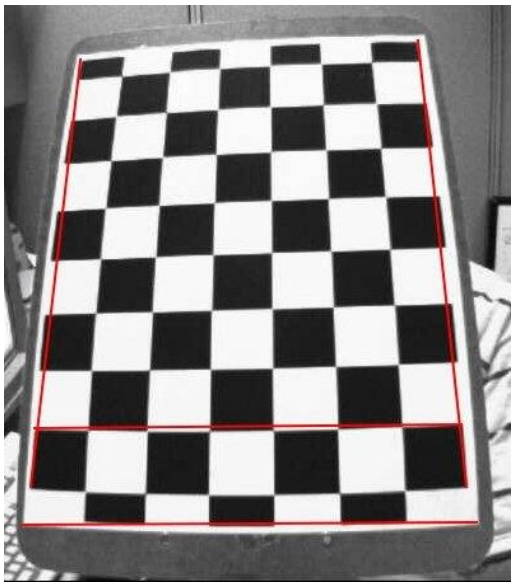


Figure 1 Distorted Image

Intrinsic parameters are always expressed as camera matrix. Camera matrix includes the information of the focal length of the camera and the optical centers (the center pixel's coordination in and captured image) of the camera. Camera matrix is fixed for a certain camera, so once we calculate the camera matrix of a camera, we can store the matrix for further use.

Extrinsic parameters include the rotation and translation vectors. These two vectors translate the camera coordination system to the real-world coordination system. They are not fixed for each image captured by the camera. And they do not contribute too much to this project.

To do Camera Calibration for a camera, we need to provide sample images of a well-defined pattern (e.g., a chessboard). The process of Camera Calibration will find the pattern in each sample image. By comparing the pattern found in the sample images with the well-defined pattern, it returns us the distortion coefficients, camera matrix, and extrinsic parameters. A very simple black box diagram is shown below.

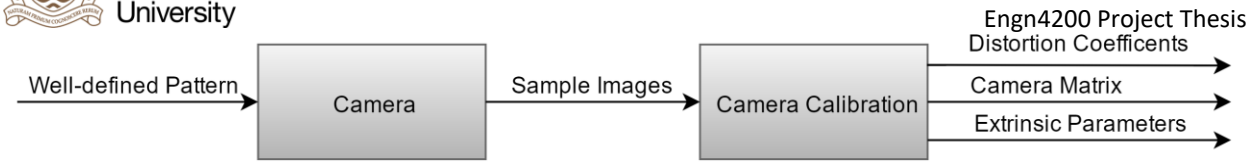


Figure 2 Black Box Diagram of Camera Calibration

IMPLEMENTATION

In this project, I used a 6*9 chessboard (7*10 squares, 6*9 internal corners) as the well-defined pattern. The square size of the chessboard was 37 millimeters. To get different sample images, I moved the camera and captured the images of the chessboard in different angles of view. 38 sample images were used for each camera for Camera Calibration. Some of the sample images are shown below. Since I only needed to find the internal corners of the chessboard in each sample image, I did not need to guarantee the whole chessboard was in each of the sample images, I just needed to make sure all the internal corners were in.

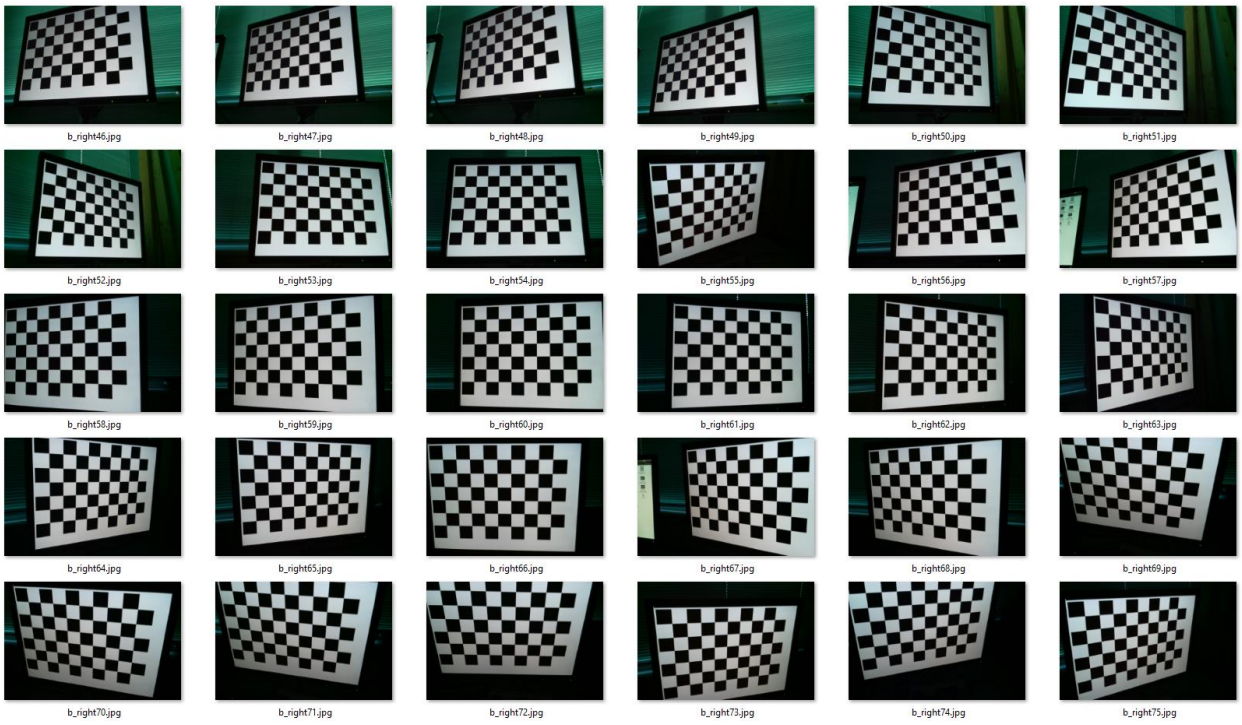


Figure 3 Sample Images for Calibration

For each sample image, we need to find the 6*9 chessboard. Then, we need to store the coordinates of all the internal corners in each sample image. These coordinates are named images points. The image below shows a found chessboard in a sample image.

Figure 4 Well-defined Pattern Found in a Sample

We also need the object points which denote the location of the chessboard internal corners in the world coordination system. Since I knew the size of the squares in the chessboard was 37 millimeters, and the screen used to display the chessboard was fixed, assuming the chessboard was stationary at XY plane, I simply set the object point as (0,0,0), (37,0,0), (74,0,0), …, (0,37,0), (37,37,0), (74,37,0), …

For doing Camera Calibration for all the 4 cameras, I did all the things above separately for each camera.

OPTIMIZATION

At the beginning of this project, I only used 6 sample images (image 01-06) for each camera to do Camera Calibration. I also got the output values I need for further steps. So, I thought everything was fine. However, when the process came to the disparity map generation, I got the result as the image shown below.

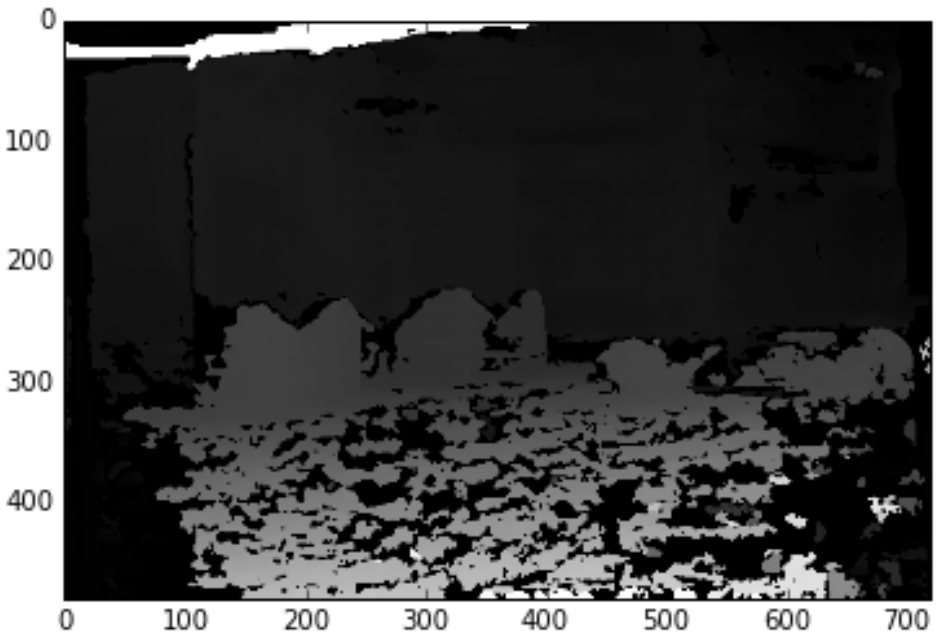


Figure 5 Bad Disparity Map Caused by Bad Camera Calibration

The disparity map here is what I got in my midterm review. It was messy and I could hardly say it was a scene in a room. One cause of the bad result was wrongly using images with red epipolar lines on them. I corrected this mistake but the result was still not desired.

Then I started to consider the other cause: the number of samples images used for Camera Calibration was few. I then recaptured 33 more sample images (image group 07-39) and checked all the groups of images to make sure that they are in the same order for each of the camera. However, things became weird even earlier before disparity map generation. The rectification result of using the whole 39 image groups (image group 01-39) is shown below.

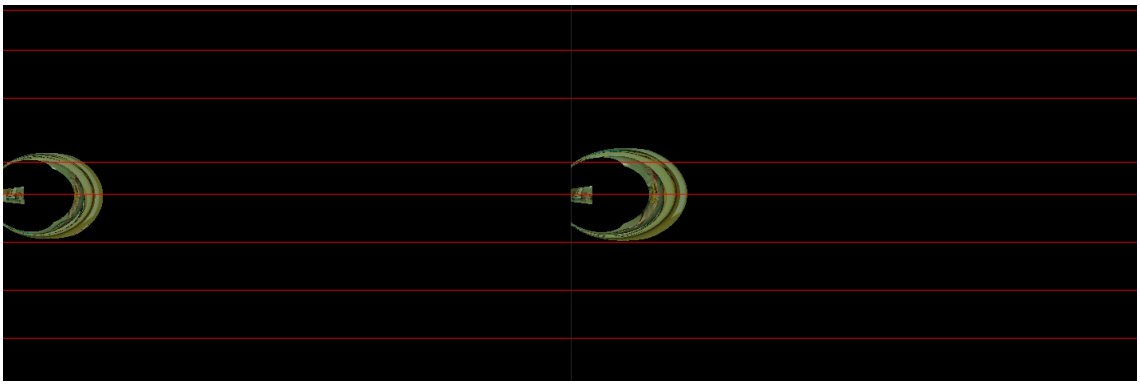


Figure 6 Bad Rectification Result Caused by Bad Calibration

The only reason I could guess was that I introduced unnecessary distortions when holding and moving the camera array manually. I always used my hands to hold the camera array when capturing image group 01-39, a slight shake of the hand could influence a lot to the result of Camera Calibration. So, I captured 38 new image groups. This time, I fixed the platform of the camera array when capturing each image group. By moving and changing the platform, the angle of view of the camera array was changed but I did not need to worry about the hand shaking anymore.

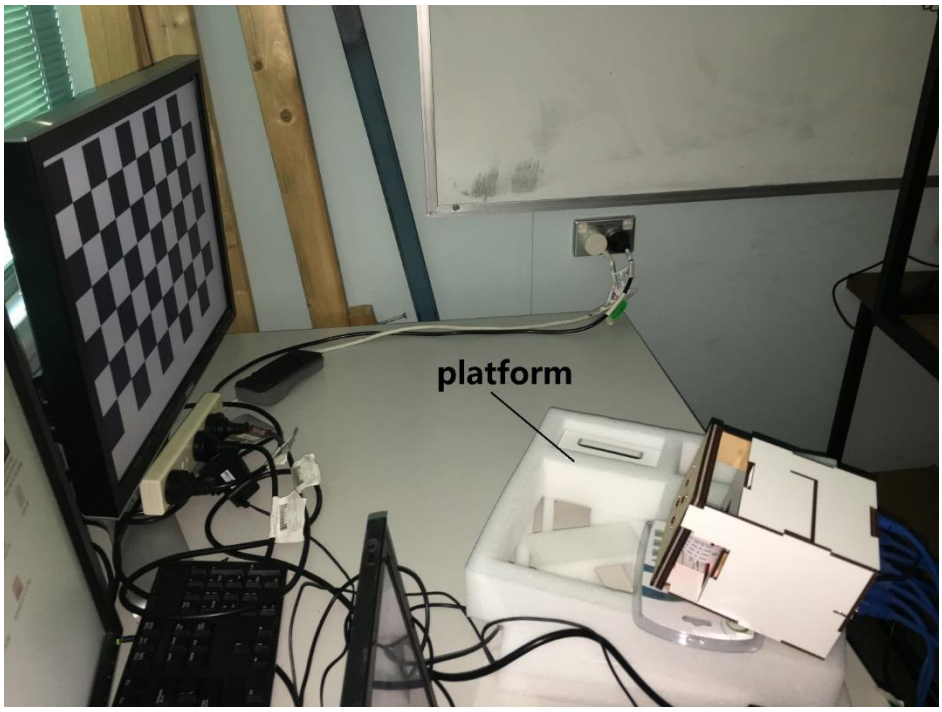


Figure 7 Use Fixed Platform to do Capture images

The result was all good this time.

STEREO CALIBRATION

CONCEPT INTRODUCTION

Camera Calibration returns the parameters of each camera alone, while Stereo Calibration returns the parameters which show the relationship of two cameras. We can use geometric relationships to calculate the depth of the scenes only after getting the spatial relationships of the cameras we use. Stereo Calibration returns Rotation Matrix, Translation Vector, Essential Matrix and

Fundamental Matrix of two cameras. In this project, I used Rotation Matrix and Translation Vector. Rotation Matrix is a 3*3 matrix which indicates the angular difference between the two cameras. Translation Vector is a vector with a length of 3 which shows the distance of the two cameras along the three axes in the 3D world coordinate system.

Since I had already gotten the images of chessboard captured by all the four cameras in the Camera Calibration step, and I also had gotten the object points and image points, the preparation for Stereo Calibration was all down. Using a simple command in python was all I needed for coding.

A simple black box diagram of Stereo Calibration is shown below. The input and output only include the parameters we are interested in this project.

Figure 8 Black Box Diagram of Stereo Calibration

STEREO RECTIFICATION

CONCEPT INTRODUCTION

Stereo Rectification is only needed for Stereo Reconstruction. When capturing the same scenes by two cameras, there is a term called epipolar line on the two cameras. The scenes on the corresponding epipolar lines on the two images captured by the two cameras are the same. If we can slightly warp the two images to make the corresponding epipolar lines aligned, the matching work will be much easier. Stereo Rectification does this work.

Using the Rotation Matrix and Translation Vector between the two cameras for which we want to do Stereo Rectification, and using the Camera Matrix and Distortion Coefficients of the two cameras, the 3*3 Rectification Matrix of each of the two cameras can be calculated. Together with the Distortion Coefficient which has been calculated before, the Rectification Matrix includes all the rest information for warping the images captured by the corresponding camera in order to make

the epipolar lines aligned. A simple black box diagram of Stereo Rectification is shown below, where the New Image1 and New Image2 are undistorted and rectified images.

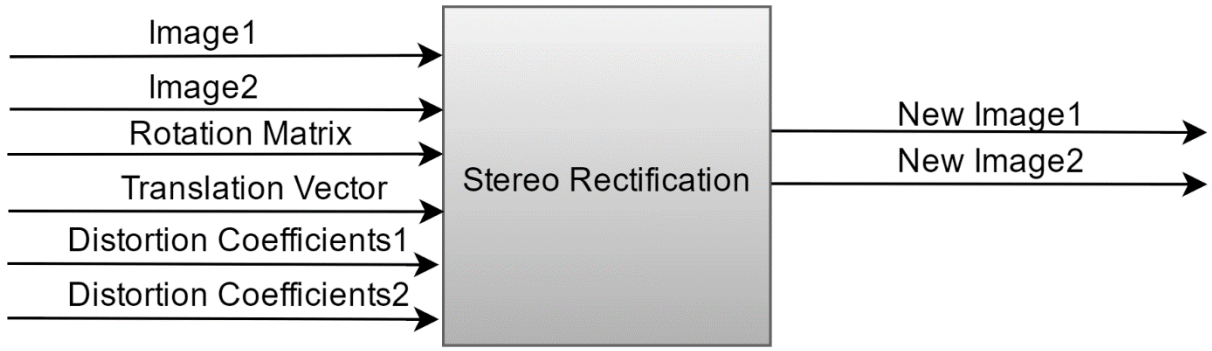


Figure 9 Black Box Diagram of Stereo Rectification

IMPLEMENTATION

The images below show the effects when applying Stereo Rectification to three sample image pairs. As we can see in the images below, the image pairs are square images. After applying Stereo Rectification, the images are no longer squares. They have been warped so that corresponding epipolar lines are aligned. The red lines on the image pairs after Stereo Rectification are some of the epipolar lines. We can see that the scenes on each red line between the two rectified images are the same.

