THESIS APPROVAL

GRADUATE SCHOOL, KASETSART UNIVERSITY

Master of Engineering (Information and Communication Technology for Embedded Systems)

DEGREE

Information and Communication Technology for Embedded Systems — **FIELD**

Electrical Engineering — **DEPARTMENT**

TITLE: Object Surface Area Approximation Using Low-Cost 3D Depth Sensor

NAME: Miss Amornrat Khongma

THIS THESIS HAS BEEN ACCEPTED BY

..... **THESIS ADVISOR**

(Mr. Miti Ruchanurucks, Ph.D.)

..... **THESIS CO-ADVISOR**

(Mr. Teera Phatrapornnant, Ph.D.)

..... **THESIS CO-ADVISOR**

(ProfessorYasuharu Koike, Ph.D.)

..... **DEPARTMENT HEAD**

(Assistant Professor Teerasit Kasetkasem, Ph.D.)

APPROVED BY THE GRADUATE SCHOOL ON .....

..... **DEAN**

(Associate Professor GunjanaTheeragool, D.Agr.)

THESIS

OBJECT SURFACE AREA APPROXIMATION USING LOW-COST 3D DEPTH SENSOR

AMORNRAT KHONGMA

A Thesis Submitted in Partial Fulfillment of
the Requirements for the Degree of
Master of Engineering (Information and Communication Technology for Embedded Systems)
Graduate School, Kasetsart University
2013

Amornrat Khongma 2013: Object Surface Area Approximation Using Low-Cost 3D Depth Sensor. Master of Engineering (Information and Communication Technology for Embedded Systems), Major Field: Information and Communication Technology for Embedded Systems, Department of Electrical Engineering. Thesis Advisor: Mr. Miti Ruchanurucks, Ph.D. 69 pages.

Currently, the technology of burn therapy is indispensible. "For patients with burns over 50% of the total body surface area, death rate is higher than 52%". This statistic of burn patients is from Nopparat Rajathanee Hospital in Thailand (2012). Burn Care in Nopparat Rajathanee Hospital and Computer-Vision Laboratory in Kasetsart University aim to apply depth sensor and computer vision theory for detecting and estimating of burn area ratio. We hope it will be more accurate than a present system that requires human estimation. We will generate the burn surface area using a low-cost 3D depth sensor device called Microsoft Kinect. As it is inexpensive, the resolution is lower than that of dedicated devices, such as 3D laser scanner. Here we need to understand basic of depth sensing and 3D triangular mesh reconstruction. The characteristic of Microsoft Kinecct is explained. In this thesis, we will be addressing the accuracy issue and propose a method to improve it. The improvement is done by spatial filtering. Finally, an experiment is done to compare accuracy of area size between using and not using our filter.

_____ _____ __ / __ / __

Student's signature Thesis Advisor's signature

ACKNOWLEDGEMENTS

I would like to gratefully thank and deeply indebted to Dr. Miti Ruchanurucks my thesis advisor for advice, encouragement and valuable suggestion for completely writing of thesis. I would sincerely like to thank Dr. Teera Phatrapornnant my thesis co-advisor from NECTEC, and also Dr.Yasuharu Koike my thesis co-advisor from Tokyo Institute of Technology for their valuable comments and suggestion.

This research is financially supported by Thailand Advanced Institute of Science and Technology - Tokyo Institute of Technology (TAIST-Tokyo Tech), National Science and Technology Development Agency (NSTDA), Tokyo Institute of Technology (Tokyo Tech) and Kasetsart University (KU).

I am especially appreciated my mother, my family and my friends for their continuing encouragements. Especially, Dr. Tawethong Koanantakool and Ms. Panjawee Rakprayoon, who also give valuable advice. Finally, I am deeply appreciated to Ms. Pornpawee Kerdkor who always devotes time and support during my graduate study.

Amornrat Khongma

March 2013

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

OBJECT SURFACE AREA APPROXIMATION USING LOW-COST 3D DEPTH SENSOR

INTRODUCTION

Currently, three-dimensional technology is widely used and is a popular topic among developers working on computer vision / 3D graphics applications. Such technology can be used in conjunction with several applications as follows:

1. Surveillance systems use it to detect people, for object indentifying, etc.
2. Robotic applications utilize it for advanced robots such as SLAM, mobile robot, etc.
3. Entertainment facilities apply it to develop graphic of an animation or for video-gaming and three-dimensional film.
4. Medical fields use it for diagnosis, surgery, and rehabilitation. The 3D scheme provides some advantage over traditional 2D image processing.

For input of 3D-technology, one conventional method is using stereo cameras. The principle of method is a trigonometry technique. This allows the camera to simulate human binocular vision, and therefore gives it the ability to capture three-dimensional images, a process known as stereo photography, Burn Care (2012). Many researches showed that it still has some error and instability in spite of that it requires several techniques in computer vision for stereo camera. Thus, stereo technique is imperfect. Partly, because the nature of stereo system that requires the use of multiple cameras. In this sense, it would be beneficial to use a depth camera instead of stereo cameras. We choose a new device called "Microsoft Kinect".

Microsoft Kinect has many pros; one is inexpensiveness. On the contrary, its cons are among precision. This research addresses the problem of increasing precision of Kinect in 3D mesh reconstruction. We will show that using a simple spatial filtering technique will greatly help improve the precision of Kinect.

What are we going to talk about Kinect? The topics are among, first, its characteristics, second, its calibration (camera model, color camera calibration, depth camera calibration, calibration between color and depth cameras). However, for our main objective of precision improvement we have to talk about other topics beforehand as will be stated below.

The motivation that leads to the main objective is our application. In this research, Burn Care in Nopparat Rajathanee Hospital and Computer Vision Laboratory in Kasetsart University aim to apply Kinect and computer vision theories for detecting and estimating burn area of human subjects. The hospital requires a good system to calculate ratio between burn area and whole body surface precisely. Statistics shows that death rate for patients with 50% burn area are 52 percent. The death rate will decrease if a patient gets the correct treatment in primary period. One crucial treatment procedure is giving a correct does of water to the patient. The dose value depends on burn area ration and can be computed by equation (1):

$$Doses = 4cc \times \%Burn \times Body\ Weight \quad (1)$$

As can be observed, to estimate the dose of water, physician must know the proportion between burn area and body surface. Nowadays, to measure the proportion, physician use eye estimation with a body chart called rule of nine to determine the percentage of burn area for each major section of body, Bouvrie (2011). However, the approximation of burn area is not accurate.

In order to improve the precision of this medical estimation, we focus on generating 3D mesh using point cloud from Kinect. Why? After generating 3D mesh we can find the ratio between burn area and body area. To generate the mesh, we use marching cube algorithm. However, using Kinect data directly, the mesh of simple one layer object surprisingly results in multiple layer of mesh. Hence we cannot find the ration of burn area from this mesh. Our research will show that applying median filter to the point clould prior to marching cube helps reduce the problem of multiple wrong mesh layers.

This document is organized as follow. Section 2 shows what is a brief objective in research? Section 3 is literature review part, consists of several methods to implement in this field, and refers to the procedure of each paper. Section 4 identifies the meterials to work in our system and explains the initial methods to develop in our system. It presents characteristics of Microsoft Kinect, basis of a comprehensive Kinect calibration method, and criterion of complete mapping 3D into 2D. It also describes the tri-state segmentation using watershed algorithm and Chebyshev's inequality. The 3D reconstruction methods and Heron's formula also are proposed in section 4. Section 5 shows more results and discussion. Finally, conclusion is in section 6. You will understand details more…

OBJECTIVES

1. To verify criteria and capability of a low cost depth device

2. To improve accuracy of a low cost depth device using computer vision techniques.

3. To apply such techniques for calculate surface area of sample object.

LITERATURE REVIEW

1. Field Survey Microsoft Kinect

There is another way to use the 3D information data. The CT scanner and Laser scanner are a most devices to take a high accuracy for improved reconstruction fields. Even if those devices are so expansive. However, the reconstruction accuracy depends on resolution of device and algorithm used. Thus, the selecting and improving of basic equipment is important for an enhancement device. The appropriate methods also aim to develop the final objective. We approach to search many papers before design steps of implementation.

Microsoft Kinect uses to generate a new system for preparing of burn care. The researchers try to understand a standard using and a basic of sensor. Such Microsoft Kinect Teardown provides a look inside this device. They show how to separate each equipment of Kinect carefully. And they also discuss about Microsoft and Project Natal. Kinect is a horizontal bar of sensors connected to a small base with a motorized pivot, and is designed to be positioned lengthwise below the video display. It can be followed as web site, Teardown (2010).

Khoshelham (2011), Accuracy Analysis of Kinect Depth Data, indicate vertify of the geometric quality of depth data. He confirms a based on the mathematical model of depth measurement by the sensor a theoretical error analysis. He also provides an insight into the factors influencing the accuracy of the data. It consists of the random error of depth measurement increases with increasing distance to the sensor, and ranges from a few millimeters up to about 4 cm at the maximum range of the sensor. He found accuracy of the data is influenced by the low resolution of the depth measurements. Figure1 illustrates the depth measurement from the speckle pattern.

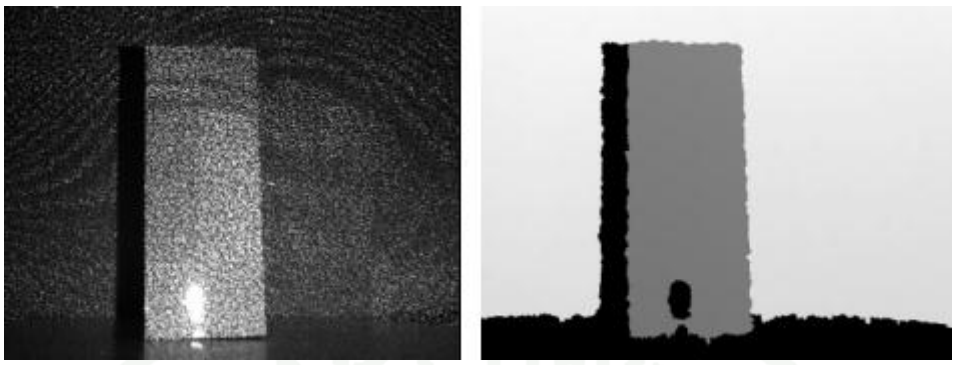


Figure 1 Left: infrared image of the pattern of speckles projected on the object; Right: the resulting depth image.

This paper also displays equation of triangular method of structure light sensor, Zlatka (2009). It is the basic mathematical model for the derivation. Thus, shadows and holes of Figure1 is a limitation of Microsoft Kinect as well, Bouvrie (2011), Khoshelham (2011). The error and imperfection in source (Microsoft Kinect) data may originate from three main sources. Fisrt is the sensor, the error in the estimation of the calibration parameters. Second is the measurement setup. It can be caused from in strong light, the laser speckle appear in low contrast in the infrared image. Finally, the limitation of usability depends on the properties of object surface that is not reflected IR infrared. He added the experimental result to describe the tests and discuss the results.

Before the usability of all of Kinect, we must to calibrate intrinsic parameters and extrinsic parameters of both cameras. K. Conley shows how to calibrate Kinect using lifefreenect library follows as, Conley (2011). On the web site, he added the calibration video that used the planar surface of chessboard. Brett Jones Rajinder Sodhi, Kinect-projector calibration, enables a host of projection-based interfaces. He built on his previous work in projector-camera calibration and structured light scanning, calibrated a Kinect sensor with a commodity projector. He took his existing projector-camera calibration software, added support for Kinect and created a simplified open source project that can be used for general projector-camera calibration (with support for the Kinect). More information about the projector-

camera calibration is available in separate documentation. He calibrates the internals of the Kinect using the ROS library. Finally, he presents a Kinect-projector visualization to demonstrate the calibrated results, Jones and Sodhi (2010).