SAMPLE1

Figure 10 Original Images of Sample1 Captured by Stereo Cameras

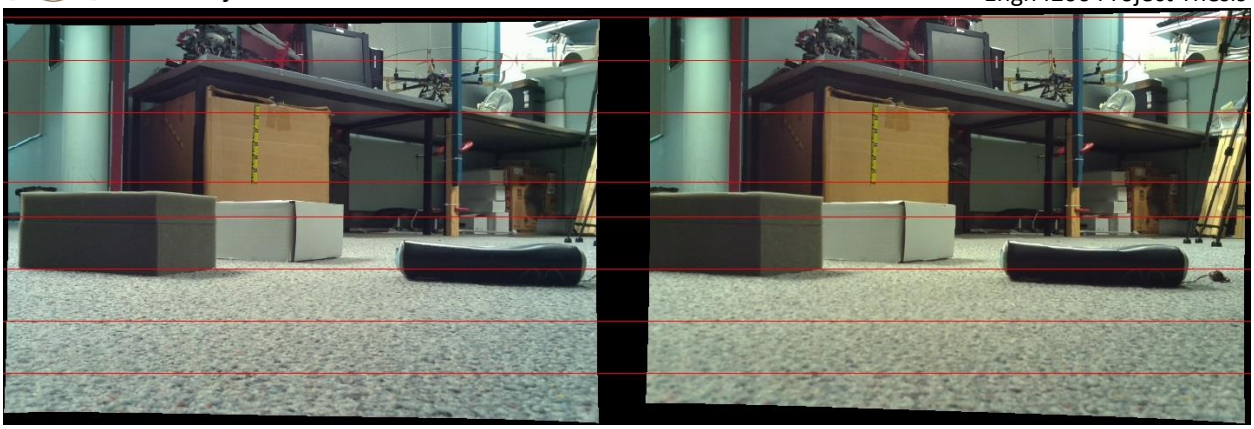


Figure 11 Undistorted, Rectified Sample1 images

SAMPLE2

Figure 12 Original Images of Sample2 Captured by Stereo Cameras

Figure 13 Undistorted, Rectified Sample2 images

SAMPLE3

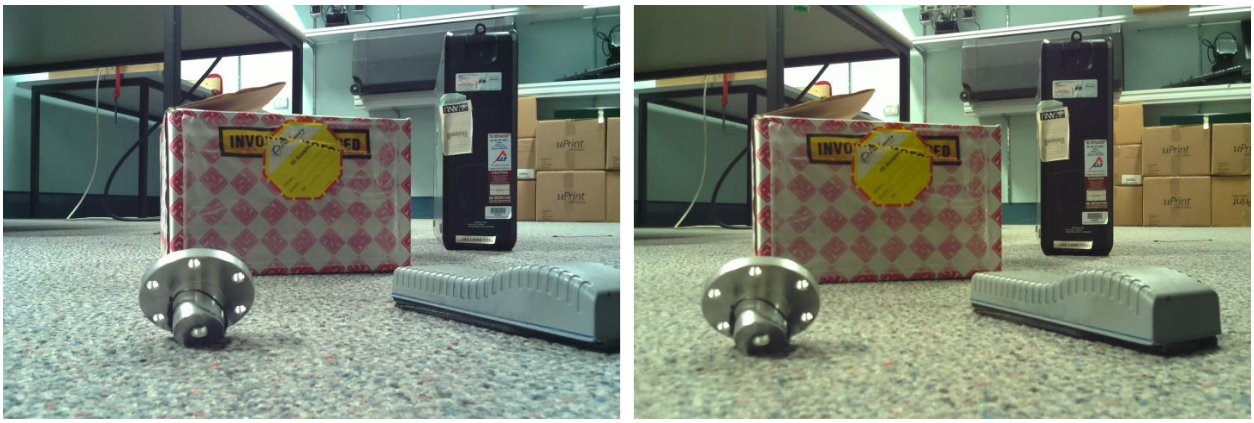


Figure 14 Original Images of Sample3 Captured by Stereo Cameras

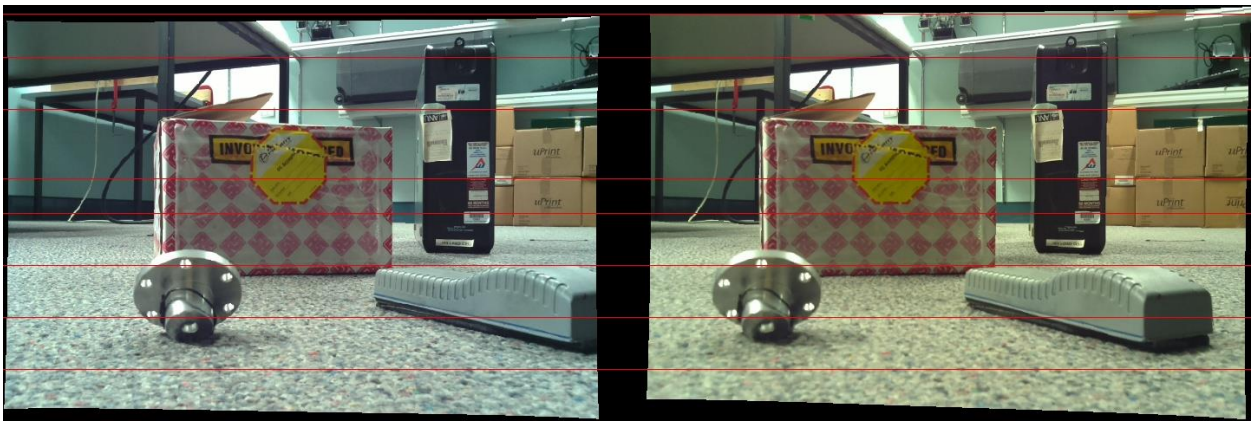


Figure 15 Undistorted, Rectified Sample3 images

CONCEPT INTRODUCTION

Having the epipolar line containing the same scene aligned in the two images, it is easy to match the pixels in the two images and use the geometry relationship described by the image and the equation below to calculate the disparity information for all the pixels.

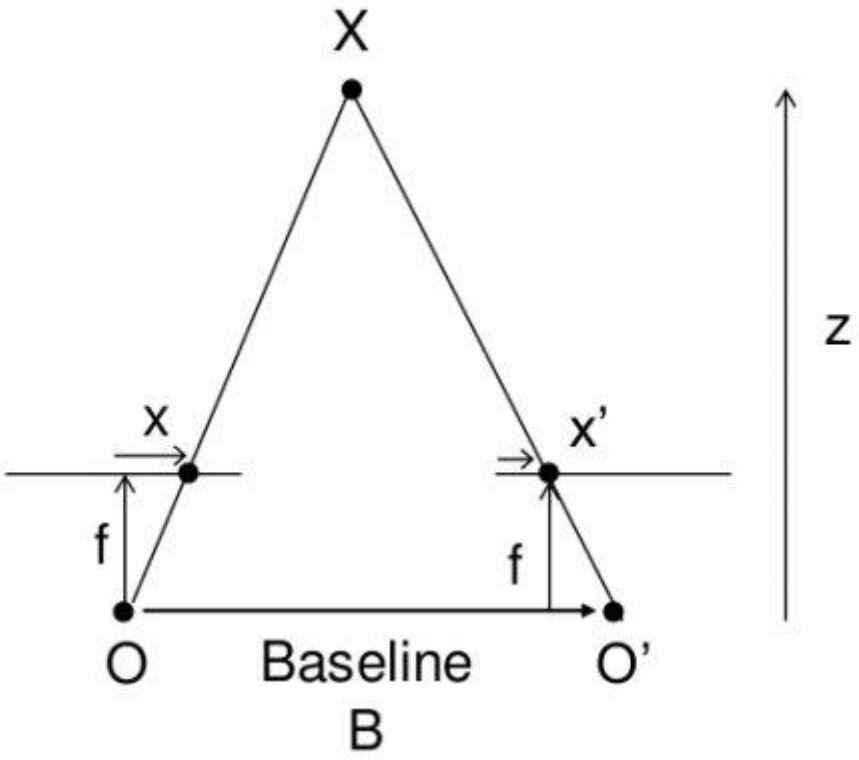


Figure 16 Equivalent Triangles for Depth Reconstruction

$$\text{disparity} = x - x' = \frac{Bf}{Z}$$

Equation 1 Disparity Calculation

The x and x' in the image are two matched pixels in the two images.

After calculating the disparity information for all the pixels in one image (reference image), a disparity map can then be generated.

The two images below are the original rectified image and its disparity map.

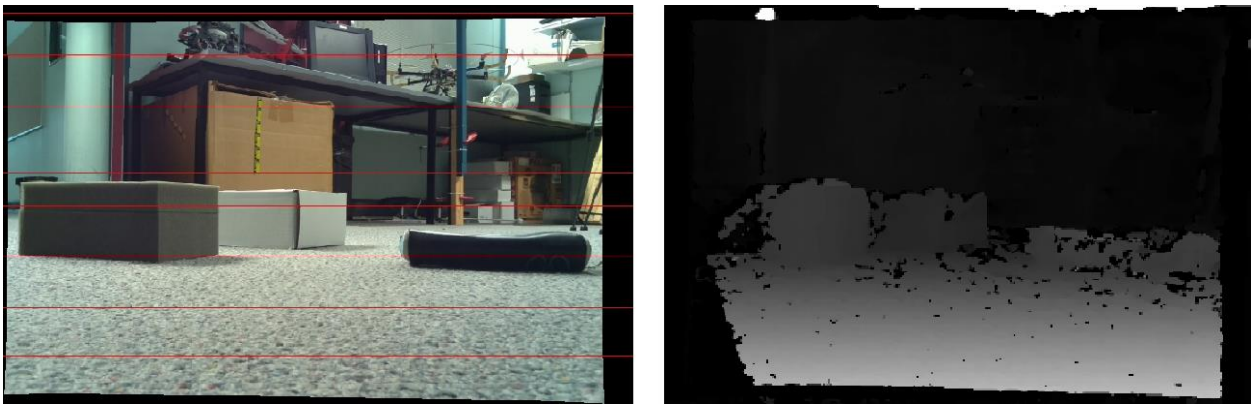


Figure 17 Rectified Image and Disparity Map

In the disparity map above, lighter pixels are nearer to the camera. Darker pixels are farther away from the camera.

Applying the color information from the original rectified image to each pixel in the disparity map, a 3D point cloud can be generated. The image below shows a screenshot of the 3D point cloud generated by Stereo Reconstruction using the original image above.

Figure 18 3D Point Cloud

In Multi-View Depth Reconstruction, the matching method in Stereo Reconstruction does not work. The major reason is illustrated in the diagram below.

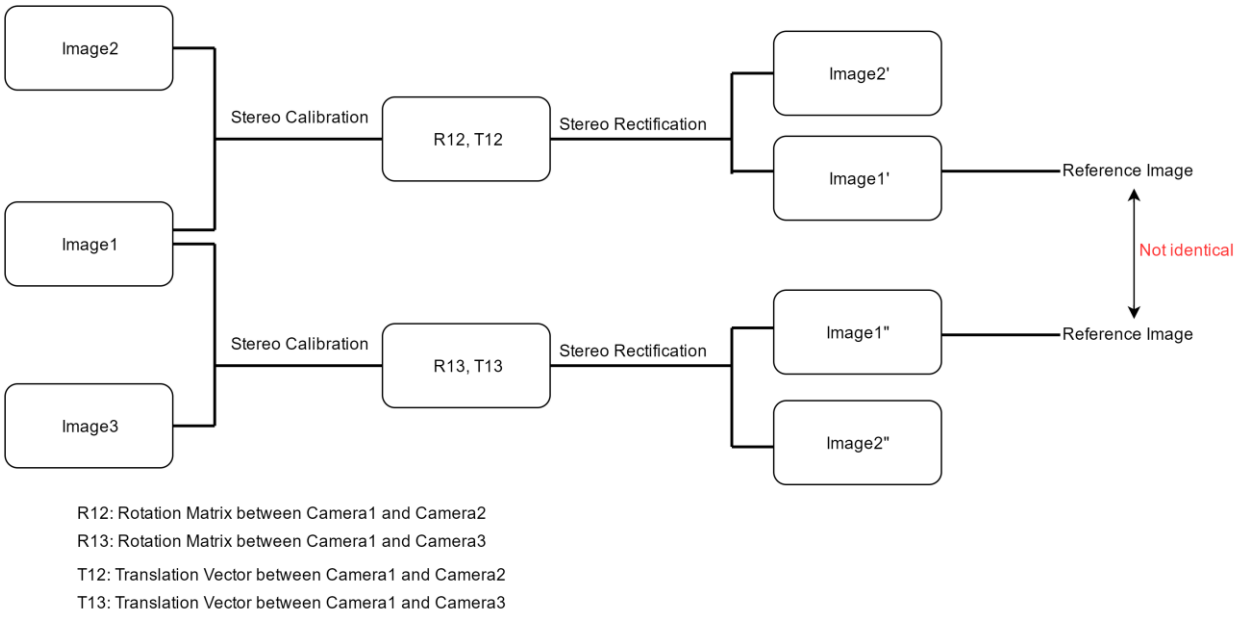


Figure 19 Illustration of Why the Stereo Matching Methos Does not Work for Multi-View

As mentioned before, we must set a reference image in Stereo Reconstruction. We calculate the disparity of each pixel in the reference image so that we can generate a disparity map.

Assuming we are using three cameras, there are two cases in which we want to implement Stereo Calibration: Stereo Calibration for camera1 and camera2, Stereo Calibration for camera1 and camera 3. If we use the matching method in Stereo Reconstruction, we should get the Rotation Matrix and Translation Vector between Camera1 and Camer2 as well as Camera1 and Camer3 (R12, R13, T12, T13 in the diagram above). However, since R12 does not equal to R13, T12 does not equal to T13, the rectified image captured by camera1 are not the same after being warped in Stereo Rectification. That means the reference images are not identical in the two cases. Thus, we cannot generate an only disparity map using all the three cameras, much less generating a disparity map using four cameras in this project.

For Multi-View Depth Reconstruction, I used a different method for matching. This method uses the image captured by one camera as the reference and generates the 3D point cloud using the images captured by all the four cameras. For explaining the method clearly, let's look at the epipolar geometry for two cameras first.

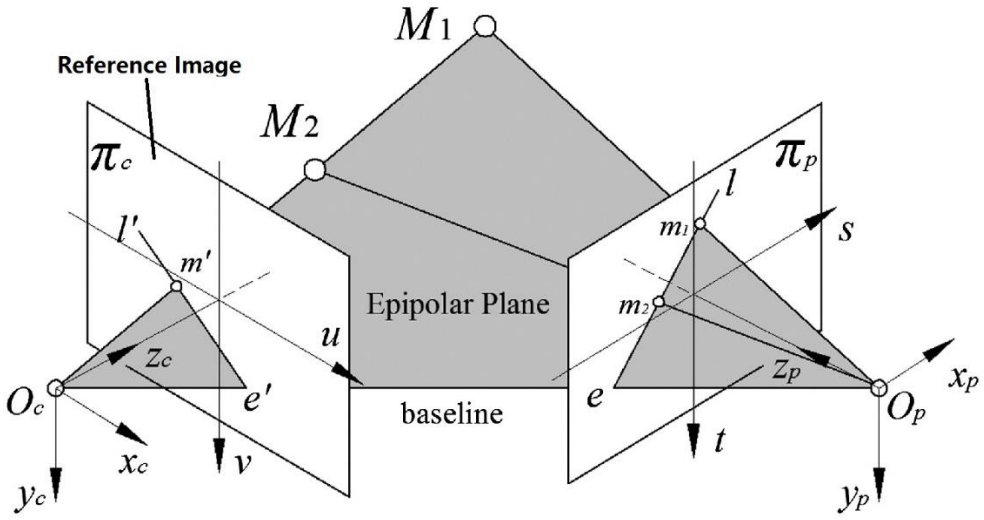


Figure 20 Epipolar Geometry (Zhang, Guo, & Asundi, 2016)

The image above illustrates the epipolar geometry for two cameras. Oc and Op stand for the two cameras. π_c and π_p are the two images captured by the two cameras. l' and l are the corresponding epipolar lines in the two images. Here, we say the image π_c is the reference image. As mentioned before, if we pick a scene on l', the same scene should be found on l. If we pick the scene m' on l' in π_c, we should be able to find a corresponding scene on l in π_p. However, we do not know the depth information of the scene m' (actually the depth information is what we want to work out). That means, we only know the corresponding scene of m' is on the epipolar line l, but we do not know where it is on l. As shown in the image above, both M_1 and M_2 can be the object of scene m' in the real world, but since M_1 and M_2 have different depths, they correspond to different scenes (m_1 and m_2) on l in π_p.