Marek Solony demonstrates the capabilities of Microsft Kinect for a building dense 3D map of the indoor environments. The camera movement tracker is shown in this paper, exact camera position and rotation known in every time frame. He told it can be used to reconstruct a consistent map from multiple depth information. It also shows effectively produce dense 3D maps of small workspaces, Šolony (2011). From the metric depth, he used the 3D position of pixel with the respect to the IR camera. It can be computed using equations:

$$\begin{aligned} X_{ir} &= \frac{f_{xir}}{(x-c_{xir})d_m} \\ Y_{ir} &= \frac{f_{yir}}{(y-c_{yir})d_m} \\ Z_{ir} &= d_m \end{aligned} \qquad (1)$$

Where; x and y are position of the depth pixel in image, f_{xir} and f_{yir} are focal length, c_{xir} and c_{yir} are position of principal point of IR camera, and d_m is depth in meters. Both focal length and position of the principal point are estimated by calibration. Knowing the extrinsic rotation R and translation T between the RGB and IR camera, the mapping between color image and depth image can be expressed by following equations:

$$\begin{pmatrix} X_{rgb} \\ Y_{rgb} \\ Z_{rgb} \end{pmatrix} = \begin{pmatrix} X_{ir} \\ Y_{ir} \\ Z_{ir} \end{pmatrix} R + T \qquad (2)$$

$$x_{rgb} = \frac{X_{rgb} f_{xrgb}}{Z_{rgb}} + c_{xrgb}\ ,\ y_{rgb} = \frac{Y_{rgb} f_{yrgb}}{Z_{rgb}} + c_{yrgb} \qquad (3)$$

2. Segmentation Fields

The process of separate areas of burn patient is difficult to segmentation. The levels of burn have many degrees that the difference colors of burns are similar to the normal skin. The region growing of image segmentation is focused. Miti Ruchanurucks, Koichi Ogawara and Katsushi Ikeuchi explain Integrating Region Growing and Classification for Segmentation and Matting, Ruchanurucks *et al*. (2008). They present a supervised foreground segmentation method. It uses local and global feature similarity with edge constraint. The framework integrates and extends the notion of region growing and classification to deal with local and global fitness. It parameterizes constraint of growing using Chebyshev's inequality. The constraint is used to stop segmentation before matting. Matting relies on both local and global information. The proposed method outperforms many of the current methods in the sense of correctness and minimal user interaction, and it does so in a reasonable computation time. Our research implement follow as methods and programming for segmentation of burn surface area.

3. 3D Reconstruction Fields

3D reconstruction fields are one popular topic in computer vision. There are many steps to enhance depth information data. After the generate 3D scenes, the arrange of point cloud is connected between point to point and generate mesh of surface by Marching Cube algorithm. Cline and Lorensen (1987), research marching cubes: a high resolution 3D surface construction algorithm. It is a most popular to applied in this field. They present a new algorithm, which creates triangle models of constant density surfaces from 3D medical data. They create a case table that defines triangle topology. They process the algorithm in the 3D medical data in scan-line order and calculate triangle vertices using linear interpolation. They find the gradient of the original data, normalize it, and use it as a basis for shading the models. The surface models are the result of maintaining the inter-slice connectivity, surface data, and gradient information present in the original 3D data. They also discuss improvements that decrease processing time and add solid modeling capabilities.

The marching cube algorithm is added a basic tool in Meshlab program, Cline and Lorensen (1987), Jianhui *et al.* (2009).

After generate 3D reconstruction, the process of calculate burn surface area is summary many triangular meshes. We aim to find triangular area using Helon's formular. Wilson (1986), solves the problem of triangular area as defined "Helon formular". He demonstrates and proofs of Heron's formula can be done from elementary consideration of geometry and algebra. Alternative proofs and derivations are suggested on the Jwilson web site, Heron's Formula and a particularly concise geometric proof is given at Heron's Formula, Geometric Proof.

MATERIALS AND METHODS

Materials

Hardware

1. Personal computer, Intel Core i5 CPU 2.30 GHz, RAM 4 GB or up to the minimum specification for Kinect.
2. Microsoft Kinect.

Software

1. Microsoft Visual C++ 2008 Express Edition software
2. OpenCV (Open Computer Vision) library
3. OpenGL (Open Graphics Library)
4. OpenKinect (Open source Kinect Library)
5. SDL (Simple DirectMedia Layer Library)
6. MATLAB Simulation software
7. Meshlab

Other

Calibration objects (Chessboard) and Burn patients

Methods

1. System Overview

Hardware system consists of a Microsoft Kinect, as a color and depth sensor, and a computer. It will be used to capture images of burn patients. Depth and color images can be obtained from Kinect. We will show how to calibrate each sensors as well as how to calibrate extrinsic parameters between sensors.

Software system includes color image segmentation, color and depth images mapping, depth image filtering, and 3D mesh reconstruction. These steps are essential for generating burn area surface and normal skin surface. Accurate ratio between such two areas can help doctors decide correct doses of liquid for patients.

Among the mention algorithms, we need color image segmentation is important because captured images still compose of an environment and the burn patient. Thus we need to segment surface of environment and patient. Furthermore, we need to segment surface of burn area and normal skin as well. This is done using tri-state segmentation.

Color and depth images mapping is necessary because we perform tri-state segmentation in color images nevertheless do 3D reconstruction using depth images. It is quite straightforward to segment the burn area in color images. However calculating the burn area in 3D images is more accurate than doing so in color images. To achieve this, both images must be in the same views. This is done using extrinsic parameters from calibration state.

Depth image filtering is essential as Kinect's accuracy is low. We have tested that generating 3D mesh directly from Kinect data is errorness. The resulting surface of even a simple plane turns out into multiple layer of mesh. We propose to correct this using average filter.

Finally, 3D mesh reconstruction is indispensible because the ratio of burn areas is to be calculated base on such mesh. Then we sum up the areas of many meshes to represent the burn/skin areas. To generate the mesh, we use a well-known Marching cube algorithm.

This thesis is organized as followed. First a background of Microsoft Kinect is discussed and characteristics of Microsoft Kinect are explained. We will talk about physical components of Kinect, resolution of depth sensor, and the comprehensive Kinect calibration method. Calibration parameters are then used to map depth information to color image, Teardown (2010). And after that the tri-state segmentation of color image is performed to detect burn image area. The segmentation uses Watershed algorithm and Chebyshev's inequality, Khongma *et al.* (2012). After that we map depth information to color images, of only burn area and body surface. Next step is filtering and then triangular mesh reconstruction. The reconstruction algorithm used in this work is marching cube, Zlatka (2009). Finally, Heron's formula is used to calculate triangular area. The overall process can be illustrated as below Figure 2. It is divided into 3 states composing of pre-processing state, processing state and post-processing state (Figure 2).

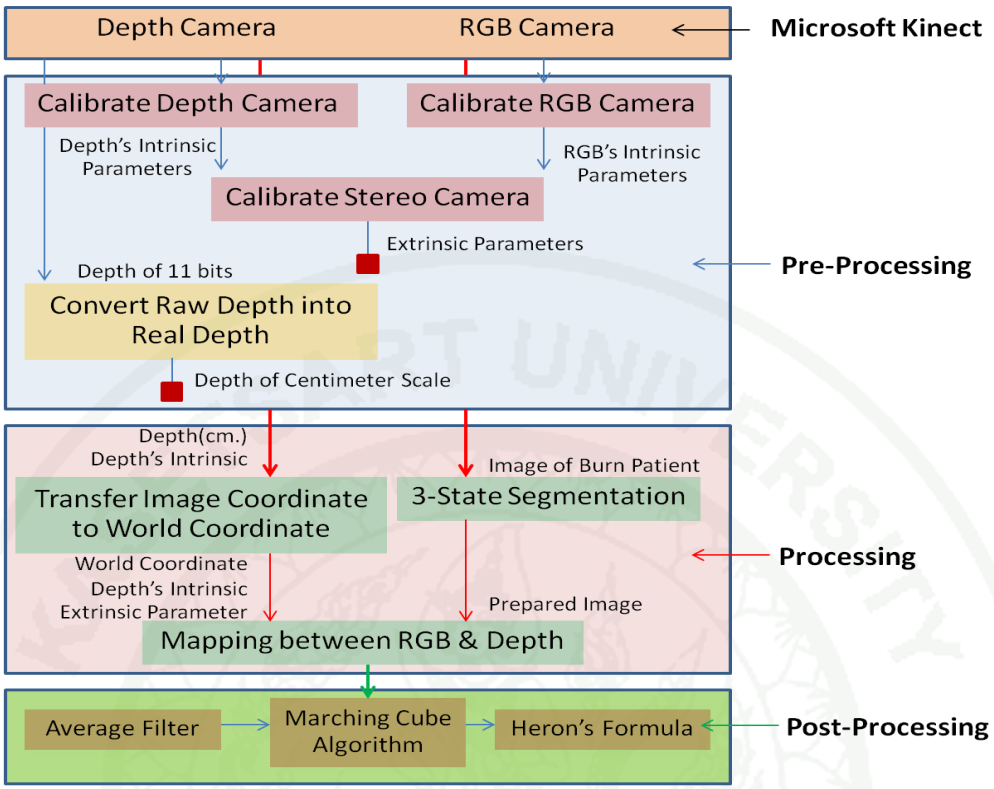


Figure 2 System Overview.

2. A Background and the Characteristics of Microsoft Kinect

Since November 4, 2010, Microsoft has released a new device that shows in North America Market. This devices is Kinect for Xbox 360, is a motion sensing input device for the Xbox 360 video game console, Wedro (2012). Kinect was developed in cooperation with PrimeSense and has a divergent concept than the Wii Remote and PlayStation Move: it creates a depth image, Ruchanurucks *et al.* (2008). Kinect is a new alternative gaming device that is capable to capture color and a depth image at the same time (Figure 4). It consists of RGB camera, 3D depth sensor: depth camera and IR projector, multi-array microphone, motorized tilt, Conley (2011) (Figure 3). It has been determined that the Kinect sensor outputs video at a frame rate of 30 Hz. It also has the image size around 640x480 pixels. It consist of RGB camera (Color CMOS- VNA38209015, Jones and Sodhi (2010)), 3D depth sensor: IR camera (IR CMOS-Microsoft / X853750001 / VCA379C7130, Jones and Sodhi (2010)) and

IR projector (IR Projector-OG12 / 0956 / D306 / JG05A, Jones and Sodhi (2010)), multi-array microphone (4 microphones), motorized tilt, Šolony (2011). The RGB video stream uses 8-bit VGA resolution with a Bayer color filter, while the monochrome depth sensing video stream is in VGA resolution with 11-bit depth, which provides 2,048 levels of sensitivity, Wedro (2012).

The depth measurement process is based on a triangulation process, Ruchanurucks *et al.* (2008), Cline and Lorensen (1987). From a technical perspective, the depth sensor uses a static Laser pattern projector and a CMOS camera to triangulate a dense depth map. Illumination intensity, baseline, depth of field and speed are tuned to cover a depth range that depends on several open source. The sensor can be use in typical indoor environments for the purpose of segmenting and 3D motion recognition, Jones and Sodhi (2010).

The resulting RGB value and point cloud can be loaded to a computer using open source libraries which enable Kinect to be operated with Windows, Linux or Mac, Bouvrie (2011). Data connection to the computer is through USB interface.

Figure 3 Microsoft Kinect.

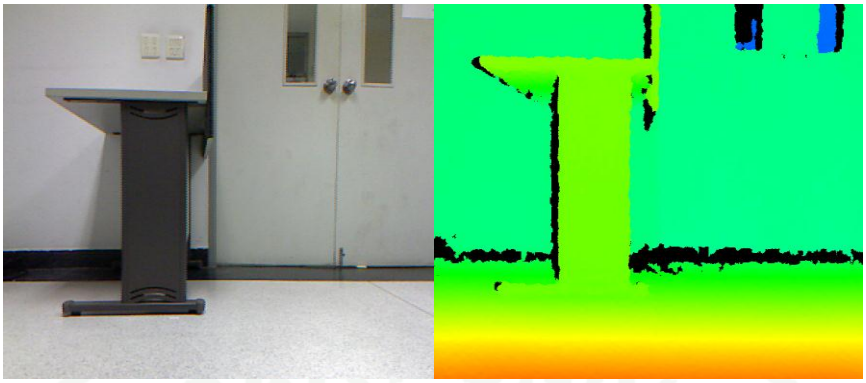


Figure 4 Color and depth images captured from kinect.

Kinect's mechanism is similar to 3D scanner that IR projector of Kinect is used for structured light scanning, Teardown (2010), Conley (2011). The principle of depth sensor is triangulation method (reference plane) which are reflected by objects and recaptured by the IR camera as follows:

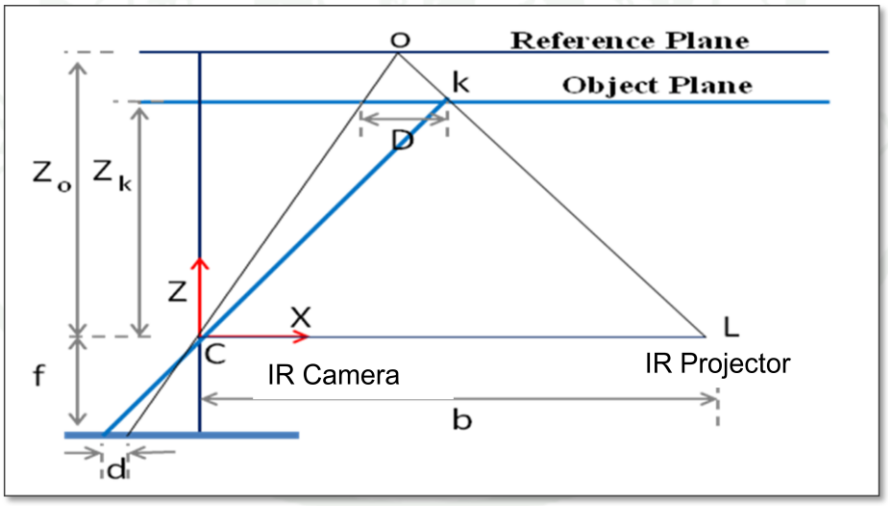


Figure 5 Triangulation methods (with structured light).

$$\frac{D}{d} = \frac{(z_o - z_k)}{z_0} \qquad (1)$$

$$\frac{d}{f} = \frac{D}{z_k} \qquad (2)$$

Where, equation (1) is a ratio of disparity and depth distance between object plane and depth sensor. Equation (2) is a ratio of intrinsic parameters and depth parameters. D is a disparity of object's position between reference and object plane. d is a disparity of object's position from the optic axis of intrinsic parameters. z_o is a depth of reference plane. z_k is a depth of object plane. Last, f is a focal length of IR camera.

However, limitation of depth sensor is still importance factor to modify 3D informations. Thus the error and imperfection of the Kinect possibly originated from two main issues:

1. The depth sensor: the random error is occurred from estimation of the calibration parameters.

2. The measurement setup: the IR laser speckle appears in low contrast in the infrared image at strong light.

Furthermore, as the depth sensor of Kinect used infrared method, some object surface does not reflect IR as they are absorption, transparent, translucent, or silky. For precision issue, Kinect cannot match with off-the-shelf laser scanner because Kinect's precision range is millimeter, not micrometer. Figure 6 shows the calculated standard deviations plotted with the depth distance from the plane to the depth sensor, Ruchanurucks *et al.* (2008). The random errors can be increase from a few millimeters at 0.5 m distance up to a few centimeters at the faraway range of the depth sensor. In fact, the random errors increase with the square distance from the depth sensor as follows:

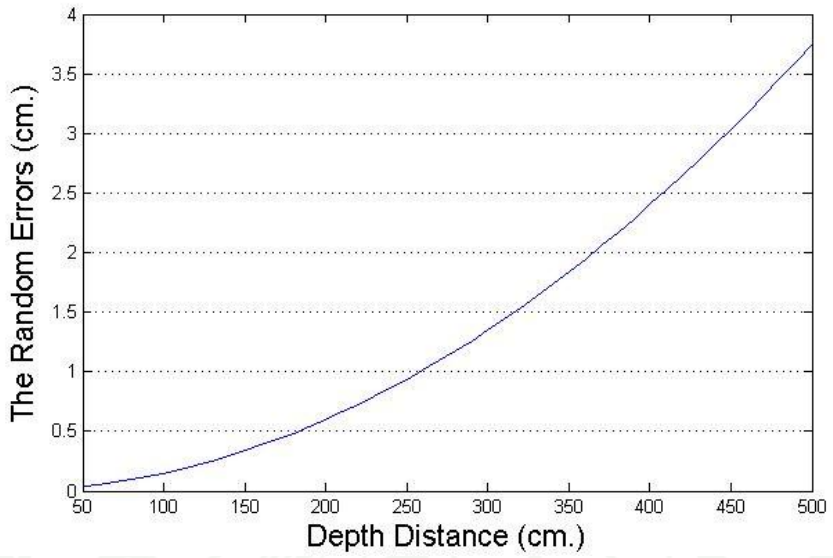


Figure 6 Shows the calculated standard deviations plotted with the depth distance from the plane to the depth sensor.

$$F(Z) = 1.5 \times 10^{-5} \times Z^2 \qquad (8)$$

Where; Z is a depth in real world coordinate. $F(Z)$is the error value of each point in the same plane. For example, at 50 cm distance, the random error is around ±0.375 mm of each point in the same plane, Ruchanurucks *et al.* (2008).

3. A Comprehensive Kinect Calibration Method

With an active scanning technique, an inexpensive Kinect is used in order to acquire RGB and depth information (Figure 4). Alignment and mapping between RGB and depth image is applied to calculate the burn surface area of patient. Prior to that, we must calibrate the equipment which can be called pre-processing state. The calibration consists of calibration of RGB camera (intrinsic parameters), calibration of depth camera (raw depth to real depth conversion, intrinsic parameters), and calibration between RGB and depth camera (extrinsic parameters). However, the basis of camera model is added to understand more.

3.1 Camera Model

3.1.1 Pinhole Camera Model

From Gary and Adrian (2008), we have to know a model of the camera's geometry and a model of lens distortion before calibrating a camera. All of these information models define intrinsic parameters. To estimate the intrinsic parameters, we have to know a physical of camera that light rays intersect a particular point in world coordinate through a pinhole camera and project onto an image plane as shown in Figure 7. Where parameter f is a focal length which is distance from the pinhole aperture to the image plane, Z is the distance from the pinhole camera to an object, X is the object's length and x is the object's image onto the image plane. Then, an Equation 1 can be written using similar triangles from the Figure 7.

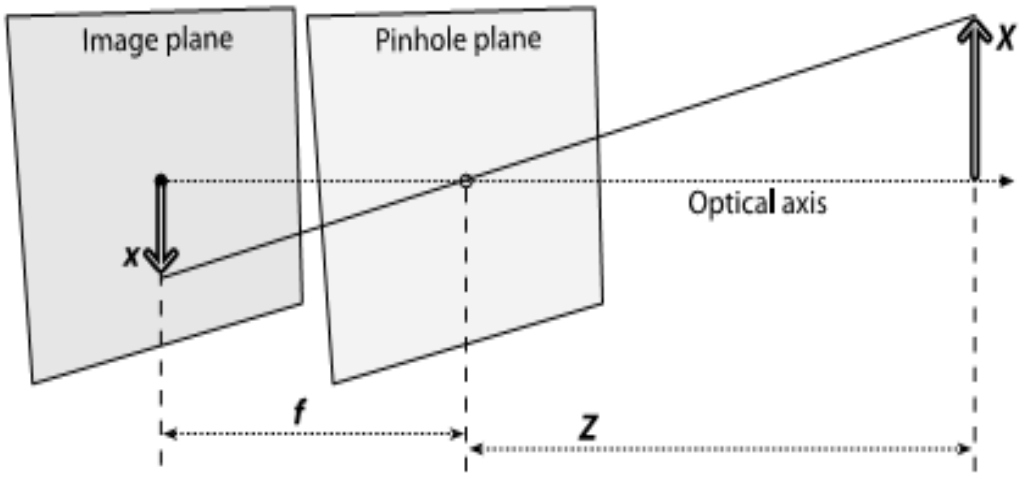


Figure 7 The pinhole camera model. The pinhole lets through light rays which intersect a particular point in space and project onto an image plane.

Source: Gary and Adrian (2008)

$$-x = f\left(\frac{X}{Z}\right) \qquad (9)$$

To eliminate the negative sign in Equation 1, the point in the pinhole is rearranged and denoted as the center of projection as shown in Figure 8. A point thatimage plane intersect onto the optical axis is referred the principal point. In fact, the principal point is not equivalent to the optical axis or not on the center of image. Then, $c_x$and $c_y$are defined as distance away from optical axis. In Figure 8, point Q in world coordinate is projected on point q in image plane as follow equation:

$$x = f_x\left(\frac{X}{Z}\right) + c_x \qquad (10)$$

$$y = f_y\left(\frac{Y}{Z}\right) + c_y \qquad (11)$$

Where, $f_x \; and \; f_y$ are the focal lengths of the camera.

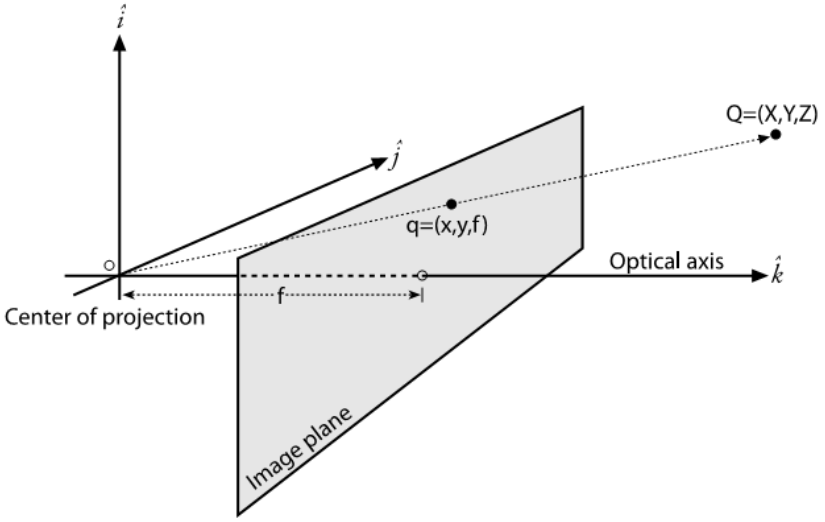


Figure 8 A point $Q = (X,Y,Z)$ is projected onto point $q = (z, y, f)$ in the image plane by the ray passing through the center of projection.

Source: Gary and Adrian (2008)

Also, the Equation 2 and 3 that can be rearranged the parameters into matrix form, is represented in Equation 4.

$$q = MQ \qquad (12)$$

$$\text{Where;} q = \begin{bmatrix} x \\ y \\ w \end{bmatrix}, M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, Q = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Matrix M in Equation 4 is determined as the intrinsics matrix of the camera. Matrix Q is the point of position in world coordinate that consist of parameter X, Y and Z. And, matrix q is the point of position in image coordinate that consist of parameter x, y (scale in pixel) and w equal to Z in matrix Q. For considering point q in 2D coordinate, Equation 4 must be divided through by w to transform point Q into point q in image plane. However, it has an effect by lens distortion that makes a distorted image. To eliminate the problem from lens distortion, we will focus on cause of the problem for getting the correct image in the next section.

3.1.2 Lens Distortions

An ideal lens would render straight lines as straight from the object point into image plane. In practice, most of lenses always bend a lot of light lines outwards (barrel distortion) or inwards (pincushion distortion) that is from the result of the shape of lens and is consisted of Radial distortions. For radial distortions in Figure 9, the distortion is zero at the center of image and increases as we move toward the periphery. For eliminating the error from radial distortions, Equation 5 and 6 are used to calculate a correct point ($x_{\text{corrected}}, y_{\text{corrected}}$) by inputting a distort point (x, y) into the equations.

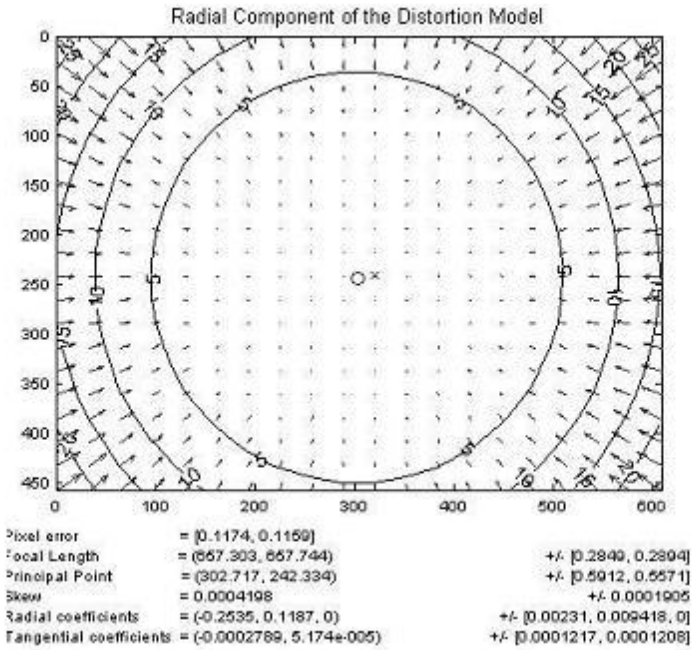


Figure 9 The arrows show location points of an external rectangular grid which is displaced in a radial distortion of image (courtesy of Jean-Yves Bouguet).

Source: Gary and Adrian (2008)

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (13)$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (14)$$

Where $k_1, k_2, and\ k_3$ are the requirements of coefficient parameters.

Another distortion is the tangential distortion which is due to defecting of the lens not being perfectly aligned. Then, it results in the displacement of image points perpendicular to a radius from the center of a field that is shown in Figure 9. For eliminating the errors from tangential distortions, Equation 7 and 8 are used to calculate a correct point ($x_{\text{corrected}}, y_{\text{corrected}}$) by inputting a distort point (x, y) in to the equations.