Let's look at this geometry relationship in another way. If we pick a scene on l' in the reference image π_c and the depth information of this scene is fixed, we can know where the object is in the real world. So, the corresponding scene on l can be found.

The two equations below show how to get the world coordinates of the object from the image coordinates of the reference image using the concept of pinhole camera model.

$$X = Z * (u1 - u0)/f$$

Equation 2 Image u1 Coordinate to World X Coordinate

$$Y = Z * (v1 - v0)/f$$

Equation 3 Image v1 Coordinate to World Y Coordinate

In the two equations above, Z is the fixed depth value, (u1, v1) is the image coordinate of the scene we selected in the reference image. (u0, v0) is the center of the image. f is the focal length of the reference image. X and Y are the calculated x, y value of the object in the world coordinate.

The two equations are derived by using theories of similar triangles in pinhole camera model. The pinhole camera model diagram below makes it easier to understand the two equations above.

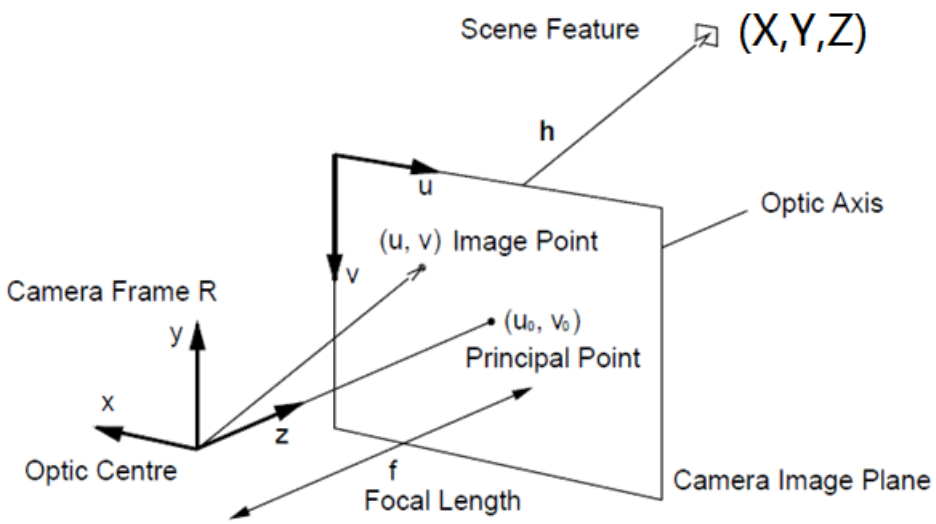


Figure 21 Pinhole Camera Model (Bruneau, Dubray, & Murguet, 2012)

The equation below shows how to derive the corresponding scene on l in image π_c. The first matrix on the right side of the equal sign is the camera matrix of the camera which captured image π_p, the derivation of the camera matrix was finished in Camera Calibration step. The second matrix on the right side of the equal sign is the combinational matrix of the 3*3 Rotation Matrix and the 3*1

Translation Vector between the two cameras. The derivation of these two matrixes was finished in Stereo Calibration step. The third matrix on the right side of the equal sign is the homogeneous world coordinate of the object which we just worked out. The 3*1 matrix on the left side of the equal sign is the homogeneous coordinate of the corresponding scene on l in image π_c.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Equation 4 Word Coordinates to Image Coordinates (opencv dev team, 2016)

For example, when we pick m' on l', and we fix the depth as Z_1 which is the depth of M_1, then we can find the only scene on l in π_p corresponding to m', which is m1. When we pick m' on l', and we fix the depth as Z_2 which is the depth of M_2, then we can find the only scene on l in π_p corresponding to m', which is m_2.

WINDOW CORRELATION

The question now is: which of the two depth Z_1 and Z_2 is nearer to the actual depth of the scene m'? To find the answer to this question, we can chop a pixel window around m', name it as window_m'. Then, we get the pixel window of the same size with window_m' around m_1 and m_2, name them as window_m1 and window_m2.

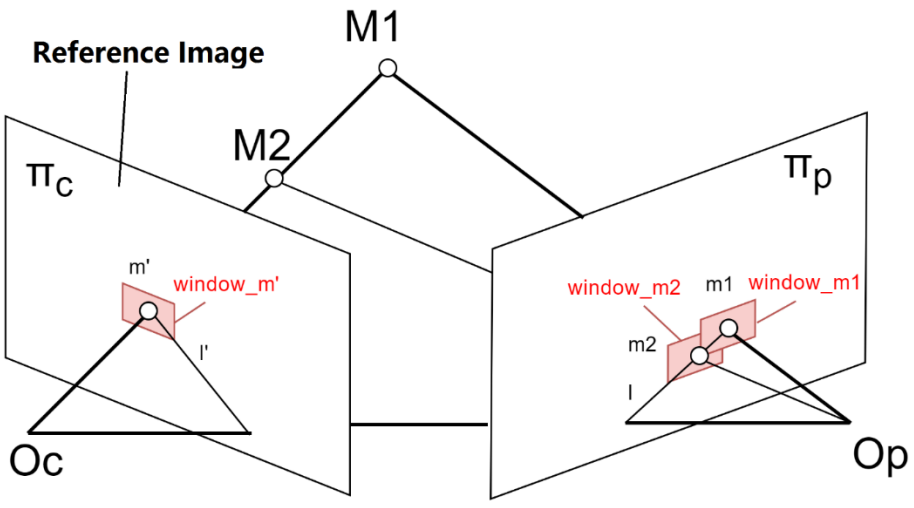


Figure 22 Get the Window from Two Images

By comparing window_m' and window_m1, we can get the correlation c_1 of the two windows. A higher value of c_1 means the scenes in the two windows are more similar to each other. In the same way, we can get the correlation c_2 between window_m' and window_m2. By comparing the value of c_1 and c_2, we can know which window of window_m1 and window_m2 are more similar to window_m'. The window with higher similarity means the depth information used for getting this window is nearer to the actual depth of the scene. For example, if c_1 is greater than c_2, then Z_1 if nearer to the actual depth of the scene than Z_2.

The image diagram below shows the steps of working out which Z is nearer to the actual depth.

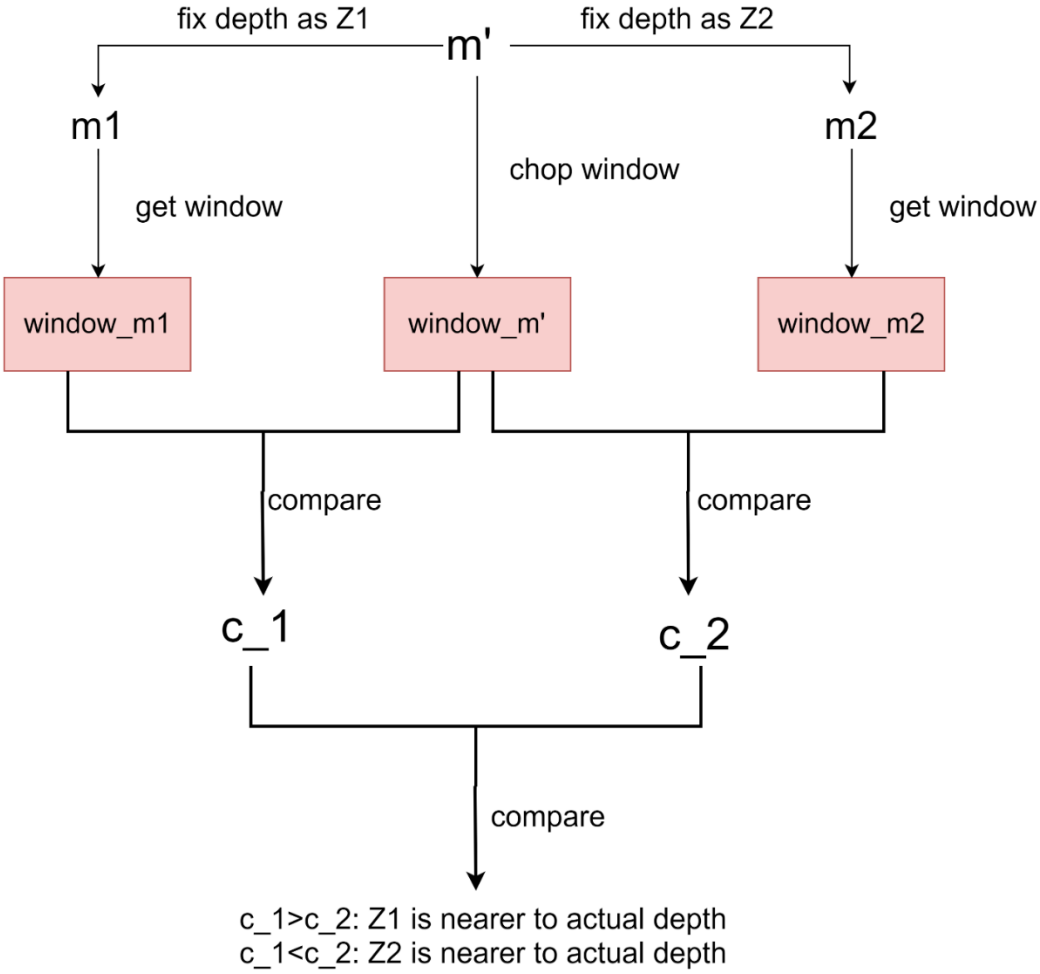


Figure 23 Steps of Working Out Nearer Z

Now, we see that by comparing the correlation, we can find out which Z value is nearer to the actual depth. In order to get the depth information of a scene, we need to exhaustively calculate the correlation for all the potential Z values with an appropriate step. For example, if we know the scene's depth is between 0.5 meters to 3 meters, we can set Z from 0.5 meters to 3 meters with the step of 0.01 meter, and calculate the correlation of each Z. We will get a curve which shows the relationship of Z and correlation as shown below.

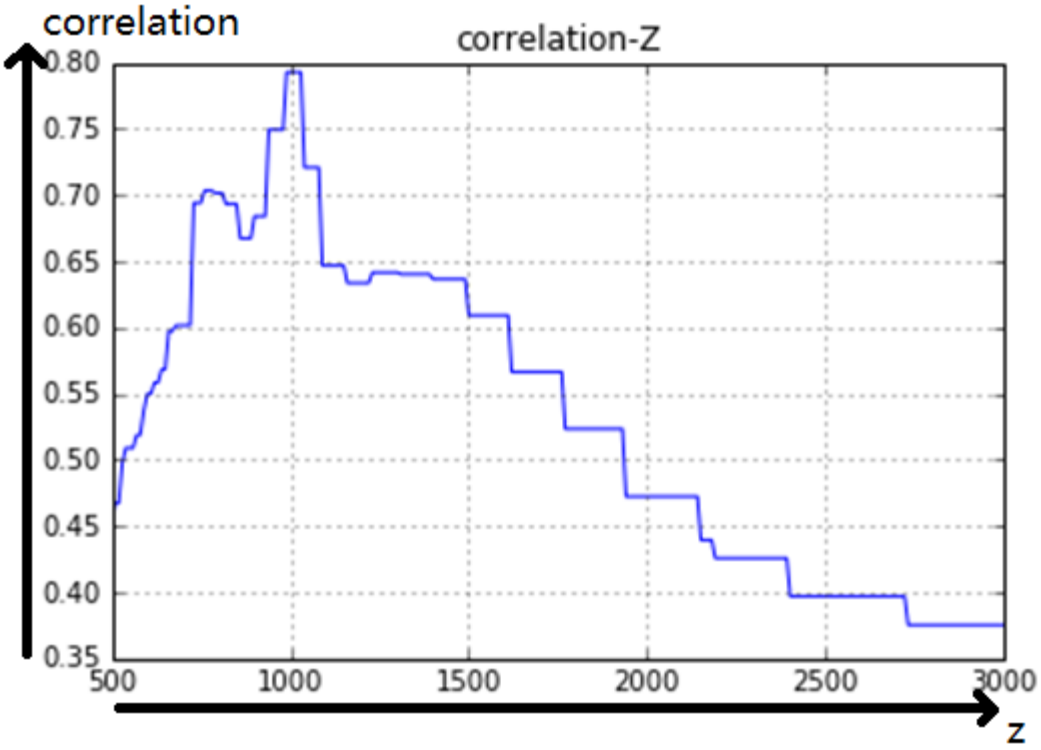


Figure 24 A Correlation-Z Curve from Chopping the Windows

The x-axis is the Z value from 500 millimeters to 3000 millimeters. The y-axis is the correlation. As we see in the curve diagram, the correlation value grows when the Z value grows at beginning, but after reaching a peak value, correlation quickly decreases and never exceed the peak value again. The peak correlation value in the diagram above is reached when Z is around 1000 millimeters. That means the depth of the scene is around 1000 millimeters.

Now, we know how to generate the correlation-Z curve for two images when we pick one pixel (scene) in the reference image, we can easily do the same thing for all the pairs of the four images

captured by the 2*2 camera array we use. However, the reference image must be always the same so that the scene is the same.

The image below shows the 2*2 camera array I used in this project. For a better description of the further steps, let's label the four cameras as camera1, camera2, camera3 and camera4. The four cameras capture the top left, top right, bottom left and bottom right images separately.

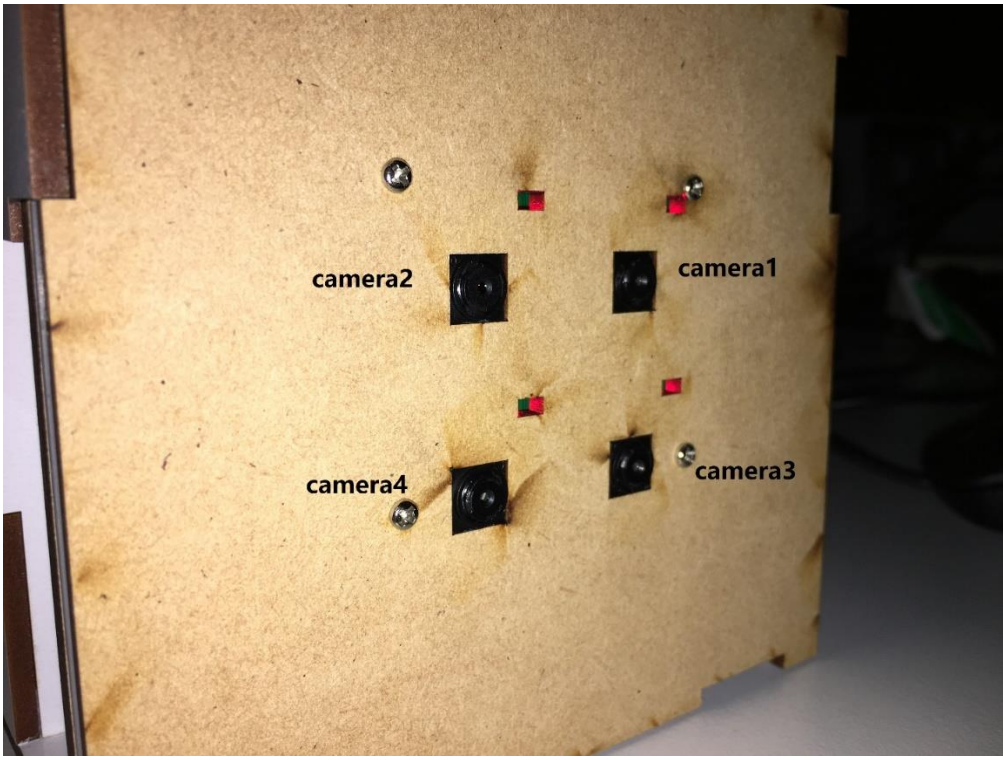


Figure 25 Camera Array Used in This Project

Let's set the image captured by camera1 as the reference image, then we can have the correlation-Z curve for the three camera pairs: (camera1, camera2), (camera1, camera3), (camera1, camera4). To take advantage of using all the four cameras together, we can also have a mean correlation-Z curve on which the correlation values are calculated by taking the average of the three correlation values on the three curves.