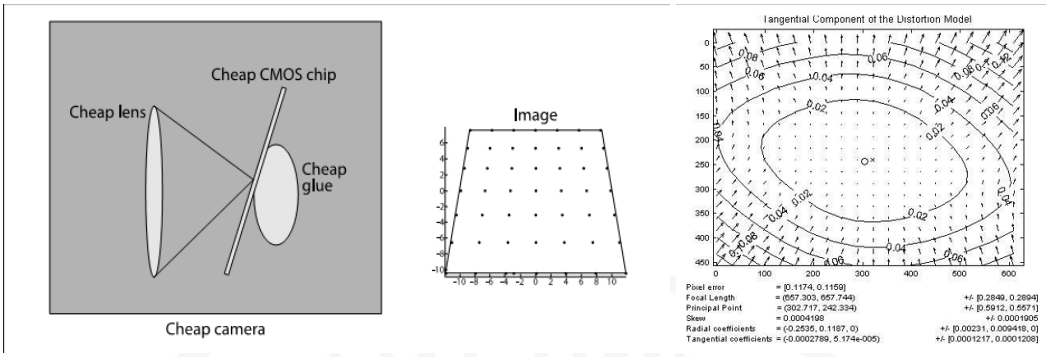


Figure 10 If the lens is not fully parallel to the image plane (left), then the tangential distortion can be occurred (middle). The arrows show location points on an external rectangular grid (right) which are displaced in a tangentially distorted image (courtesy of Jean-Yves bouguet).

Source: Gary and Adrian (2008)

$$x_{corrected} = x + [2p_1y + p_2(r^2 + 2x^2)] \quad (15)$$

$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2x] \quad (16)$$

3.2 Camera Calibration

3.2.1 Intrinsic and Extrinsic matrix

In order to locate point on image plane into world coordinate, Intrinsic and extrinsic parameters are needed to require. The intrinsic parameters compose of camera's parameter, such as focal length, principal point, and lens distortion. The extrinsic parameters compose of the 3D position and orientation of the camera. The process that identifies these two parameters is called *Camera Calibration*. Whereas calibration use to determine a relationship between a point appears on image and where it locate in the world coordinate.

According to the pinhole model, the projective between 3D point ($\tilde{Q}$) and its image ($\tilde{q}$) using homogeneous coordinates can be expressed as Equation 17.

$$\tilde{q} = sH\tilde{Q} \qquad (17)$$

Where $\tilde{q} = [x \quad y \quad 1]^T, \tilde{Q} = [X \quad Y \quad Z \quad 1]^T, and\ s$ are an arbitrary scale factor and a 3x4 projection matrix (H) is defined as homography matrix can be represented in Equation 18.

$$H = MW \qquad (18)$$

Where; $M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} and\ W = [R|T]$

The homography matrixes consist of the camera matrix (M) and a physical transformation matrix (W) is represented by rotation matrix (R) and translation vector (T). For converting from object to camera coordinate systems in Figure 21, the rotation matrix (R) is calculated by $R = R_z(\theta)R_y(\varphi)R_x(\psi)$ and the translation vector (T) is simply $T = origin_{object} - origin_{camera}$

Where;

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix}, R_y = \begin{bmatrix} \cos(\varphi) & 0 & -\sin(\varphi) \\ 0 & 1 & 0 \\ \sin(\varphi) & 0 & \cos(\varphi) \end{bmatrix}, R_z = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

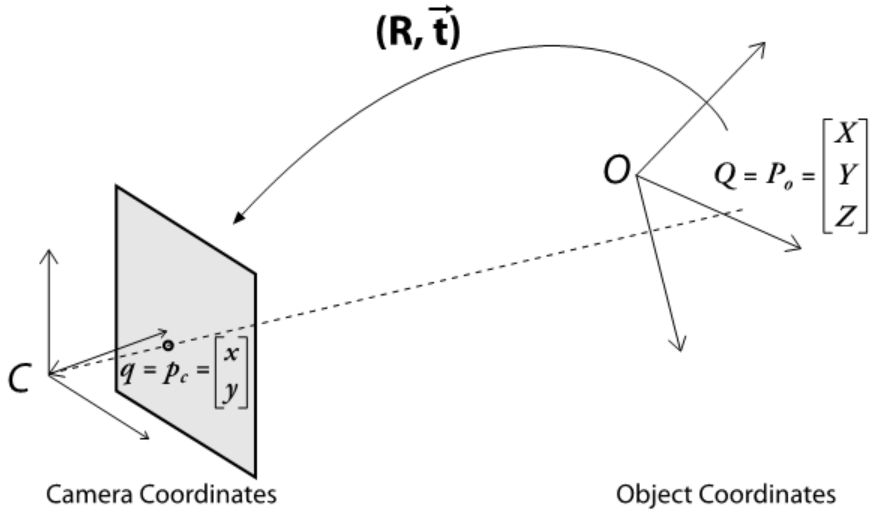


Figure 11 The point P in the object coordinate can be projected into point p on image plane by applying rotation matrix *R* and a translation vector *t* from object to camera coordinate.

Source: Gary and Adrian (2008)

Then, Equation 17 can be rewritten to Equation 19.

$$\tilde{q} = sM[R|T]\tilde{Q} \qquad (19)$$

Or

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

For acquiring the accurate intrinsic and extrinsic parameters, a chessboard (Figure 15) that is known a position of each corner in world coordinate is captured to receive corners position in image coordinate (scale in pixel). In OpenCV, function cvCalibratecamera2 () that input the point of chessboard corners in world and image coordinate returns the camera intrinsic matrix, the distortion coefficients and the 19 extrinsic matrix (rotation vectors and translation vector between camera and object point).

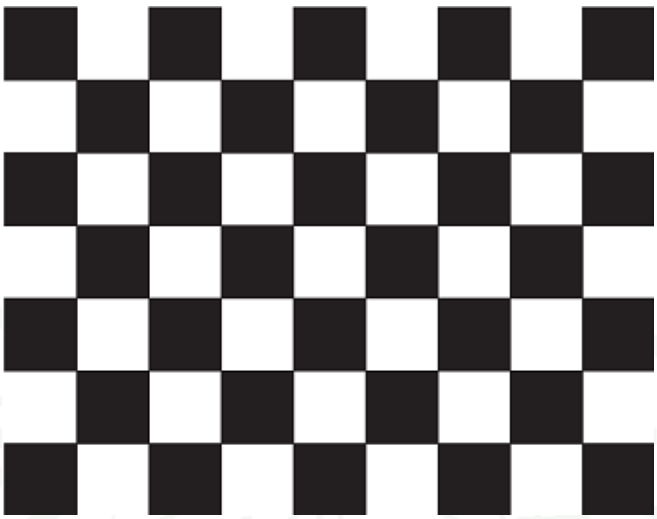


Figure 12 Pattern of chessboard.

3.2.2 Stereo Calibration

In order to compute fundamental and essential matrices, the rotation matrix and translation vector that relate the right camera to the left camera must be acquired. *Stereo calibration* is the process of computing the relation geometry between the two cameras in world coordinate. In Figure 21, the left and right cameras can be separately calibrated and the two views of 3D point are related by $P_l = R^T(P_r - T)$. Where R and T are the rotation matrix and translation vector between the two cameras that are given by $R = R_r(R_l)^T \; and \; T = T_r - RT_l$.

In practice, the function cvStereoCalibrate() in OpenCV that input object points of chessboard corner and points of chessboard corner for both camera views returns the solution for intrinsic matrix*(M2,M2),* distortion Matrix *(D1,D2),* R and T . The results of *R* and *T* are use to estimate the fundamental matrix *F* and the essential matrix *E* in order to rectify the two stereo images in the next section.

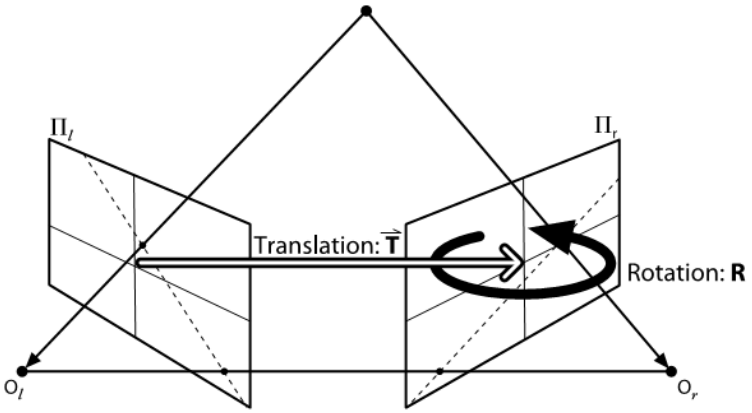


Figure 13 The relation between the two cameras use to find the rotation and translation parameters for projecting 3D point in each camera coordinate.

Source: Gary and Adrian (2008)

3.2.3 Stereo Rectification

To compute stereo disparity, the image rows between two cameras must be aligned. *Stereo rectification* is the process that reprojects image planes of the two cameras to be horizontal parallel and row alignment. So that, the result of the two images rectification contain the principal ray of these two images are parallel and epipoles of each image which are then located at infinity. In OpenCV, there are two algorithms to rectify the two images which are *Hartley's* and *Bouguet's* algorithm.

In this work, we compute rectify images using *Bouguet's* algorithm. Because the disadvantage of *Hartley's* algorithm is the image scale is unknown (in Figure 22) since the intrinsic parameters of both cameras are implicitly contained in the fundamental matrix F.

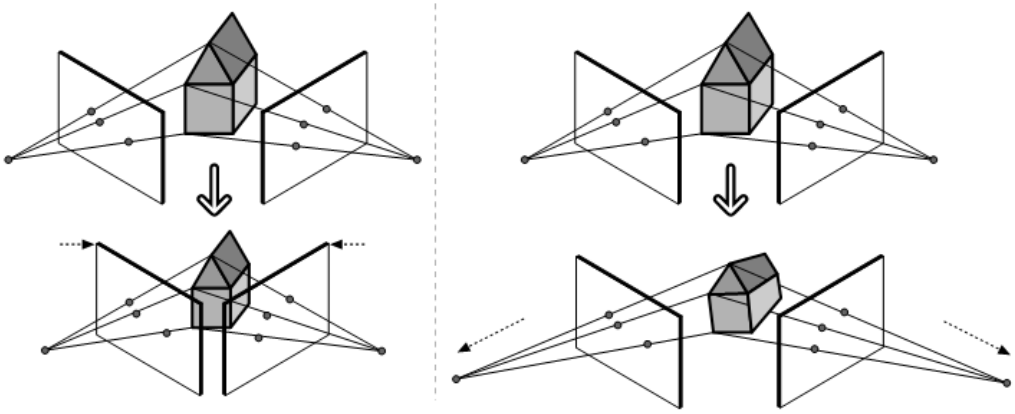


Figure 14 Stereo reconstruction is ambiguity because Hartley's algorithm uses the fundamental matrix which contains the implicit intrinsic camera parameters to compute a matching homography.

Source: Gary and Adrian (2008)

3.2.4 Calibrate Stereo Rectification (Bouget's Algorithm)

To align the epipolar lines horizontally, *Bouget's algorithm* is used to minimize the amount of change reprojection products for each of the two images. Thereby using the two resulting rotation matrixes and for the left and the right camera, each image of the camera can rotates for parallel their principal rays. Nevertheless, the outputs of the two images that rotate of each camera are not row alignment. To map left camera's epipoles to infinity and horizontally align the epipolar lines, matrix R_{rect} is computed as follow.

The unit vector $e_l = \frac{T}{\|T\|}$ which takes the principal point (c_x, c_y) as the left image's origin and T is the translation vector between the two camera's centers of projection is defined.

The unit vector $e_2 = \frac{[-T_y \quad T_x \quad 0]^T}{\sqrt{T_x^2+T_y^2}}$ which is chosen a direction orthogonal to the principal ray and tend to be along the image plane is defined to orthogonal to e_l. Also, it can be acquired using the cross product of e_l with the direction of the principal ray and then normalizing.

The unit vector $e_3 = e_1 \times e_2$ which must be orthogonal to e_1 and $e_2$can be calculated using cross product.

Thus, the matrix R_{rect} which map the left camera's epipoles to infinity is given by

$$R_{rect} = \begin{bmatrix} (e_1)^T \\ (e_2)^T \\ (e_3)^T \end{bmatrix}$$

In order to align the epipolar lines horizontally and map left camera's epipoles to infinity, the row alignment of the two cameras is accomplished by:

$$R_l = R_{rect} r_l$$

$$R_r = R_{rect} r_r$$

Also, the matrices M_{rect_l} and M_{rect_r} which consist of the rectified left and right camera's parameters respectively are used to compute projection matrices $P_l and P_r$ and as follow.

$$P_l = M_{rect\_l} P_l = \begin{bmatrix} f_{x\_l} & \alpha_1 & c_{x\_l} \\ 0 & f_{y\_l} & c_{y\_l} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

And

$$P_r = M_{rect\_r} P_r = \begin{bmatrix} f_{x\_r} & \alpha_r & c_{x\_r} \\ 0 & f_{y\_r} & c_{y\_r} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Where α_l and α_r allow for a pixel skew factor which always set to 0.

To project a 3D point in world coordinate to a 2D point in image coordinate, the projection matrices are represented as follow.

$$P\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Where the point in image coordinate can be computed as (*x/w, y/w*). For reprojecting points in two dimensions into three dimensions, the reprojection matrix is given as follow.

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & \dfrac{(c_x - c'_x)}{T_x} \end{bmatrix}$$

Where c'_x is the principal point *x* coordinate in the right image. If the both principal rays are parallel or intersect at infinity, so that $c_x = c'_x$ and $(c_x - c'_x)/T_x$ in the matrix Q is equal to 0. Then, the 2 dimension points can be project in to the points in three dimensions using the follow equation.

$$Q\begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$

For, it is the disparity that can be calculated as the image rows between two cameras are aligned. And the point in 3D coordinate is given by (*X/W, Y/W, Z/W*).

In OpenCV, Bouget's method is applied using function *cvStereoRectify()* which input the both of camera's intrinsic matrices, distortion vectors, the rotation

matrix and translation vector between the right and left cameras. This function returns and matrices that are row-aligned rectification rotations for the left and right image planes. Moreover, the reprojection matrix and the left and right projection equations ($P_l \; and \; P_r$) are returned from this function.

3.2.5 Rectification Map

To acquire the both of row-aligned images, the process of rectification is shown in Figure 23. Since an actual value of each pixel in rectify image (c) contains the floating-point coordinate pixel value, then it must be interpolated to the nearby integer value. In practice, this process proceeds backward from (c) to (a) for finding the each pixel value of source image in order to fill the rectified integer pixel location in the rectify image (c).

The function *cvIntUndistortRectifyMap()* that inputs the camera matrix *M*, the rectified camera matrix *Mrect*, the rotation matrix *Rrect* and the camera distortion parameters returns lookup maps *mapx* and *mapy* as output. These maps indicate the pixels location of rectify image to interpolate the pixels of source image. Also, this function must be called separately for the left and right cameras. To acquire the rectification of a stereo pair of images, the *cvRemap()* function uses *mapx* and *mapy* repapping parameters of the left and right maps. For matching the stereo correspondence points, the rectification of stereo images is used as the input images. In the next section, the matching 3D point from the two images view is expressed.

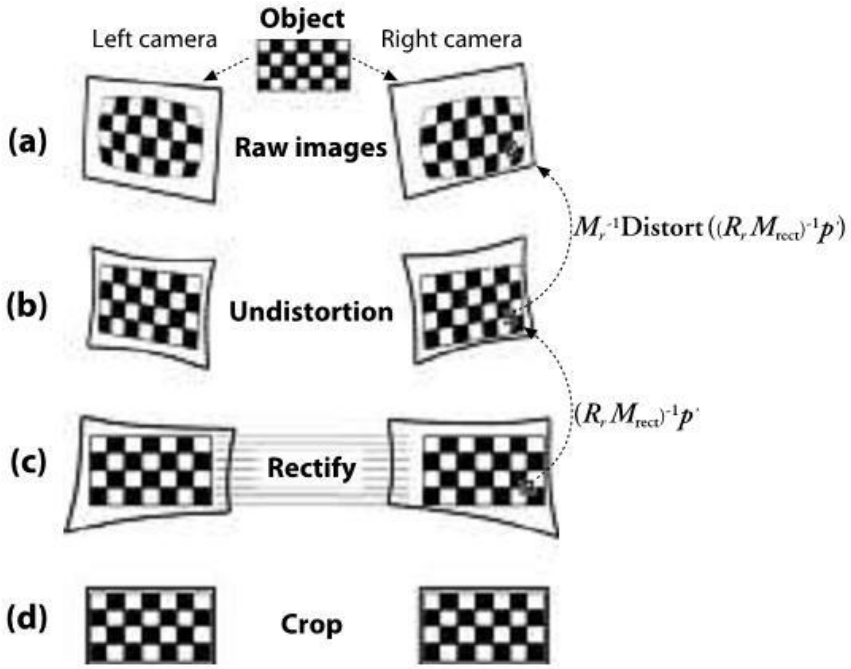


Figure 15 Stereo rectification process.

Source: Gary and Adrian (2008)

3.2.6 Stereo Correspondence

For finding the accurate correspondence point from the stereo images, the both images are arranged to be frontal parallel. Then, the rectify images that receive from the previous section are used as input images in Figure 24. To reduce the computer time and efficiently find the matching point, OpenCV implements a block-matching stereo algorithm via *cvFindStereoCorrespondenceBM()* function. Using "sum of absolute diffence" (SAD) windows, the matching points between the left and right stereo rectified images can be found.

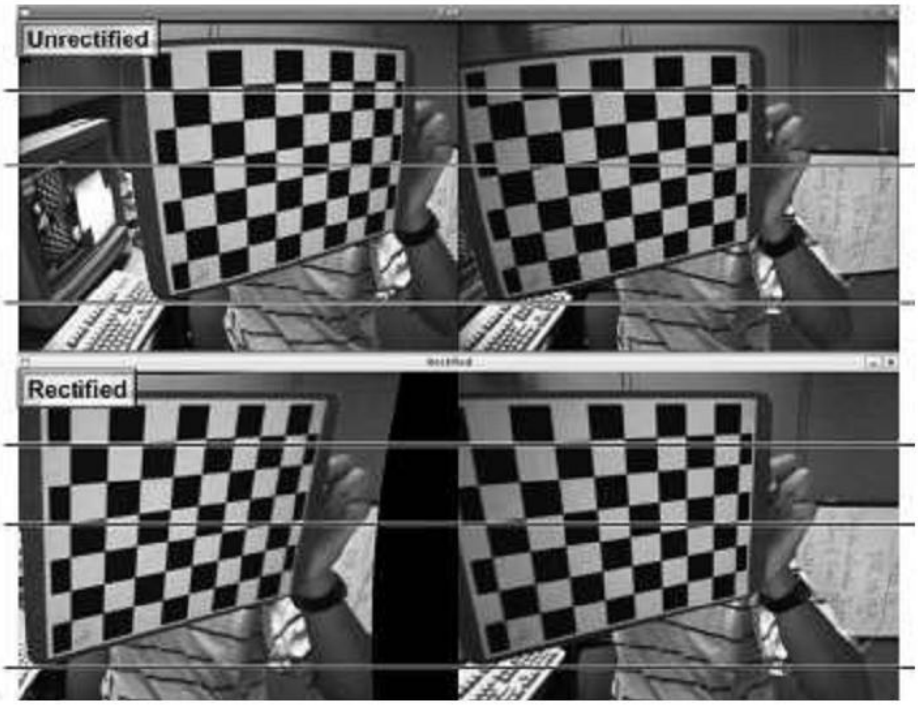


Figure 16 The original left and right image pairs (upper) are rectified to be the lower left and right image pairs. The scan lines in lower rectified image pairs are aligned and the barrel distortion has been corrected.

Source: Gary and Adrian (2008)

There are three main steps of the block-matching stereo correspondence algorithm that proceeds with undistorted and rectified stereo images:

The prefiltering step is used to normalize image brightness and enhance texture.

Using as SAD window, correspondence can be searched along horizontal epipolar lines.

The postfiltering step is used to eliminate fault correspondence matches.

In the prefiltering step, the input rectify images are normalized to reduce lighting differences and to enhance image texture between left and right images. Correspondence point is computed by a sliding SAD window that is *SADWindowSize* parameter and the size of window can be extended from 5x5,7x7, to 21x21(the maximum) over the image. The value of each center pixel in the input image that is under the window is replaced by $min\left[max\left(I_c - \bar{I}, -I_{cap}\right), I_{cap}\right]$. Where $\bar{I}$ is the average value in the window and I_{cap} is a positive numeric limit which is *preFilterCap* parameter and default value is 30. Also, a flag in this method can be chosen from CV_NORMALIZED_RESPONSE which is set as a default flag or CV_LAPLACIAN_OF_GAUSSIAN which runs a peak detector over of the image.

After a rectification and prefiltering step, a sliding SAD window is used to search a best match of each feature in the left image from the corresponding row in the right image. The matching location in the right image must be along the same y-coordinate or the same row which is an epipolar line as in the left image. It is possible to find this matching location (in Figure 17) if the feature is not occluded in the right image view and it has enough texture to be detectable. The right pixel match point at (x_0, y_0) for a horizontal frontal parallel camera arrangement must be found in the same row of a left image feature at x_0. Also, the zero disparity is started at x_0 and larger disparities are to the left (in Figure 18). However, negative disparities may be occurred (to the right of x_0, y_0) to the cameras which are angled toward each other. The first parameter of function *cvFindStereoCorrespondenceBM()* is *minDisparity* which is the location to start the matching search and the default for it is 0. The disparity search is implemented over *numberOfDisparities* that counted in pixels (the default parameters is 64 pixels). For disparities of discrete distributions, each pixel is set to contain 16 subdisparities (the subpixel resolution). To reduce the computation time, decreasing the number of disparities to be searched by limiting the length along the epipolar line for a matching point.

After correspondence step, the postfiltering process is used to prevent false matching points. A *uniqurnessRatio* parameter can be filters out false matches by 35 *uniquenessRatio > (match_val-min_match)/min_match*. To eliminate random noises during matching using textureThreshold parameters, if the response on the SAD

window is less than this parameter which default value is 12. Eventually, the problem of matching is occurred near the boundaries of objects. These results near the boundary are small and large disparities that are defined *speckle*. Using a speckle window, a *speckleWindowSize* parameter can be set from 5 (5-by-5) up to 21 (21-by-21) which default value is 9. The minimum and maximum disparities are not eliminated, if these values are within speckleRange (default value is 4).

To set all of these parameters, a function *cvStereoBMState()* is used to compute the accurate correspondence point. However, there is another function that is can be called to compute the correspondence point. That is the *cv::StereoSGBM()* which use all parameters of *cvSteroBMState()* and improve to get the fine disparity by adding some parameters. For controlling disparity smoothness, parameter P_1 is set to *8*SADWindowSize*SADWindowsize* and parameter P_2 is set to *32*SADWindowSize*SADWindowSize*. To allow the maximum difference in the left-right disparity check, parameter disp*12MaxDiff* must set to non-positive value. The last parameter is *fullDP* that set to true for running full-scale 2-pass dynamic programming algorithm. Finally, a result of disparity map can be obtained by using function *cvFindStereoCorrespondenceBM()* which inputs left and right rectified images and function *cvStereoBMState()* for computing precision disparity. Moreover, *cv::StereoSGBM::operator()* that inputs rectified image pairs and function *cv::StereoSGBM()* for computing the fine disparity.

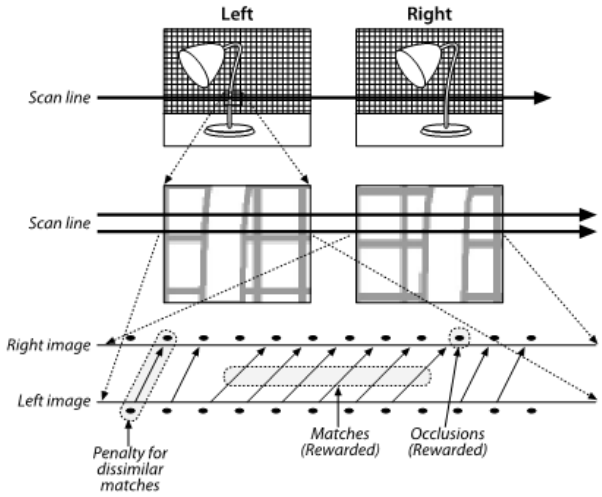


Figure 17 The upper panel shows the left and right images of a lamp. The middle panel displays an enlargement of a single scan line. Finally, the lower panel expresses visualization of the correspondence.