The diagram below shows all the correlation-Z curves of the three pairs of cameras as well as the mean correlation-Z curve (the red curve). We can then easily find the peak correlation value on the mean correlation-Z curve and note down the corresponding Z value Z*. Z* is then the depth value of the scene we picked in the reference image.

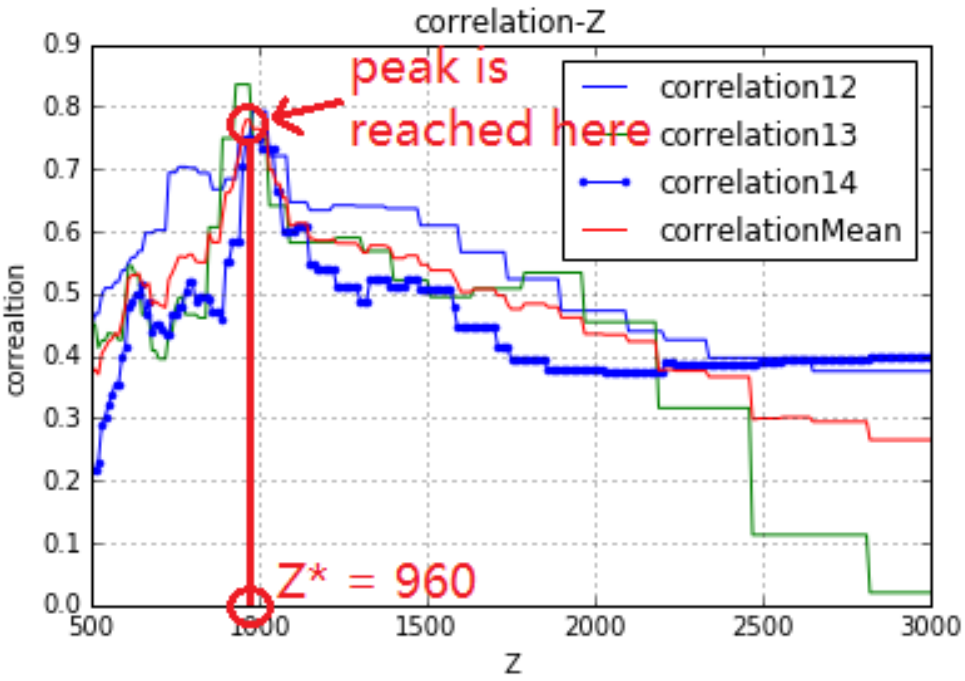


Figure 26 All Correlation-Z Curves from Chopping Windows

Having the Z* which is nearest to the actual depth of the scene, we can use the Equation 4 to get the corresponding pixel (scene) in the images captured by camera2, camera3, and camera4. The four images below are captured by the four cameras. The blue point in the image captured by camera1 is the scene I picked in the reference image. The blue points in the other three images are the corresponding pixels in the images captured by the other three cameras. As we see, the four blue points are at the same position on the white paper box in the four images. This means the method of calculating Z* works well.

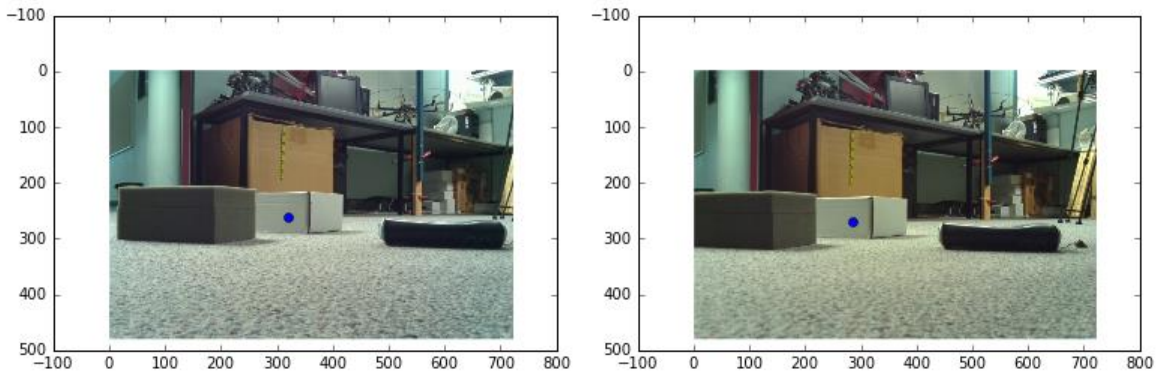


Figure 27 Selected Point in Camera1 Image and Corresponding Point in Camera2 Image

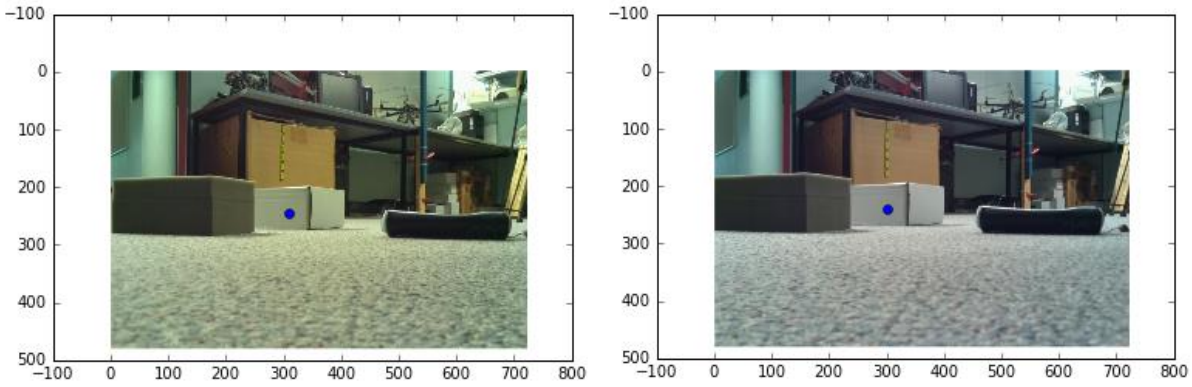


Figure 28 Corresponding Point in Camera3 Image and Camera4 Image

OPTIMIZATION

INTERPOLATION FOR SMOOTH

There is a problem of the correlation-Z curves below: they are not smooth. The curves are made up of a lot of steps. A bunch of Z values may have a same value of correlation. As a result, there are a bunch of Z values having the highest correlation value in each curve. We do not want this happen because we want to have an only Z* for each pixel in the reference image. We would like to make the curve smooth.

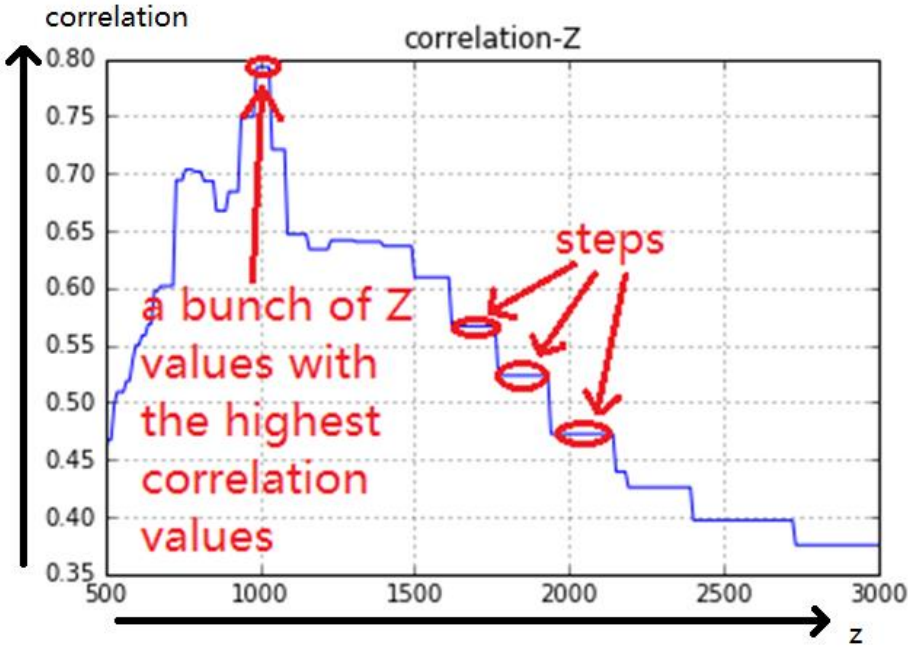


Figure 29 Correlation-Z Curve with Steps

The reason why the curves are not smooth is that the windows in the images captured by camera2, camera3 and camera4 are chopped from the images. When picking the pixel m'($u_{m'}$,$v_{m'}$) in the reference image, and when the Z value is fixed as Z_1, the m_1 can be calculated. However, m_1's image coordinates u_{m1} and v_{m1} are always not integers in most cases. If we want to chop a pixel window around (u_{m1},v_{m1}), we need to firstly round u_{m1} and v_{m1} to the nearest integers. For example, if m' in the reference image is picked as (320, 260), and we calculated m_1 in the images captured be camera2 as (306.45, 255.96), then the m_1 is rounded as (306, 256).

However, if we fix the Z value as Z_2 and found that m_2 coordinates are something like (305.75, 256.06), then m_2 is also rounded to (306,256). That means, for the pixel m' in the reference image, Z_1 and Z_2 have the same corresponding pixel in the other images. The pixel windows around the corresponding pixels of Z_1 and Z_2 are also the same. And of course, Z_1 and Z_2 have the same correlation values in the correlation-Z curve. Having different Z values with the same correlation values, the correlation-Z curve becomes not smooth.

To make the curves smooth, we need to use interpolation and remapping methods to generate new windows instead of directly chop the windows from the original images. There are three interpolation methods could be used in Python for remapping: Linear, Cubic, and Lanczos4. Linear interpolation method uses fewer pixels around for interpolating. Lanczos4 uses the most. The three diagrams below are the three correlation-Z diagrams using different interpolation methods.

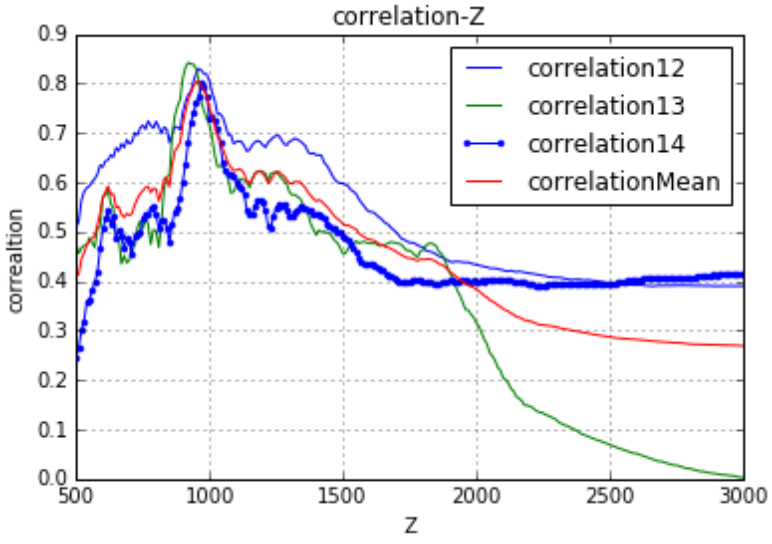


Figure 30 Correlation-Z Curve with Linear Interpolation Window Remapping Method

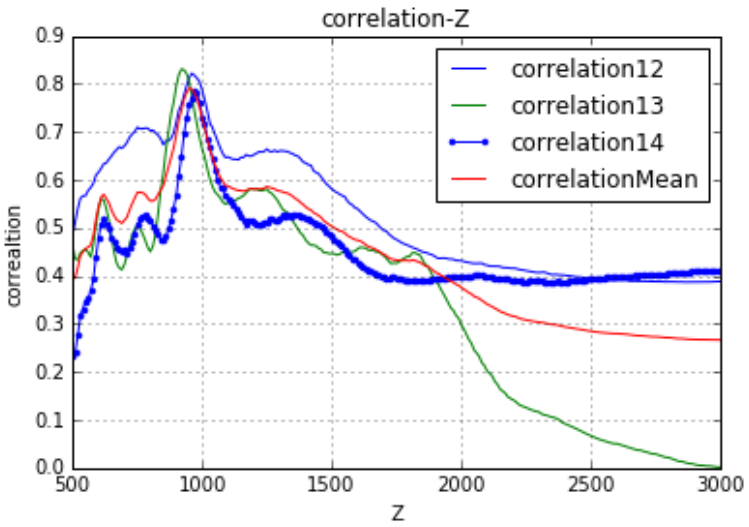


Figure 31 Correlation-Z Curve with Cubic Interpolation Window Remapping Method

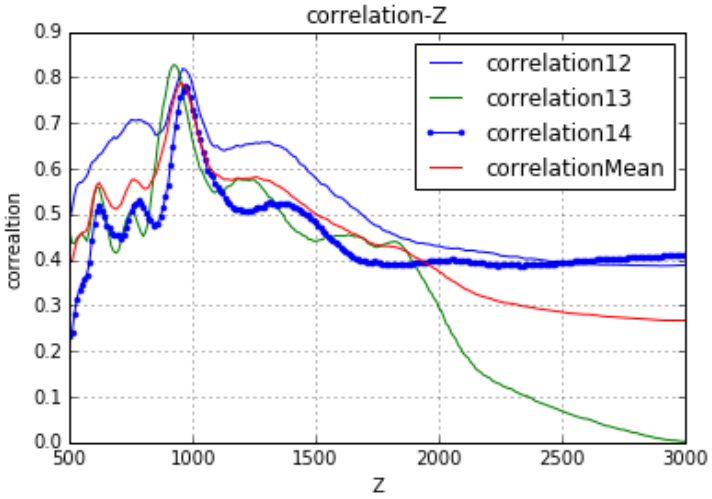


Figure 32 Correlation-Z Curve with Lanczos4 Interpolation Window Remapping Method

As we see, when we use more pixels around for interpolation, the final correlation-Z curve is smoother. But since more pixels are used, the time of interpolation becomes longer. There are trade-offs between the performance of interpolation and the time used for interpolation. Practically, the influence of the interpolation method on the algorithm's running time is much smaller than the influence of some other parameters which I will introduce in this thesis later. Therefore, I used Lanczos4 interpolation method in this project.

DICHOTOMY

Now, we know how to get the depth information for one pixel in the reference image. To generate a depth map, all we need to do is getting the depth information for all the pixels in the reference image. After that, we can have the 3D point cloud of the image by applying color information to the depth map. When I implemented this method using Python, here came a very big issue of this algorithm: The speed was too slow. The algorithm was expected to run for more than 4 days at the beginning.

To make the time of running the algorithm reasonable, I needed to change the value of some parameters to sacrifice the performance for higher speed. The Z step was one of the parameters I could change. When the depth of the scene is from 500 millimeters to 3000 millimeters, if we set the Z step as 50 (which means the resolution of the depth information in the final depth map is 50), then the algorithm will try (3000-500)/50 = 50 different Z values for each pixel. If we set the Z step as 5, the algorithm will try (3000-500)/5 = 500 different Z values for each pixel. We can see that:

$$Z\_step \propto \frac{1}{Resolution}$$

Equation 5 Relationship between Z_step and Resolution of the Depth

$$Z\_step \propto \frac{1}{T}$$

Equation 6 Relationship between Z_step and Running Time of the Program

A greater value of Z step results in less time consumed by running the algorithm but a lower quality of the depth map.

Finding an appropriate Z step to make both the speed and the resolution of the algorithm in the reasonable range is a way to make a trade-off between resolution and speed. But is there a way to ensure a high resolution and a high speed? The answer is yes. To do that, we need to use dichotomy theory.

The images below show the mean correlation-Z curves when picking three different pixels in the reference image.