Source: Gary and Adrian (2008)

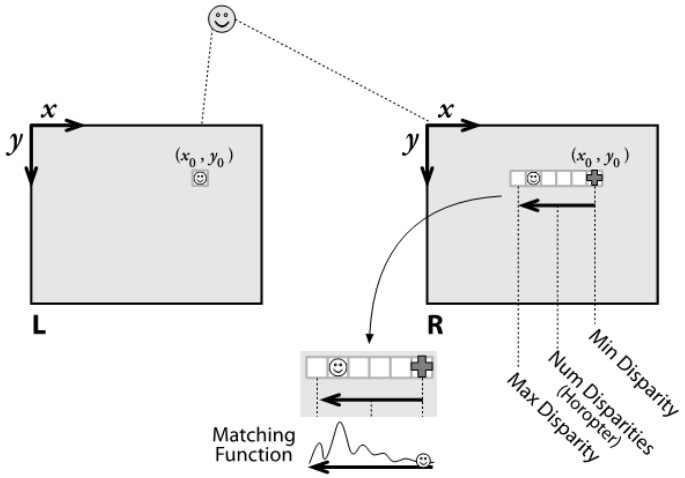


Figure 18 A feature of left image must occurs the same row and at the same coordinate point to search the match point of right image. The lower part of the figure shows the characteristic matching function of window-based feature matching.

Source: Gary and Adrian (2008)

3.2.7 Reprojection of 3D

Using disparity map from the previous topic, the 3D reconstruction or depth map can be obtained from function *cvReprojectImageTo3D()*. Given a 2D point the and disparity, the corresponding 3D point (*X/W,Y/W,Z/W*) can be derived using the roprojection matrix to compute in equation (16). Then, the results of *cvReprojectImageTo3D()* is a three-channel floating-point.

3.3 Kinect Calibration

3.3.1 Calibration of RGB/Depth Camera

In this research, we assume that lens distortion is so small and negligible. The method and process for camera calibration in the section 2.2 is utilized to calibrate the RGB and depth camera of Kinect. For RGB camera, the chessboard is captured into color image to detect its corners (in Figure 20a). To estimate the intrinsic parameters of the RGB camera, the points of chessboard corners in world and color image coordinate are input into *cvCalibratecamera2()* function of OpenCV.

For depth camera, we test to calibrate of the depth image fisrt. We added input of the depth image to estimate intrinsic paramers. The same chessboard is captured into depth image to detect its corner (in Figure 20b). It is necessary to fix the four corners of the chessboard in depth image (in Figure 19). Then, the points of chessboard corners in world and depth image coordinate are input into *cvCalibratecamera2 ()* to calculate the intrinsic parameters of the depth camera. It is not perfectly when we manually configure the four corners of chessboard as shown Figure 19. Thus, the IR images are captured and are used to calibrate function and stereo function.

For mapping between RGB and depth images, we must use intrinsic and extrinsic parameters of the two cameras. In order to do so, we also calibrated the extrinsic parameters between the RGB and depth cameras. A known object, chessboard, is captured into RGB and depth images simultaneously. Then, corners of

the chessboard in the RGB and IR images are detected to estimate the intrinsic and extrinsic parameters using *cvStereoCalibrate()* function with OpenCV. The experimental results are shown in Figure 21 and the result section.

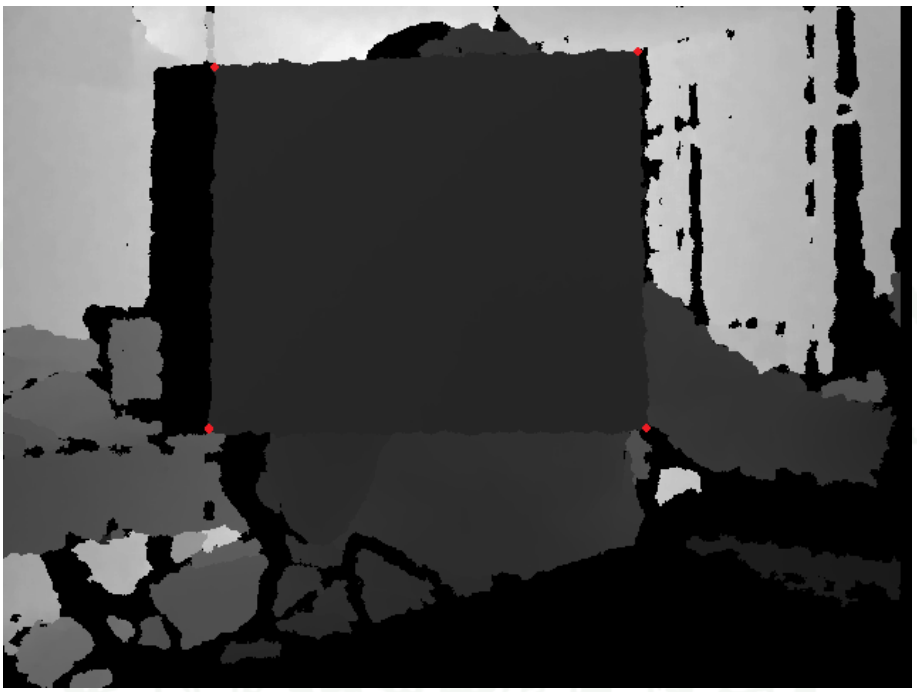
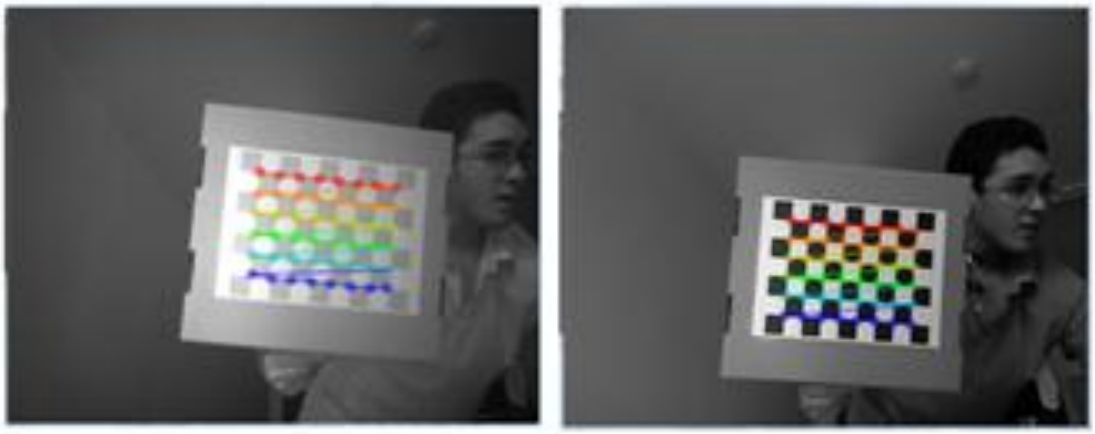


Figure 19 The depth image consists of a chessboard with manually detected corners shown as red points.

(a) Color image. (b) Depth image.

Figure 20 The output images from RGB and depth cameras from Kinect.



(a) IR image (b) RGB image

Figure 21 Show the result of the experimental image.

3.3.2 Calibration of Depth Sensor

Since, Kinect's depth camera returns the raw depth data. For acquiring real depth data, depth calibration is applied using line regression technique. Then, raw depth data must be transformed into the depth data (in centimeter) using equation (24) and (25):

$$Depth = \frac{1.0}{f(d)} \quad (24)$$

$$f(d) = -0.00307110156374373d + 3.33094951605675 \quad (25)$$

Where; d is the depth data which measure from depth sensor. $f(d)$ is a linear function which can be estimated from regression techniques, Ruchanurucks *et al.* (2008).

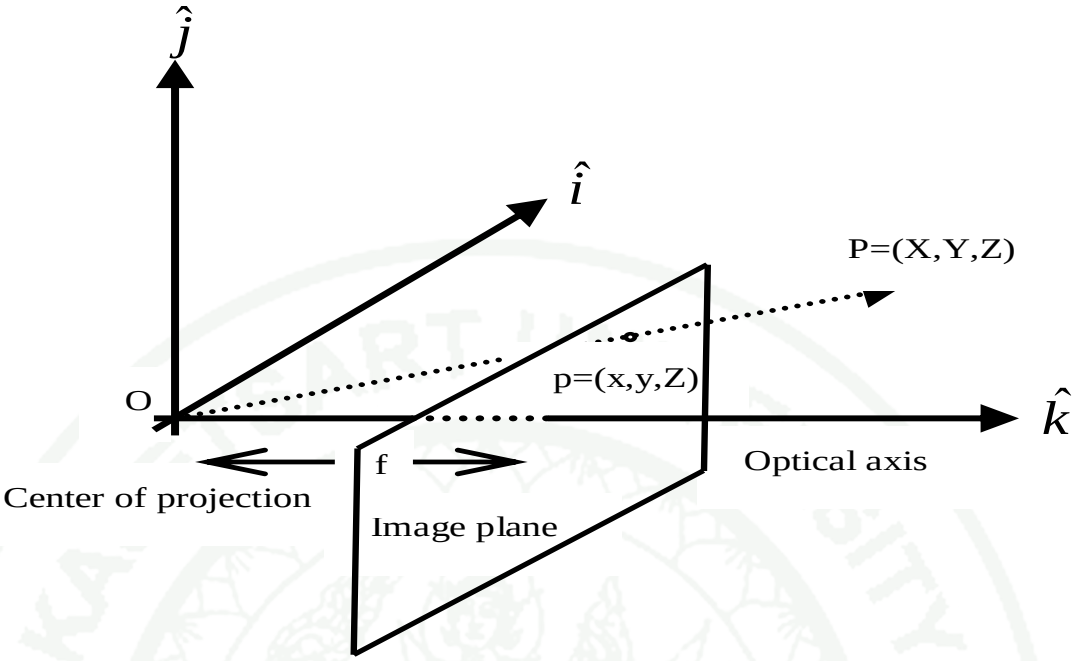


Figure 22 Displays the image coordinates (xyZ) are projected into the world coordinates (XYZ).

Using the intrinsic parameters of depth camera, the 3D point (X, Y, Z) can be approximated from the depth image (x, y, Z) in equation (26) and (27).

$$X = \frac{Z(x-c_x)}{f_x} \quad (26)$$

$$Y = \frac{Z(y-c_y)}{f_y} \quad (27)$$

$$Z = depth \quad (28)$$

Where;

x , y are pixel of depth image.

Z is a distance between object and Kinect.

f_x, f_y, c_x, c_y are the elements of intrinsic parameter[M] of depth camera.

3.3.3 Mapping between RGB and Depth Image

Using the extrinsic parameters [R|T], the equation (29) is used to project each 3D point (XYZ) in depth camera coordinate onto the RGB camera coordinate:

$$P3D' = R * P3D + T \qquad (29)$$

$P3D$ is a matrix consists of X, Y, Z in depth camera coordinate. The rotation and translation matrices are transferred in RGB camera coordinate ($P3D'$) as shown Figure 23.

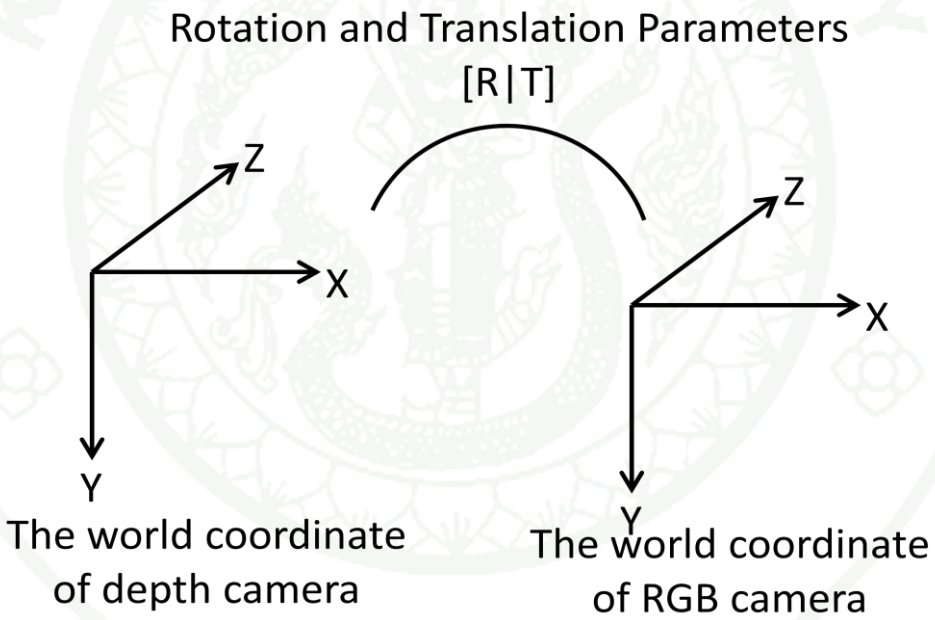


Figure 23 Transfering the depth coordinate into the RGB coordinate.