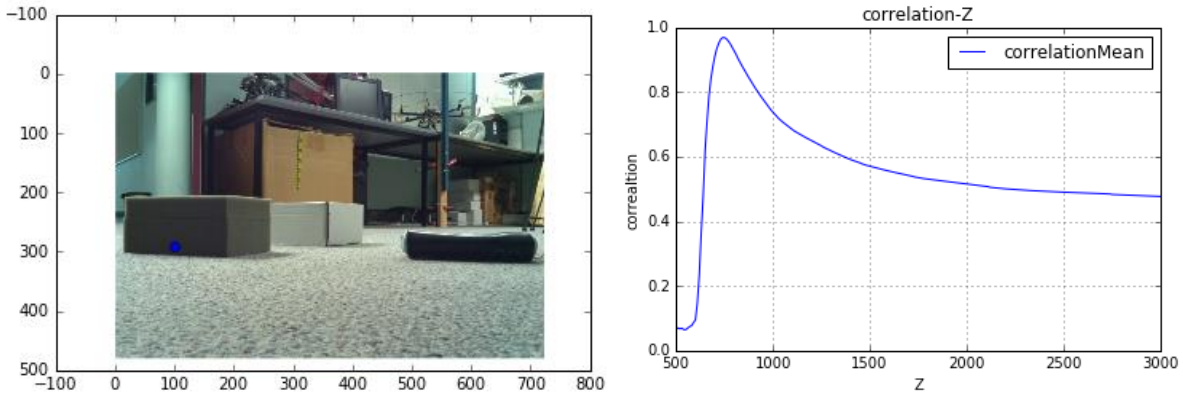


Figure 33 Point (100,290) in the Reference Image and Its Correlation-Z Curve

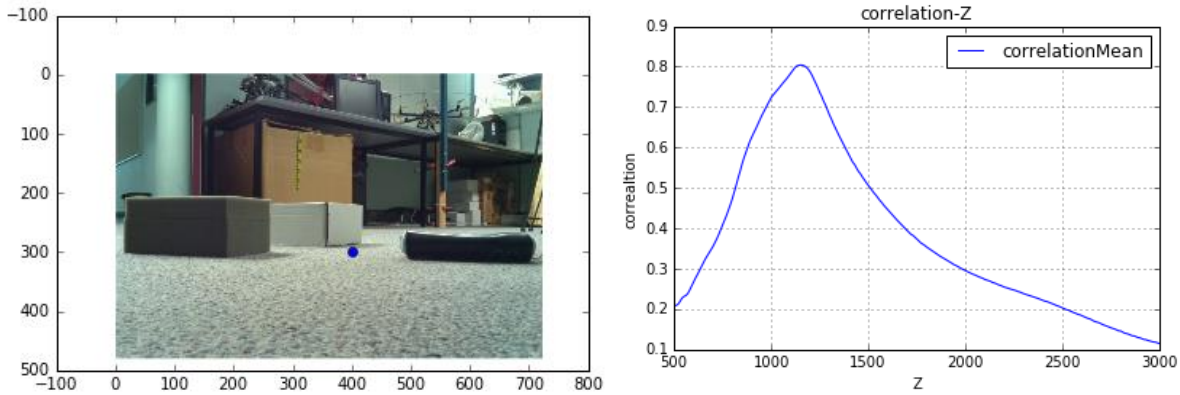


Figure 34 Point (400,300) in the Reference Image and Its Correlation-Z Curve

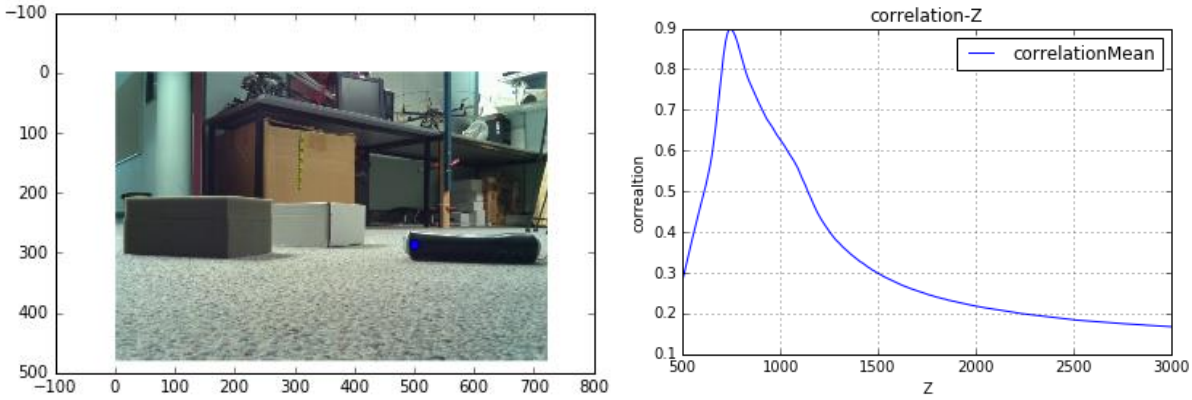


Figure 35 Point (500,285) in the Reference Image and Its Correlation-Z Curve

The three curves above illustrate the most cases when picking a pixel with depth from 500 millimeters to 3000 millimeters. The curves are monotone increasing before reaching the peak, then monotone decreasing after reaching the peak. In this case, it is very easy to do dichotomy. Let's take the third curve above as an example.

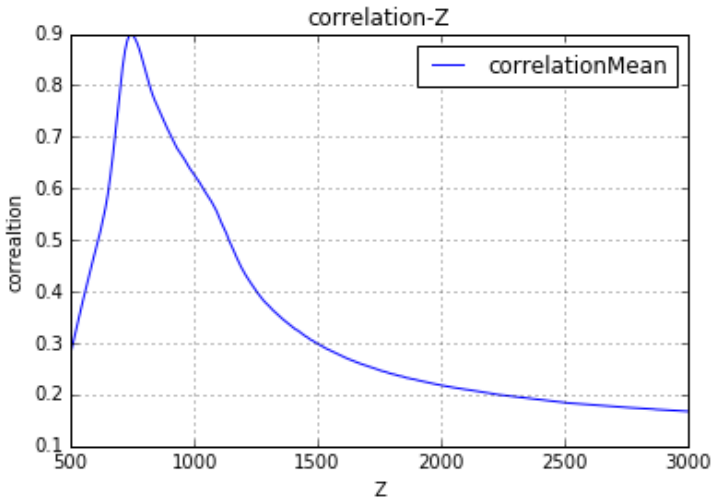


Figure 36 Correlation-Z Curve of Point (500,285) in the Reference Image

At the beginning, each Z value has a correlation value. The Z value is from 500 millimeters to 3000 millimeters. We know nothing about what is the Z value with the highest correlation.

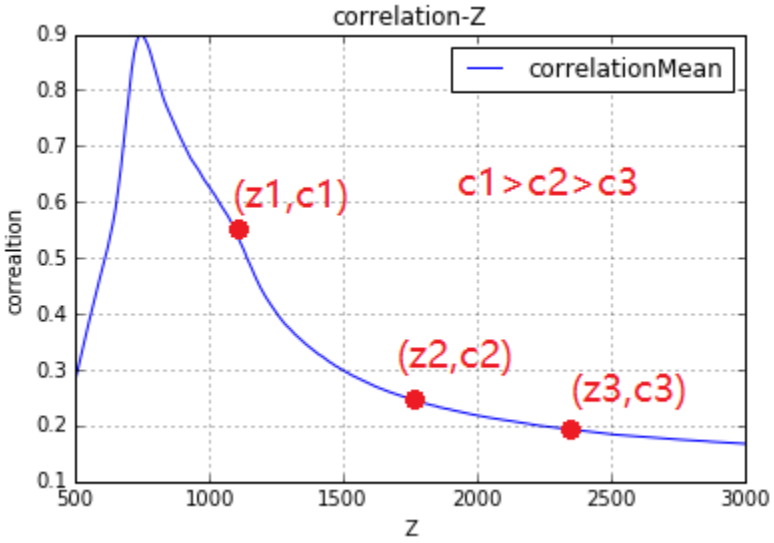


Figure 37 Dichotomy 1

We take the three quartiles on the range of Z values, z1, z2, and z3. They have correlations c1, c2, and c3. We see that c1 is the highest correlation among the three quartiles' correlation.

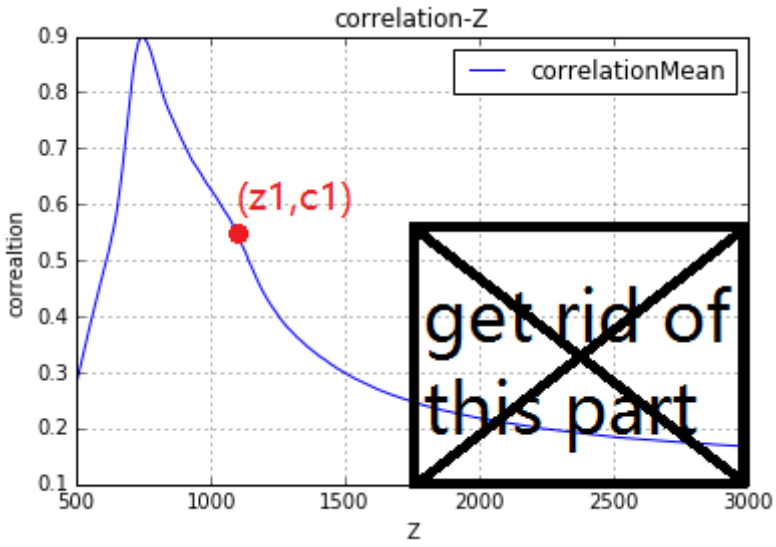


Figure 38 Dichotomy 2

Since we know the curve is monotone increasing before the peak and monotone decreasing after the peak, the peak must be around the quartile with the highest value of correlation. We can get rid of the Z values from the other two quartiles to the far away directions from the quartiles with the highest correlation values. That means, half of the Z value range can be considered as Z values with low correlations.

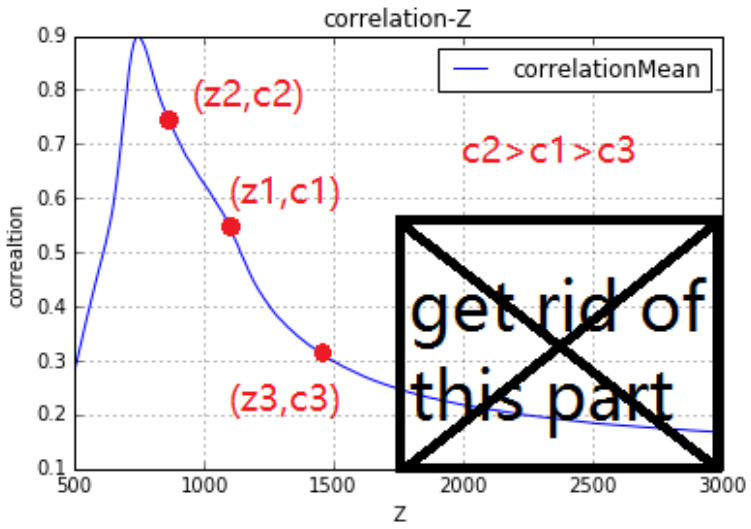


Figure 39 Dichotomy 3

Within the remained Z value range, we can take the three quartiles again, and compare the correlations of the three Z values again just like before.

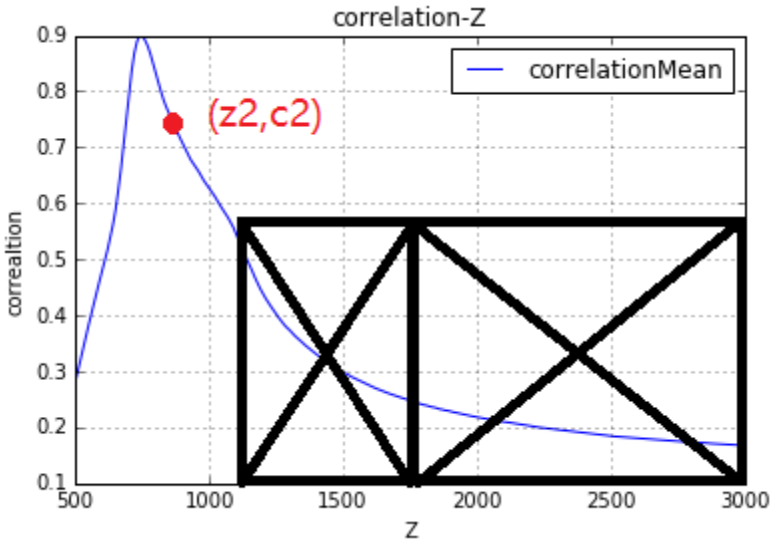


Figure 40 Dichotomy 4

By doing this, we again get rid of half of the Z value range from the remained Z value range. Now we have already gotten rid of ¾ Z value range from 500 millimeters to 3000 millimeters.

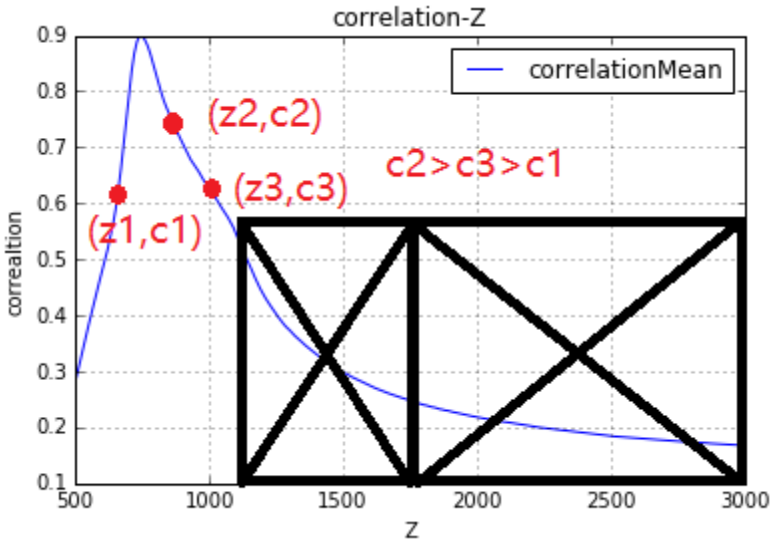


Figure 41 Dichotomy 5

Each time, we take the three quartiles in the remained Z values and compare the three correlations of them and find out the Z value with the highest correlation among the three quartiles. Then we can get rid of half of the remained Z value range.

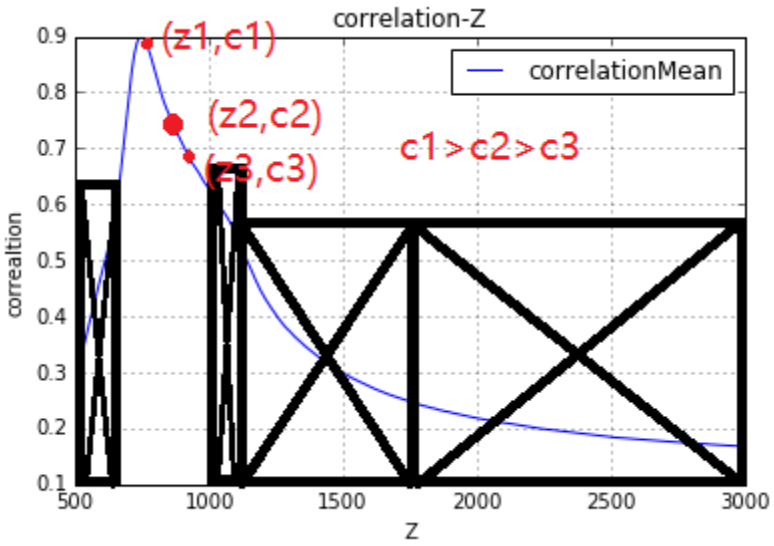


Figure 42 Dichotomy 6

When the remained Z value range is small enough, we can take the three quartiles, and use the Z value of the quartile with the highest correlation as the final depth information of the pixel in the reference image. By doing this, a high resolution of depth can be ensured. For example, when we

have remained Z values range between 20 millimeters, we can use the quartile with the highest correlation from the 20 millimeters of Z values as the final depth. Then a resolution of 5 millimeters is ensured.

From 500 millimeters to 3000 millimeters, if we do not use dichotomy and set the Z step as 5 millimeters, we need to try 500 Z values. By using dichotomy, we only need to try about 24 Z values. The time of running the algorithm for the whole image using dichotomy is shortened to the 1/20 of the original time in which we do not use dichotomy.

It is good if all the correlation-Z curves are monotone increasing before the peak and monotone decreasing after the peak. However, the correlation-Z curve can be sometimes complicated like the curve shown below.

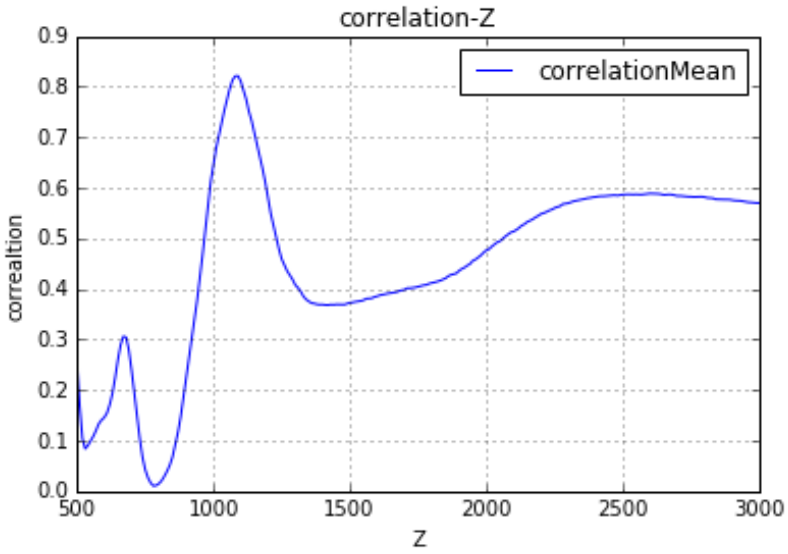


Figure 43 Correlation-Z Curve of Point (320,240) in the Reference Image

In this case, we cannot simply use the dichotomy method introduced above because we may sometimes miss the peak when getting rid of some range of Z values just as shown in the two below images.

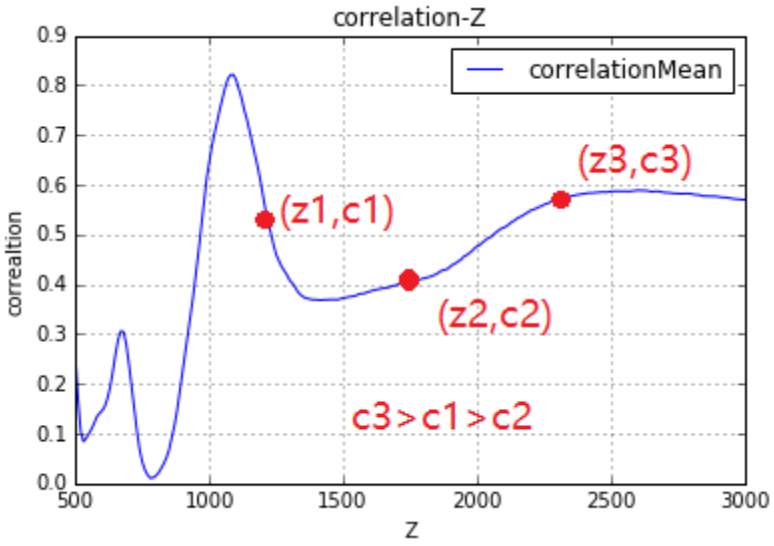


Figure 44 Dichotomy Error 1

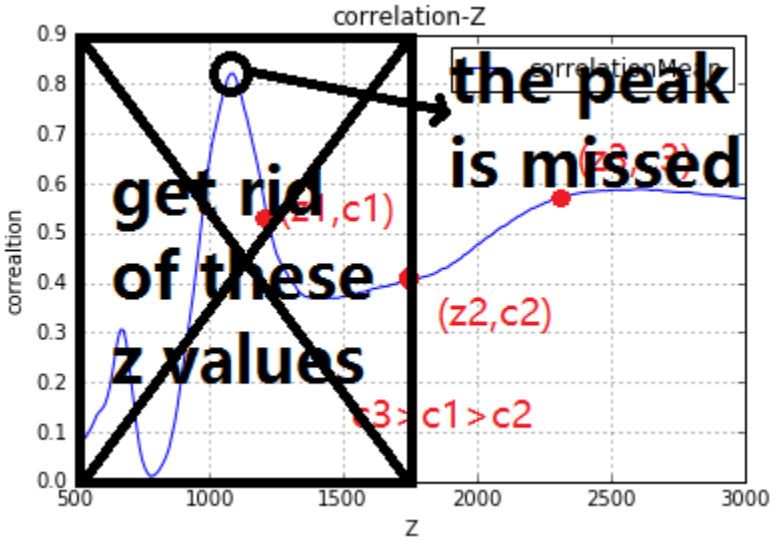


Figure 45 Dichotomy Error 2

We want to find a way to use dichotomy as while as do not miss the peak.

ADVANCED DICHOTOMY

By observing a huge number of correlation-Z curves, I found that they have a similarity: The width of the pulse which includes the peak was always greater than 100 millimeters.

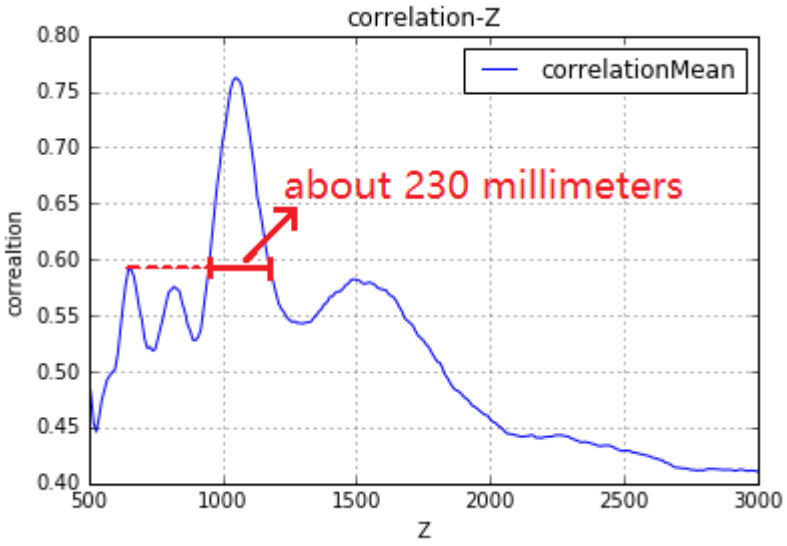


Figure 46 Width of the Pulse is Smaller than 100 Millimeters

As shown in the correlation-Z curve above, when we get the peak which has the second highest correlation value among all the peaks, and if we find the two Z values with the same correlation value from the pulse including the highest peak, we see that the distance between the two Z values is about 230 millimeters. That is greater than 100 millimeters.

That means if we firstly set 80 millimeters as the Z step, and then use dichotomy method for these Z values with the step of 80 millimeters, we will not miss the peak. This method is called advanced dichotomy in this project. Back to the example in which the simple dichotomy methods does not work. This time, we use advanced dichotomy and see what happens.

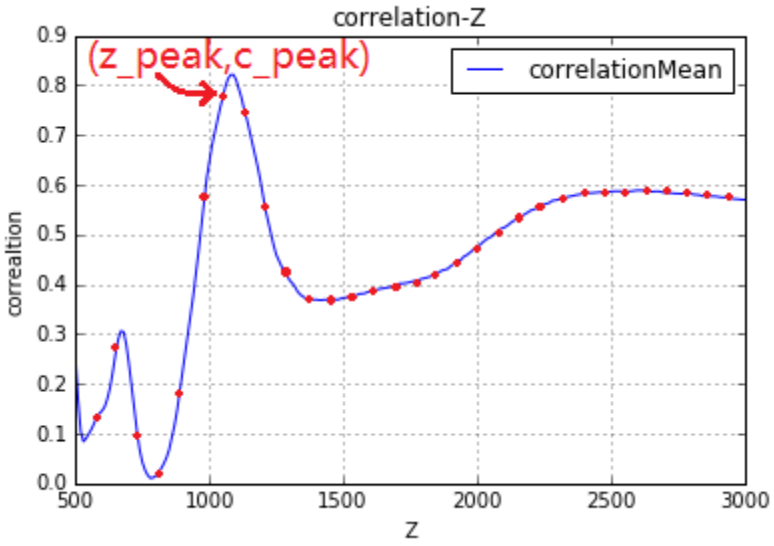


Figure 47 Advanced Dichotomy 1

Firstly, we get all the Z_80 values with the step of 80 millimeters. And get the Z_80_peak value with the highest correlation. Now we have the depth value of the pixel with the resolution of 80 millimeters.

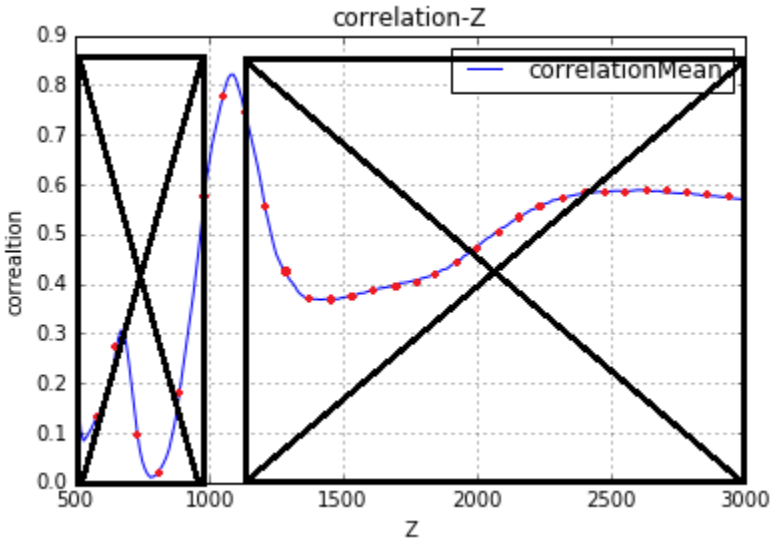


Figure 48 Advanced Dichotomy 2

Then, we get rid of Z ranges so that only the Z range near the Z_80 with the highest correlation remains.

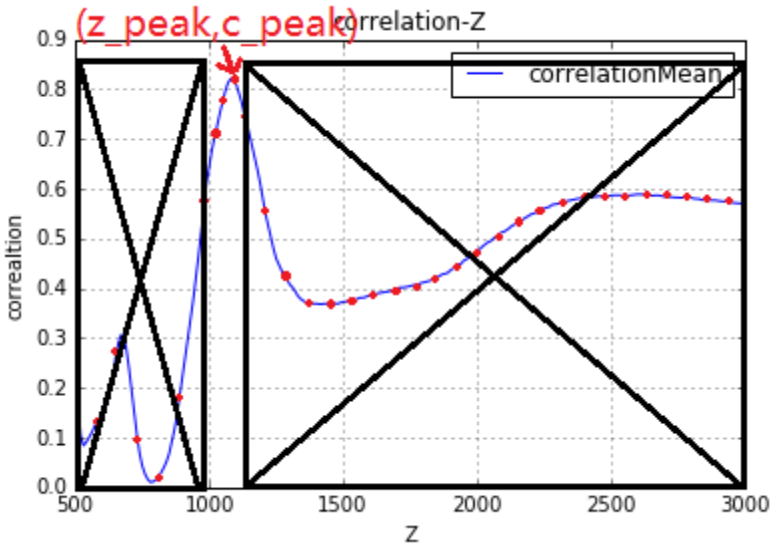


Figure 49 Advanced Dichotomy 3

After that, we find the three quartiles on the remained Z range and pick the Z value with the highest correlation among the three. Now we have the depth value of the pixel with the resolution of 40 millimeters.

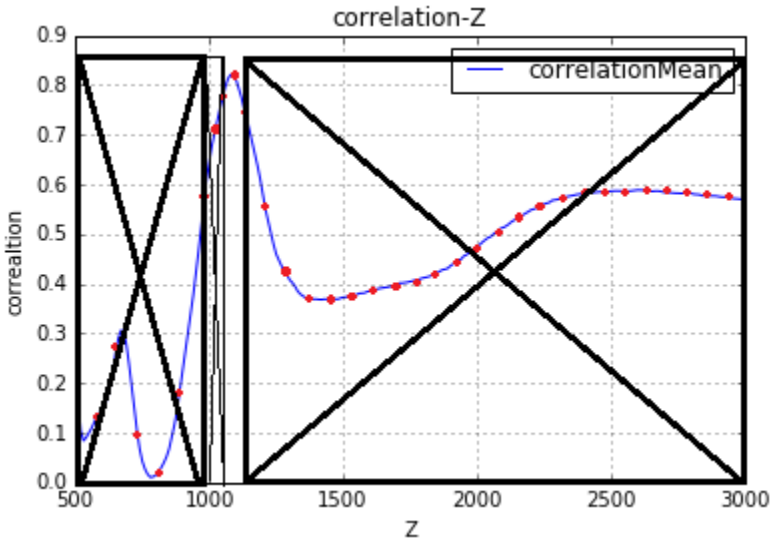


Figure 50 Advanced Dichotomy 4

Repeating the steps above, we can furtherly get the depth value of the pixel in the image with a resolution of 20 millimeters, 10 millimeters, 5 millimeters etc.

Assuming we are looking for the Z value of the pixel from 500 millimeters to 3060 millimeters, using advanced dichotomy, we need to try (3060-500)/80-1+3+3+3+3 = 43 different Z values to get the final depth value of the pixel with a resolution of 5 millimeters. Remember using simple dichotomy method we need to try about 25 different Z values to get the depth value with a resolution of 5 millimeters. The advanced dichotomy is, therefore, a little slower than the simple dichotomy but it ensures a very low error rate of the depth values.

THRESHOLDING

Using thresholding method can furtherly shorten the time for running the algorithm as while as keep the almost same quality of the result. The images below are the source image and its 3D point cloud (top view) generated by using advanced dichotomy.

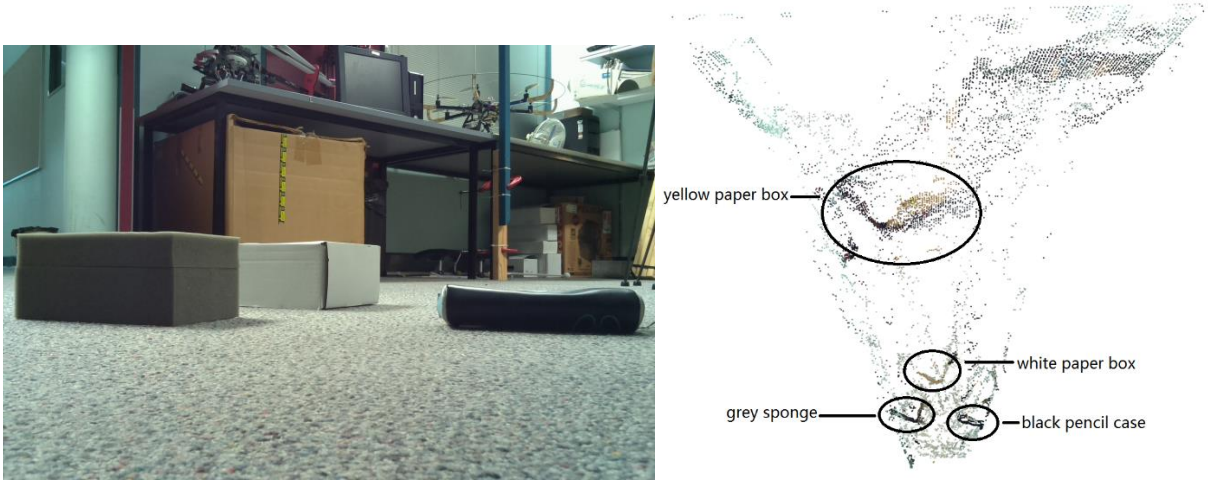


Figure 51 Original Image and Its 3D Point Cloud

As we see, the three objects' (grey sponge, white paper box, and black pencil case) depth are generated well. They are within the depth from about 0.5 meters to 1 meter. However, for farther objects like the yellow paper box, the generated depth information becomes bad. It is easy to understand because the vertical and horizontal distances between the cameras used in this project are only about 2.7 centimeters. It is too small for dealing with objects with large depth values.

Therefore, the performance of the algorithm decreases when the depth of the object increase. That means, when the depth is large, we do not need a high level of resolution because the influence of error is greater than the resolution.

In another word, we want to have a higher resolution for objects with smaller depth. The thresholding method helps us here. For the pixels with a depth greater than 2500 millimeters, the algorithm stops at having the depth value with a resolution of 40 millimeters. For the pixels with depth from 1500 millimeters to 2500 millimeter, then algorithm stops at having the depth value with a resolution of 20 millimeters. For the pixels with a depth smaller than 1500 millimeters, the algorithm runs until having the depth value with a resolution of 5 millimeters. Using this thresholding method, the time of running the algorithm can be further shortened because the number of Z values tried for pixels with great depth is reduced.

Calibration

For Camera Calibration and Stereo Calibration, I used 38 images for each of the cameras. The well-defined pattern was a 9*6 chessboard (9*6 corners) with 37 millimeters as the side length of each square.

Stereo Reconstruction

I used camera1 and camera2 (two cameras on the top) for Stereo Reconstruction. The images captured by camera 1 (left view) were set as the reference images.

Multi-View Depth Reconstruction

The images captured by camera 1 were set as the reference images for Multi-View Depth Reconstruction.

The objects in the images were from 0.5 meters to 3 meters.

The step of the pixel was set as 5. That means in every 25 pixels in the reference image, one pixel was used for generating the 3D point cloud.