After that, the intrinsic parameters of RGB camera are processed with 3D point on the color image. We can then map the 3D depth images onto 2D color images using equation (30) and (31):

$$P2D.x = (\frac{P3D'.x * f_x}{P3D'.z}) + c_x \quad (30)$$

$$P2D.y = \left(\frac{P3D'.y * f_y}{P3D'.z}\right) + c_y \quad (31)$$

Where, f_x, f_y, c_x, c_y are the elements of intrinsic parameter [M] of RGB camera. The $P2D$ is a position of 2D color image as shown Figure3a. Do not confuse the mapped image with the RGB image. The mapped image consists of RGB plus depth information, shown in 2D coordinate. In other words, we can also show the mapped image in 3D coordinate as well, as shown in Figure3b, using OpenGL.

3.4 The Tri-States Segmentation

To achieve our goal of burn area proportion estimation, we must divide the depth information into 3 areas, namely, the burn area, the body area and the background area.

Before burn segmentation cases, we design to threshold the area of sample objects. It also uses the same proceder that consists of sub-sample area, sample area, and environment area to compare accuracy of the surface area resulting with known ground truth object in Figure 35. Luckily we already mapped the RGB image and depth image. Hence, the segmentation can be done based on color and edge information in RGB image. Then the result can be reflected onto the same area in the depth image. The segmentation program is based on a locally segmentation method called watershed algorithm and a statistical threshold called Chebyshev's inequality. The detail of algorithm can be found from, Khongma (2012)

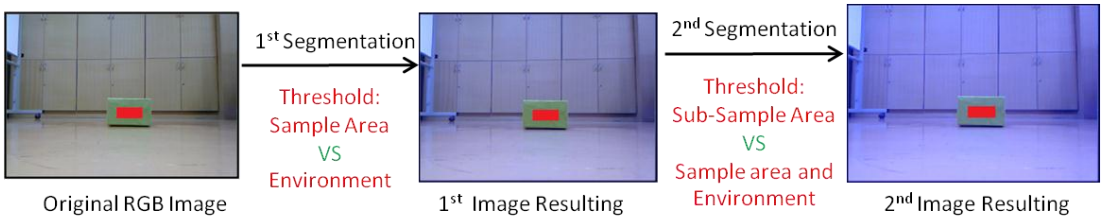


Figure 24 The image resulting of segmentation program. There are 3-states segmentation: Sub-Sample Area, Sample Area, and Environment.

3.5 Triangular Mesh Reconstruction

3.5.1 Average filter

Average filter normalizes noisy surface into an enhanced one.

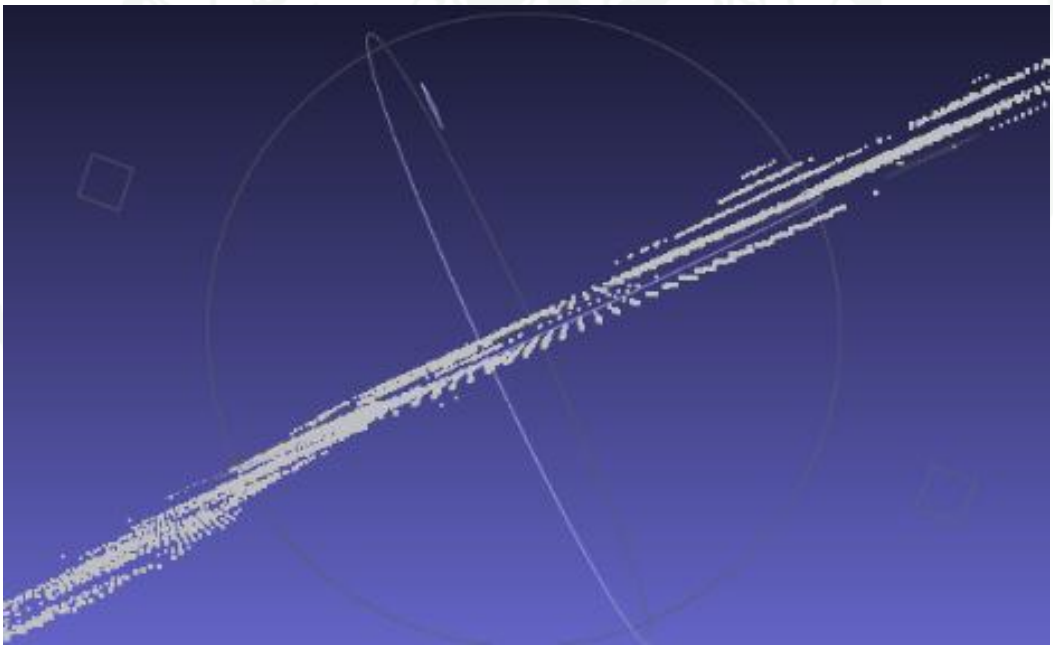


Figure 25 A set of point cloud of the planar object that is shown in top view.

Figure 25 shows a point cloud of a planar surface. One may notice that it is not in the same plane, even it should be so. This is due to the fact that Kinect's resolution

is coarse. So we normalize it using the average filter only for depth information (Z axis) of 3D-world coordinate.

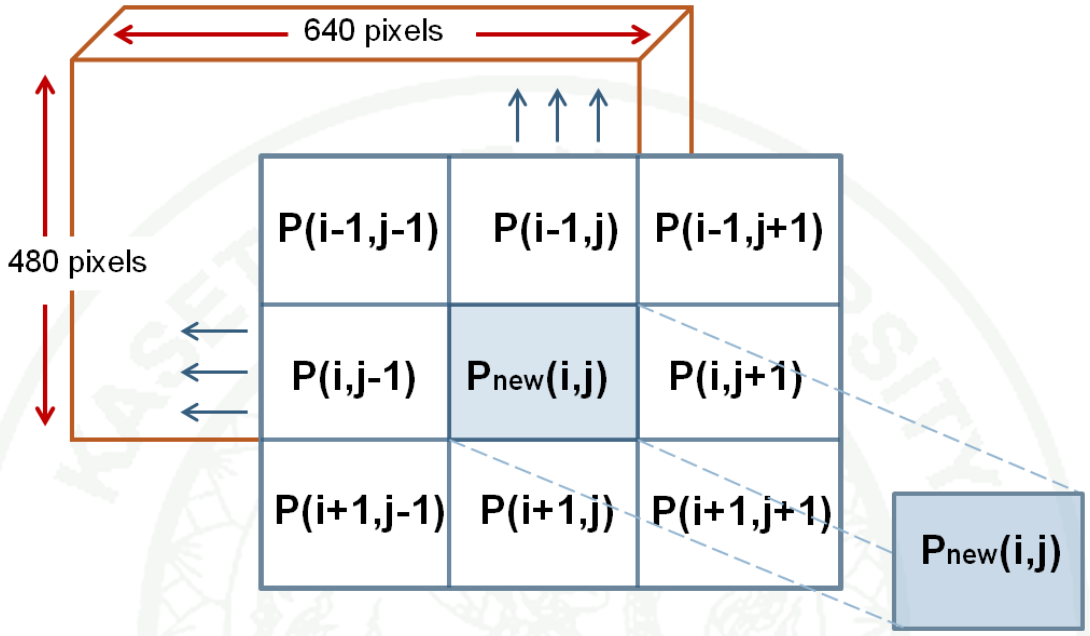


Figure 26 Searching windows size used 3 × 3 of the average filter using mean equation.

$$\overline{Z} = \frac{1}{n}\sum_{i=1}^{n} z_i \qquad (32)$$

Where, z_i is a raw depth data of each point cloud. $\overline{Z}$ is a normalize depth data of average values. n is a number of average set.

3.5.2 Marching cubes algorithm

For point cloud to 3D triangular mesh reconstruction, there are many methods, e.g. intrinsic property driven (IPD) method, Cline and Lorensen (1987). We will focus on marching cubes as it is a well-known technique and, currently, many variants of it are still being developed.

Marching cubes algorithm is one of the latest algorithms of surface construction used for viewing 3D data. In 1987 Lorensen and Cline (GE) described the marching cubes algorithm. It has been published at the SIGGRAPH convention. This algorithm produces a triangle mesh by computing iso-surfaces from discrete data. By connecting the patches from all cubes on the iso-surface boundary, we get a surface representation. It is an algorithm for rendering isosurfaces in volumetric data. The basic notion is that we can define a voxel (cube) by the pixel values at the eight corners of the cube. If one or more pixels of a cube have values less than the user-specified isovalue, and one or more have values greater than this value, we know the voxel must contribute some component of the isosurface. By determining which edges of the cube are intersected by the isosurface, we can create triangular patches which divide the cube between regions within the isosurface and regions outside. By connecting the patches from all cubes on the isosurface boundary, we get a surface representation. This algorithm is often used to extract the surface of medical organs. It provides a fast and easy way to get from serial sections to a complete 3D object.

In order to explain this technique, we are going to indroduce the marching squares algorithm which uses the same approach in 2D. After having presented some marching squares ambiguous cases we will be able to describe the marching cubes algorithm and an ambiguous case resolution.

1) The marching squares algorithm

The marching squares algorithm aims to draw lines between interpolated values along the edges of a square, considering given weights of the corners and a reference value. You can consider a 2D grid as shown in Figure.

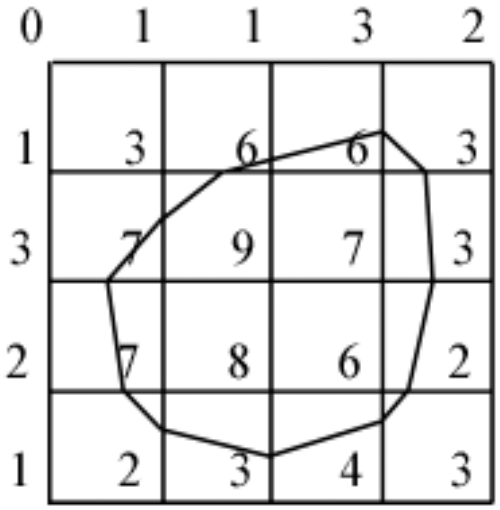


Figure 27 The marching squares algorithm.

Each point of this grid has a weight and here the reference value is known as 5. To draw the curve whose value is constant and equals the reference one, different kinds of interpolation can be used. The most used is the linear interpolation.

In order to display this curve, different methods can be used. One of them consists in considering individually each square of the grid. This is the marching square method. For this method 16 configurations have been enumerated, which allows the representation of all kinds of lines in 2D space.

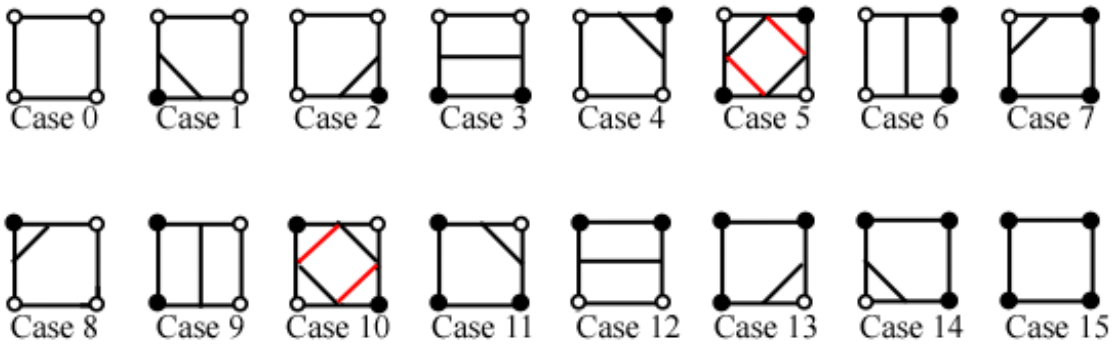


Figure 28 16th configurations of the marching squares algorithm.

The marching squares ambiguous cases, while trying this algorithm on different configurations we realise that some cases may be ambiguous. That is the situation for the squares 5 and 10 as Figure. We are not able to take a decision on the interpretation of this kind of situation. However, these exceptions do not imply any real error because the edges keep closed.