The window size was set as 20. That means the square window with side length 41 pixels was used for calculating the correlations.

INTER_LANCZOS4 was used as interpolation method since it provided the smoothest correlation-Z curve and did not take much more time at the same time.

RESULTS AND DISCUSSION

Three scene samples were used for comparing the 3D point clouds generated by Stereo Reconstruction and Multi-View Depth Reconstruction. Image groups of the three samples were all captured in the laboratory. We focus on the objects from depth 0.5 meters to 3 meters.

SAMPLE1

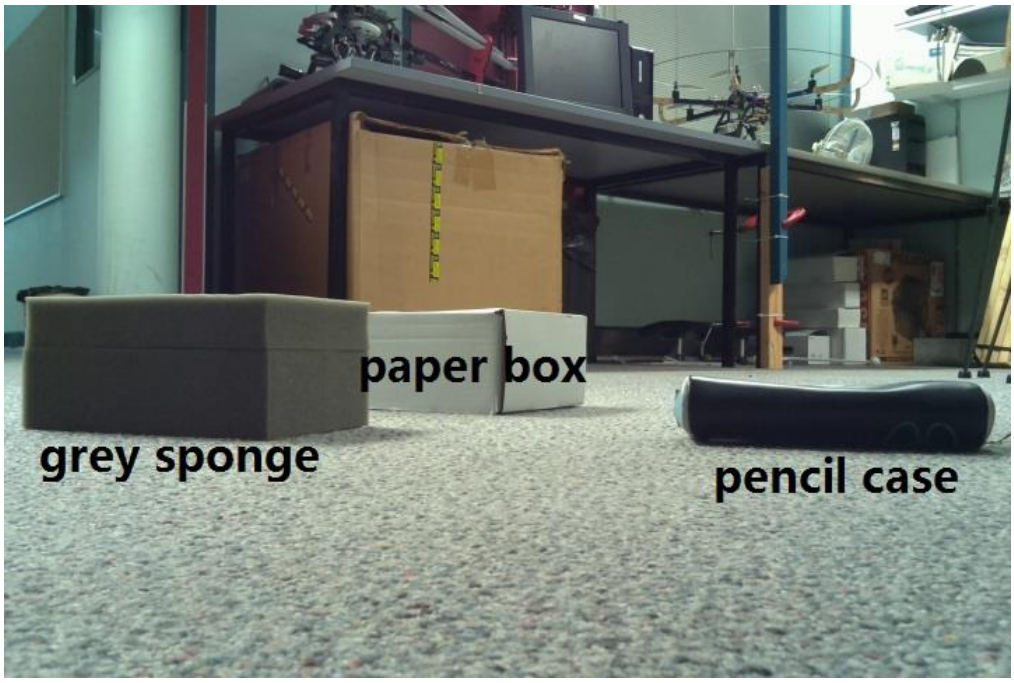


Figure 52 Sample1: Original Image

In the original reference image, there are three objects near the camera. A grey sponge, a white paper box, and a black pencil case.

PERFORMANCE

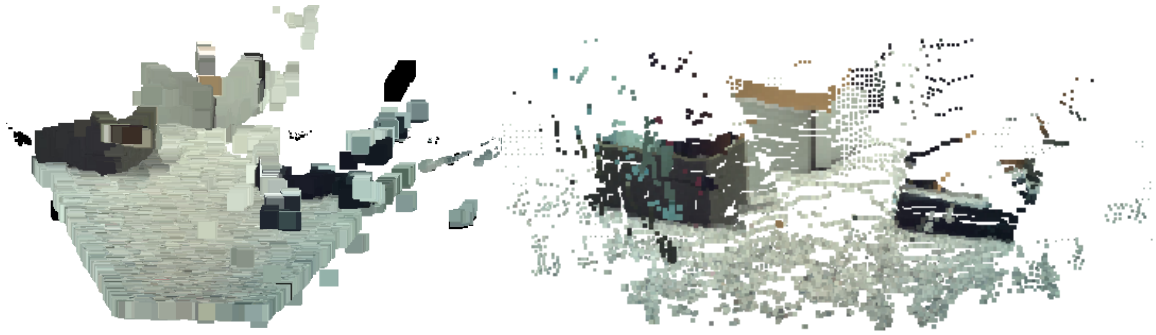


Figure 53 Sample1: 3D Point Clouds from Stereo Reconstruction and Multi-View Depth Reconstruction 1

The texture of the objects in the 3D point cloud generated by using Multi-View Depth Reconstruction is better remained. The corner in the 3D point cloud generated by using Stereo Reconstruction is not good because of the noise on the objects' surface in the 3D point cloud.

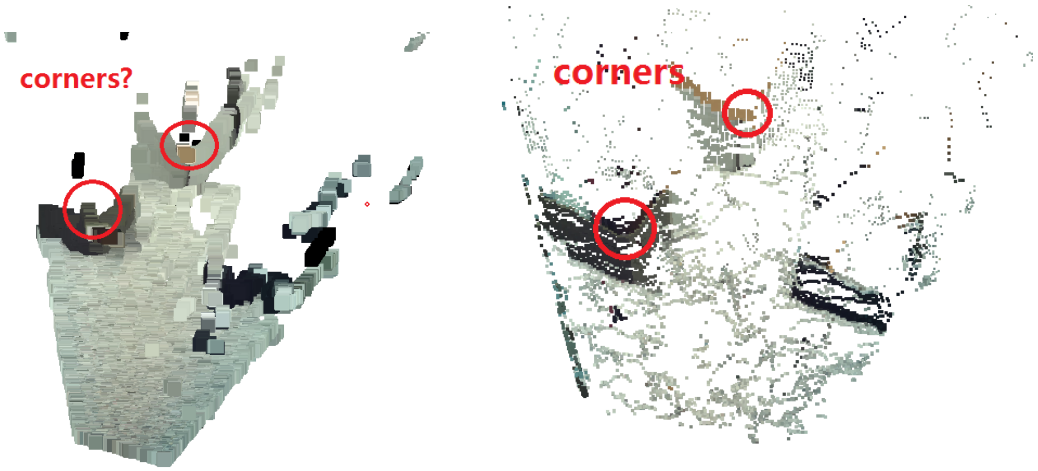


Figure 54 Sample1: 3D Point Clouds from Stereo Reconstruction and Multi-View Depth Reconstruction 2

The corners of the objects in the 3D point cloud generated by using Multi-View Depth Reconstruction are better. In the Stereo Reconstruction, we can hardly say the objects' corners are corners.

TIME

The running time of Stereo Reconstruction for Sample1 is 13.5120596602 seconds.

The running time of Multi-View Depth Reconstruction for Sample1 is 4156.01385458 seconds.

The running time of Multi-View Depth Reconstruction is much longer than the running time of Stereo Reconstruction.

SAMPLE2

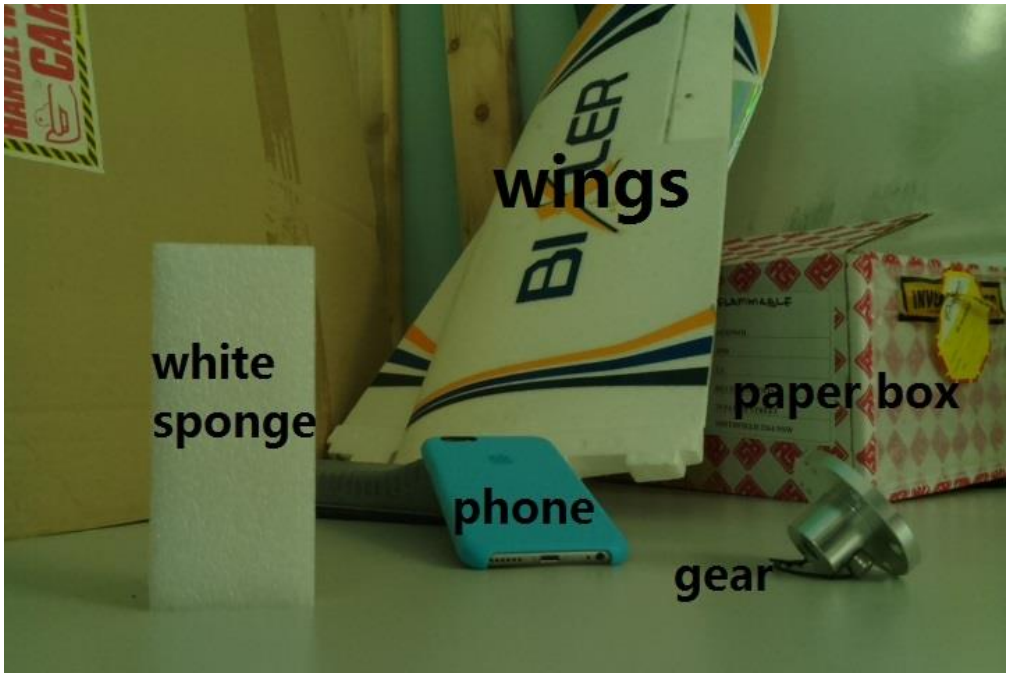


Figure 55 Sample2: Original Image

PERFORMANCE

In the original reference image, there are five objects near the camera. A gear, a smart phone, a paper box, a white sponge and some wings of model airplanes.

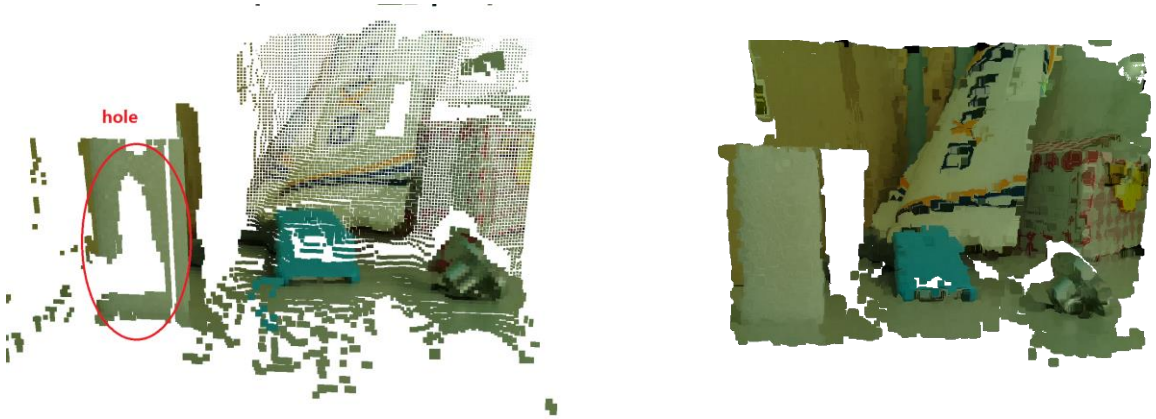


Figure 56 Sample2: 3D Point Clouds from Stereo Reconstruction and Multi-View Depth Reconstruction 1

The surface of the white sponges in the 3D point cloud from Stereo Reconstruction is quite good. But for the result of the Multi-View Depth Reconstruction, there is a big hole. The same problem happens to the blue phone and some area of the wings. The reason is that: The window size in the Multi-View Depth Reconstruction was set as 20, which means only 41*41 pixels were used for calculating the correlation near each pixel. So, this method becomes weak when dealing with a large area which is much greater than the 41*41 pixel window of the same color, because contend in the window will almost not change when sliding the window within this area.

Figure 57 Sample2: 3D Point Clouds from Stereo Reconstruction and Multi-View Depth Reconstruction 2

The depth reconstruction result of the wings is good in both Stereo Reconstruction and Multi-View Depth Reconstruction despite having some errors.

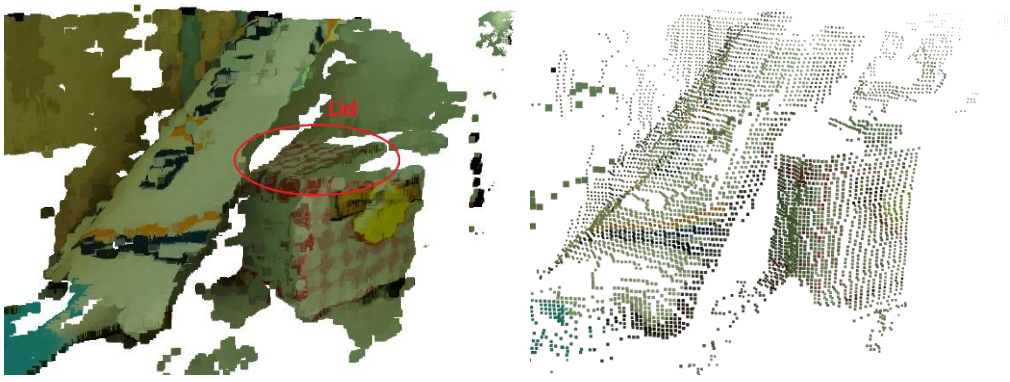


Figure 58 Sample2: 3D Point Clouds from Stereo Reconstruction and Multi-View Depth Reconstruction 3

Since only 1/25 pixels of all the pixels in the original reference image are used for Multi-View Depth Reconstruction, some details in the images are not generated well. For example, the lid of

the paper box is reconstructed well in the 3D point cloud from Stereo Reconstruction, while we can hardly see the lid in the 3D point cloud from Multi-View Depth Reconstruction.

TIME

The running time of Stereo Reconstruction for Sample1 is 12.7863427881 seconds.

The running time of Multi-View Depth Reconstruction for Sample1 is 4315.66430865 seconds.

The running time of Multi-View Depth Reconstruction is much longer than the running time of Stereo Reconstruction.

SAMPLE3

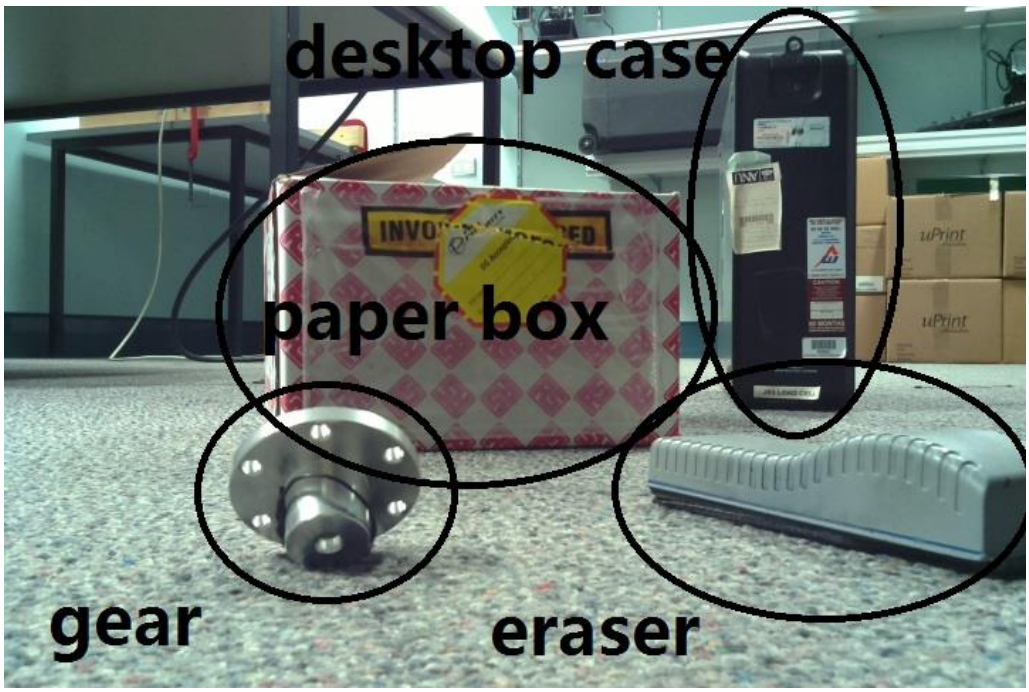


Figure 59 Sample3: Original Image

In the original reference image, there are four objects near the camera. A gear, a white board eraser, a paper box, and a desktop case.

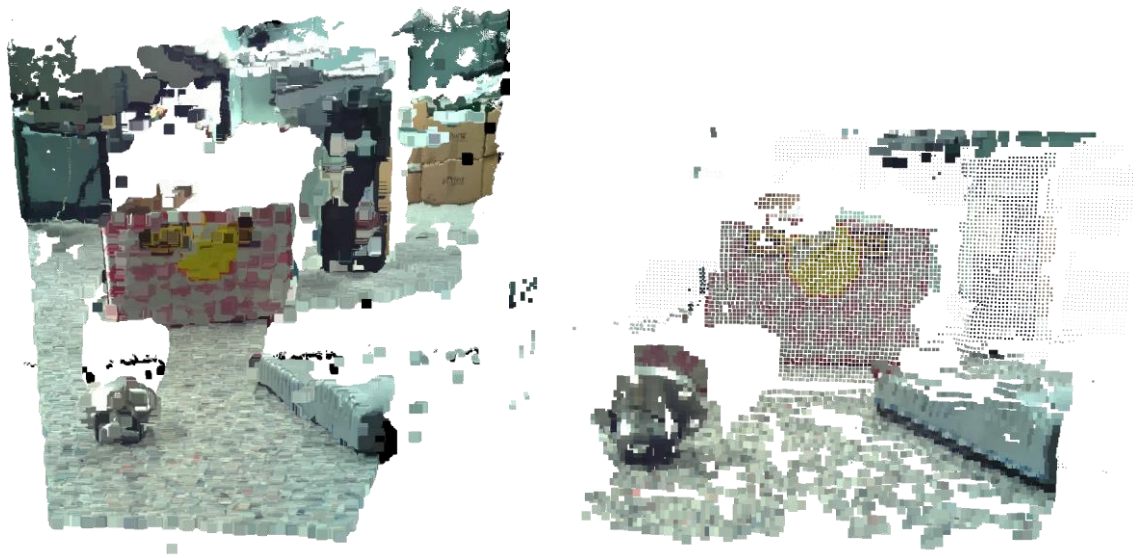


Figure 60 Sample3: 3D Point Clouds from Stereo Reconstruction and Multi-View Depth Reconstruction 1

The depth reconstruction of the gear and the eraser are good in both two 3D point clouds. We focus on the paper box and the desktop case which are farther from the camera.

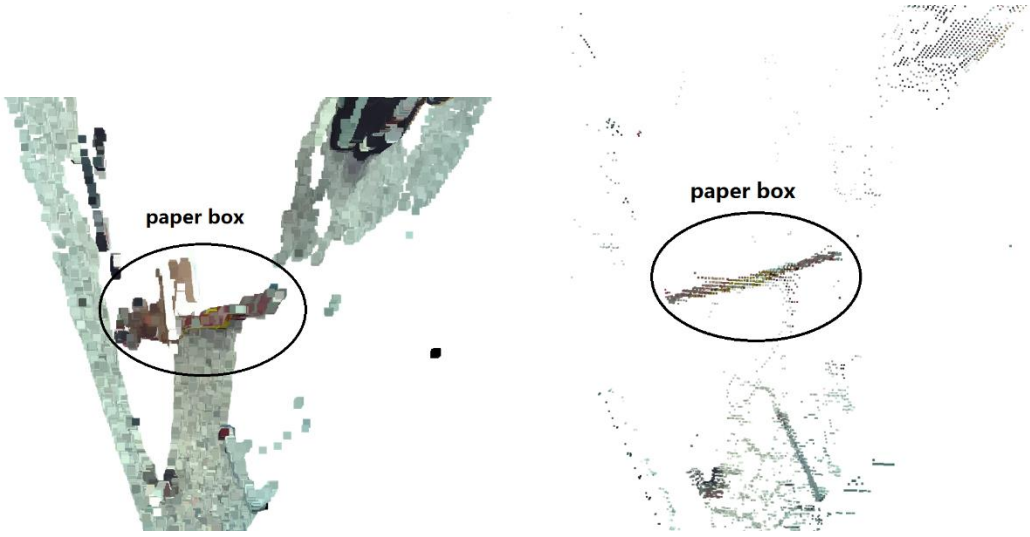


Figure 61 Sample3: 3D Point Clouds from Stereo Reconstruction and Multi-View Depth Reconstruction 2

The two images above are captured when viewing the two 3D point clouds from the top. As we see, the paper box in the top view of the 3D point cloud generated by using Multi-View Depth Reconstruction is converged as a straight line with a small thickness, while the paper box in the

top view of 3D point cloud from Stereo Reconstruction is messier. The result of Multi-View Depth Reconstruction is better than the result of Stereo Reconstruction for this paper box.

Let's look at the desktop case.

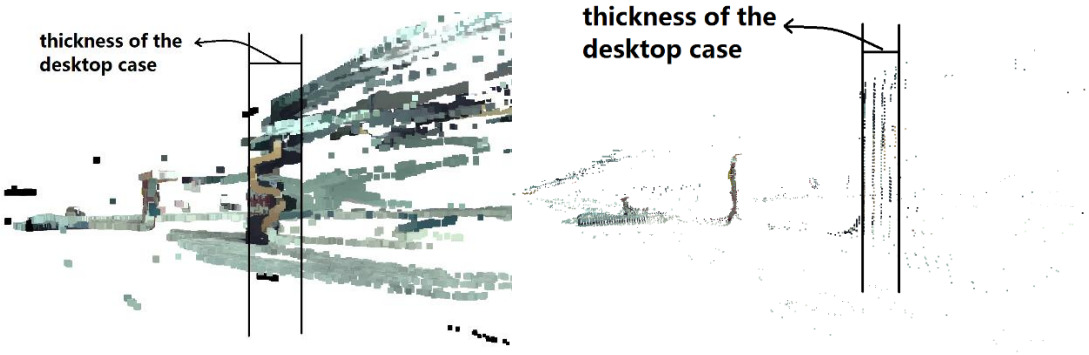


Figure 62 Sample3: 3D Point Clouds from Stereo Reconstruction and Multi-View Depth Reconstruction 3

The two images above are the side view of the 3D point clouds with the same scale. As we see, the thickness of the desktop is greater in Stereo Reconstruction case. Therefore, the result of Multi-View Depth Reconstruction is better than the result of Stereo Reconstruction for this desktop case.

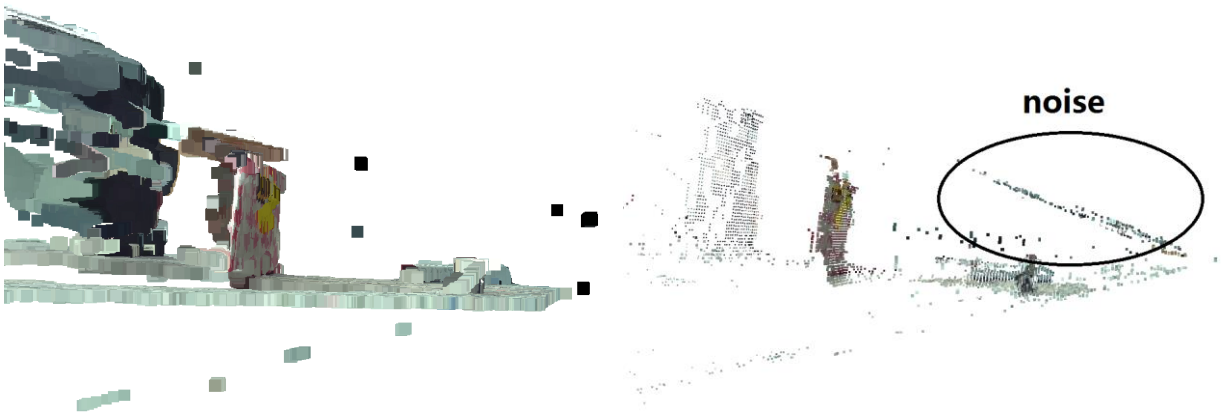


Figure 63 Sample3: 3D Point Clouds from Stereo Reconstruction and Multi-View Depth Reconstruction 4

The two images above show that there are lots of noise in the result of Multi-View Depth Reconstruction when the depth is small.

TIME

The running time of Stereo Reconstruction for Sample3 is 12.9792947905 seconds.

The running time of Multi-View Depth Reconstruction for Sample3 is 4142.75927341 seconds.

The running time of Multi-View Depth Reconstruction is much longer than the running time of Stereo Reconstruction.

CONCLUSION AND FURTHER WORK

By comparing the 3D point cloud generated by Stereo Reconstruction and Multi-View Depth Reconstruction method in this project, the conclusion can be made as following.

Comparing with using only two cameras, using camera array with more than two cameras makes the reconstruction result in most cases. By taking advantage of using four cameras instead of two cameras, the error rate is reduced so that the depth information for most pixels are more accurate to the reality. The concept of Stereo Reconstruction does not work when using Multi-View Depth Reconstruction, so new methodologies should be developed for Multi-View Depth Reconstruction. The Multi-View Depth Reconstruction methodology used in this project returns better results than Stereo Reconstruction. However, it still has technical drawbacks. The window size for correlation cannot be selected as too large because it significantly influences the running time of the program. The Multi-View Depth Reconstruction method in this project is therefore not robust for dealing with objects with a large area of the same color. The pixel step is another parameter which significantly influences the speed of the algorithm and it cannot be chosen as too small. A lot of pixel information is therefore gotten rid of so that the 3D point cloud does not contain so many details as the original reference image. The Multi-View Depth Reconstruction method used in this project also takes too much time. It cannot be used for the real-time applications.

There are some optimizations could be made, however, they were not achieved in this project due to the time limitation. Firstly, the color of the four images captured by the four cameras was not identical. It is probably because the time of exposure of each camera is different since their positions and nearby scenes are different. Working out a solution for this problem may make the performance of this algorithm better. Secondly, the correlation-Z curve of the camera pair (camera1, camera4) was more reliable since the distance between these two cameras was larger than the camera distance of (camera1, camera2) and (camera1, camera3), but the algorithm in this project simply treated the three correlation-Z curve as the same, and calculated the mean correlation values without applying any weight for each of the three curves. Finding a better way to get the mean correlation-Z curve is, therefore, another optimization can be made for this project. Also, trying to use camera arrays with greater camera distance is also necessary since it directly influences the effective distance of depth reconstruction. The thresholding method used in this

project was also simple. So, the further work includes applying some existing continuous functions which can make the depth resolution increase while the depth of the object decrease.

REFERENCES

Bruneau, G., Dubray, S., & Murguet, A. (2012, 2 13). *MATLAB Implementation of MonoSLAM.* Retrieved from ENSTA ParisTech: http://perso.ensta-paristech.fr/~filliat/Courses/2011_projets_C10-2/BRUNEAU_DUBRAY_MURGUET/monoSLAM_bruneau_dubray_murguet_en.html

Burden, R. L., & Faires, J. D. (1985). *Numerical Analysis.* PWS Publishing Company.

Foundation, R. P. (2016). *Getting Started with Picamera*. Retrieved from Raspberry Pi Learning Resources: https://www.raspberrypi.org/learning/getting-started-with-picamera/worksheet/

Hartley, R., & Zisserman, A. (2003). *Multiple view geometry in computer vision.* New York: Cambridge University Press.

Hirschmuller, H. (2008). Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Transactions on pattern analysis and machine intelligence, 30(2)*, 328-341.

Itseez. (2016). *OpenCV (Open Source Computer Vision)*. Retrieved from OpenCV: http://opencv.org/

Mordvintsev, A., & K, A. (2013). *Epiporlar Geometry.* Retrieved from OpenCV-Python Tutorials: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_epipolar_geometry/py_epipolar_geometry.html

opencv dev team. (2014, 11 10). *Depth Map from Stereo Images*. Retrieved from OpenCV 3.0.0-dev documentation: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html

opencv dev team. (2016, 11 15). *Camera Calibration and 3D Reconstruction.* Retrieved from OpenCV 2.4.13.1 documentation: http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

opencv dev team. (2016, 11 15). *Camera calibration With OpenCV*. Retrieved from OpenCV 2.4.13.1 documentation: http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

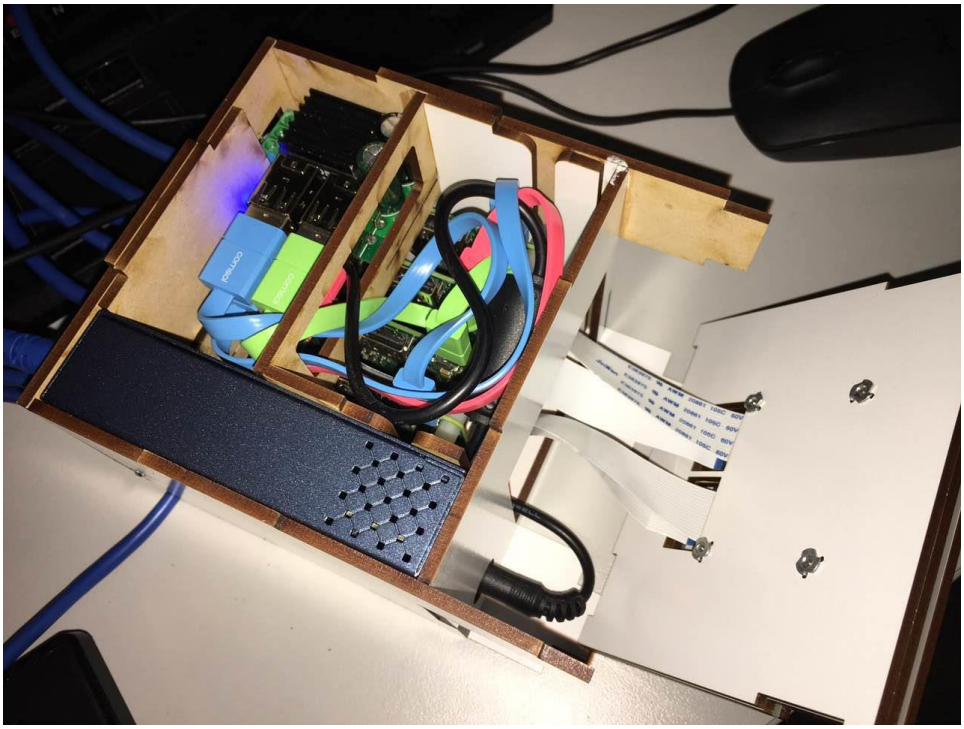
The Spyder Project Contributors. (2016). *Spyder - Documentation*. Retrieved from Spyder 3 documentation: https://pythonhosted.org/spyder/

Zhang, R., Guo, H., & Asundi, A. K. (2016). Geometric analysis of influence of fringe directions on phase sensitivities in fringe projection profilometry. *Applied Optics, 55(27)*, 7675-7687.

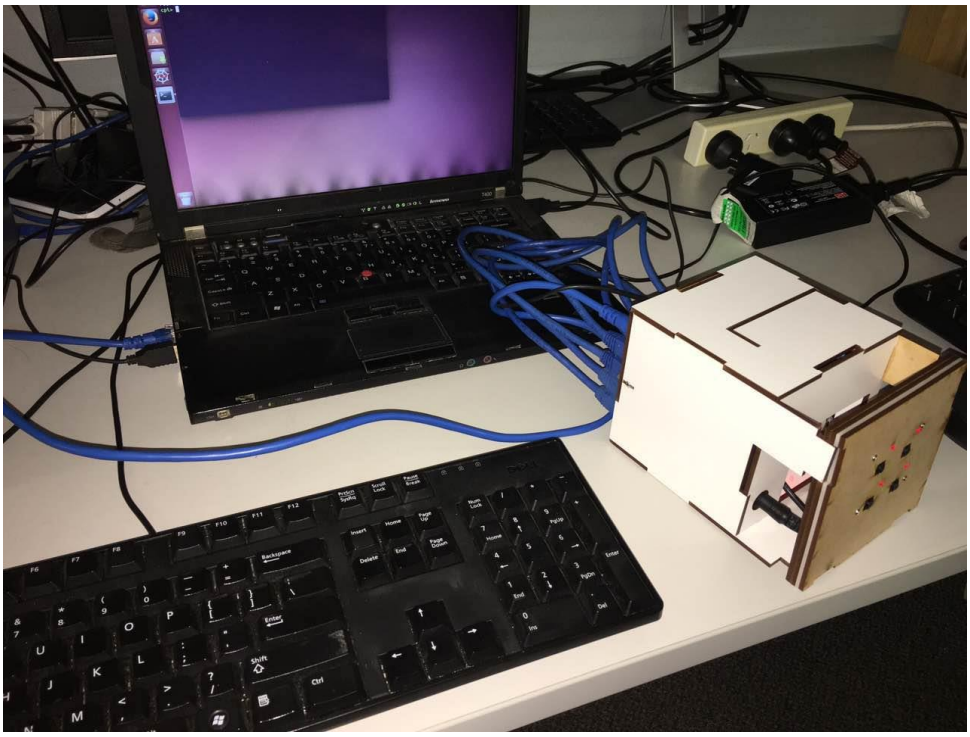
Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence, 22(11)*, 1330-1334.

APPENDIX

APPENDIX 1: THE CAMERA ARRAY MADE UP WITH FOUR RASPBERRY PI CAMERA MODELS



APPENDIX 2: CONNECTING THE CAMERA ARRAY TO A LAPTOP



APPENDIX 3: COMMAND FOR CAPTURING AND DOWNLOADING IMAGES