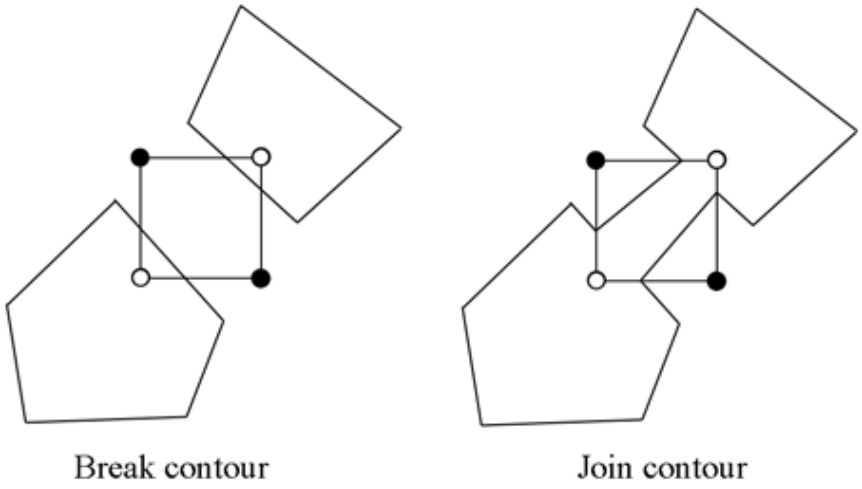


Figure 29 Break and join contour of the marching squares algorithm.

2) The marching cubes algorithm

The marching cubes algorithm is a high resolution 3D surface construction algorithm, Zlatka (2009). This algorithm generates a triangular mesh estimates iso-surface. It calculates the normal vector of the surface at each vertex of the triangle.

Following the marching squares algorithm we can adapt our approach to the 3D case: this is the marching cubes algorithm. The principle of marching cubes algorithm locates the surface of point cloud in a cube of eight vertexes. The surface intersects edge of this cube where one vertex is outside and the other inside the surface. In a 3D space we enumerate 256 different situations for the marching cubes representation. All these cases can be generalized in 15 patterns by rotations and symetries:

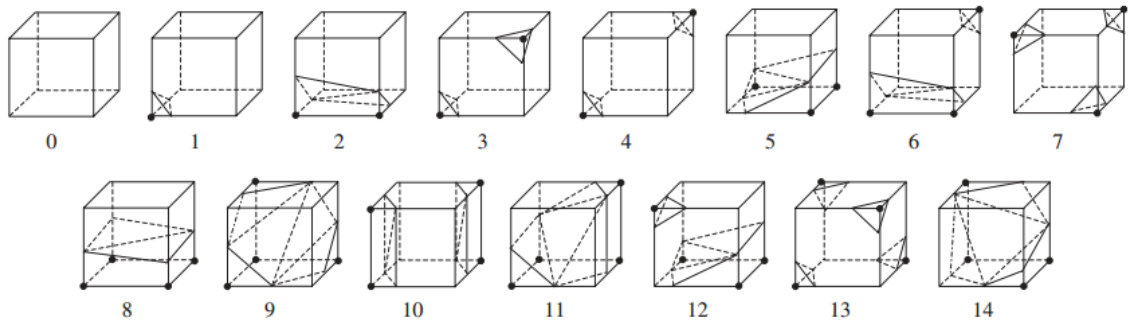


Figure 30 The marching cube algorithm.

In order to be able to determine each real case, a notation has been adopted. It aims to refer each case by an index created from a binary interpretation of the corner weights.

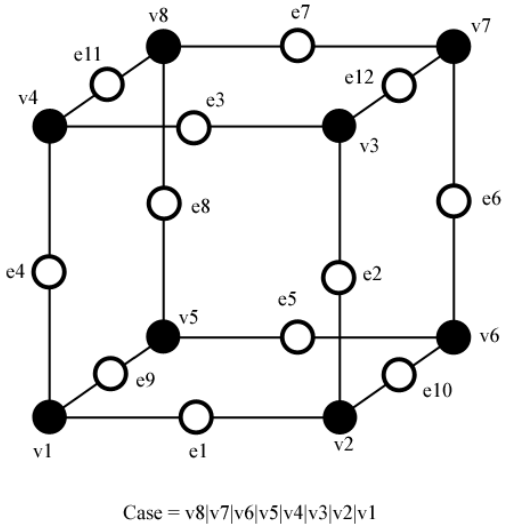


Figure 31 The corner weights of the marching cube algorithm.

In this way, vertexes from 1 to 8 are weighted from 1 to 128 (v1 = 1, v2 = 2, v3 = 4, etc.); for example, the family case 3 example you can see in the picture above, corresponds to the number 5 (v1 and v3 are positive, 1 + 4 = 5).

After that, it calculates normal vectors and marches to the next cube. To all this cubes we can attribute a complementary one. Building a complementary cube consists in reversing normals of the original cube.

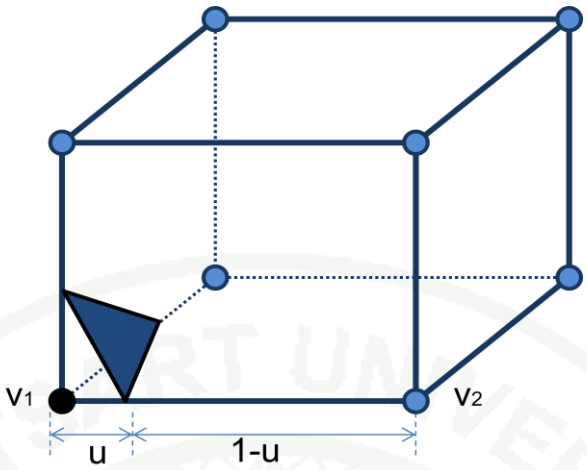


Figure 32 The cube of eight vertexes is represented with surface intersection by triangular mesh.

Interpolate surface intersection along each edge:

$$v_i = v_i * (1 - u) + v_2 \quad (33)$$

$$u = \frac{(v_1 - v_i)}{(v_1 - v_2)} \quad (34)$$

Calculate normal for each cube vertex:

$$G_x(i,j,k) = \frac{(D(i+1,j,k) - D(i-1,j,k))}{\Delta x} \quad (35)$$

$$G_y(i,j,k) = \frac{(D(i,j+1,k) - D(i.j-1,k))}{\Delta y} \quad (36)$$

$$G_z(i,j,k) = \frac{(D(i,j,k+1) - D(i,j,k-1))}{\Delta z} \quad (37)$$

The central differences with the three coordinate axes are used to estimate the gradient at cube vertex (i, j, k).

Interpolate the normals at the vertices of the triangles:

$$\vec{n_1} = u\overrightarrow{g_2} + (1-u)(\vec{g_1}) \qquad (38)$$

In the next picture you can see some instances of cubes with their normals.

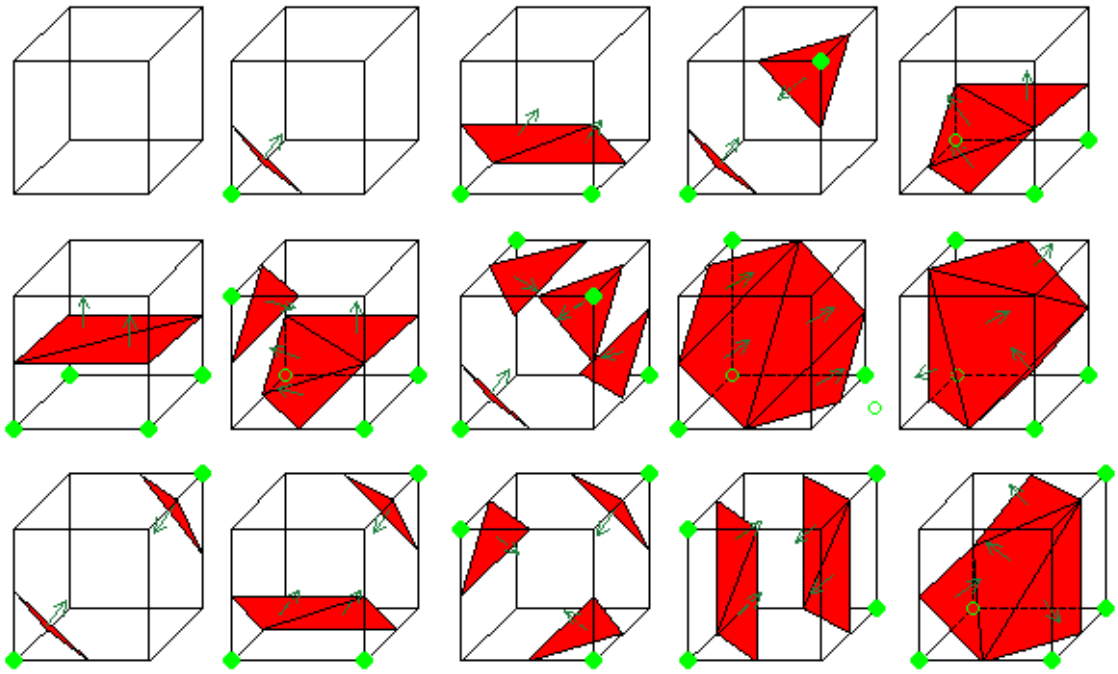


Figure 33 The instances of cubes with their normal.

The creation of these complementary cubes allows to give an orientation to the surface.

3.5.3 Area Calculation

From the triangular mesh, we use the Heron's formula to calculate each triangle's area, Lenz *et al.* (2011).

Heron's formula is a method for calculating the area of a triangle when you know the lengths of all three sides. The formula is credited to Heron (or Hero) of Alexandria, and a proof can be found in his book, Metrica, written c. A.D. 60.

The area T of a triangle whose sides have lengths a, b, and c is

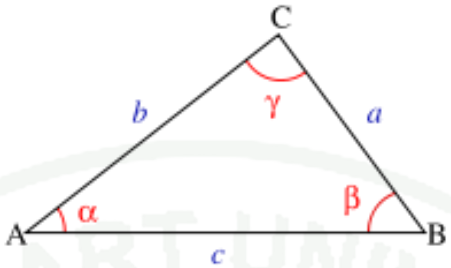


Figure 34 A triangle with sides a, b, and c.

$$T = \sqrt{(s(s-a)(s-b)(s-c))} \qquad (39)$$

$$s = \frac{(a+b+c)}{2} \qquad (40)$$

Where;

T is a triangular area of each triangular mesh.

s is the semiperimeter of the triangle.

a,b,c are lengths of each sides.

RESULTS AND DISCUSSION

Results

1. Pre-processing step

1.1 Cameras Calibration from Chessboard Pattern

Calibration of stereo cameras using OpenCV can find intrinsic matrices, distortion matrices of depth and color cameras. The depth camera means as IR camera. This program which can uses either .bmp or .jpg images will be identifying the corners on a chessboard pattern. cvStereoCalibrate() function estimates the calibration parameters. It can extract 48 corner points as the shown in Figure 41 (width=8 and height=6). After that, remove len's distortion and rectify images using distortion parameters and Bouget's algorithm respectively.

Using program, outputs of stereo cameras calibration are saved in YAML (.yml) formats. In this thesis, we calibrate offline and save it data. After that, read the saved file to acquire all of parameters. Here, the results of cvStereoCalibrate() function are expressed as follow.

For intrinsic parameters of the depth and the color cameras:

```
%YAML:1.0
M1: !!opencv-matrix          rows: 3          cols: 3          dt: d
  data: [ 5.7292463178507114e+002, 0., 2.2411772367052211e+002, 0.,
      5.7292463178507114e+002, 2.0865327223984048e+002, 0., 0., 1. ]
D1: !!opencv-matrix          rows: 1          cols: 5          dt: d
  data: [ 0., 0., 1.1232442132870080e-001, -2.4234620748647059e-003, 0. ]
```

M2: !!opencv-matrix rows: 3 cols: 3 dt: d

data: [5.7292463178507114e+002, 0., 3.5963548025891407e+002, 0.,
5.7292463178507114e+002, 3.3982547538584805e+002, 0., 0., 1.]

D2: !!opencv-matrix rows: 1 cols: 5 dt: d

data: [0., 0., -4.3120815329761961e-002, 1.9400079857200880e-002, 0.]

It means intrinsic and distortion parameters are in Table 1

Table 1 Experimental Intrinsic Parameters of both Cameras.

Parameters	**Values**
Intrinsic parameters of depth camera (M1 Matrix)	f_x =5.7292, f_y =5.7292 c_x =2.2412, c_y =2.0865
Distortion parameters of depth camera (D1 Matrix)	k_1 =0, k_2 =0, k_3 = 0.1123 p_1 =-0.0024, p_2 =0
Intrinsic parameters of the color camera (M2 Matrix)	f_x =5.7292, f_y =5.7292 c_x =3.5964, c_y =3.3983
Distortion parameters of color camera (D2 Matrix)	k_1 =0, k_2 =0, k_3 = -0.0431 p_1 =0.0194, p_2 =0

For extrinsic parameters of both cameras:

%YAML:1.0

R: !!opencv-matrix rows: 3 cols: 3 dt: d

data: [9.6950053903822131e-001, -5.9489462462601339e-002,
-2.3775977090439221e-001, 6.3657304101372783e-002,
9.9792287495097953e-001, 9.8834854235215330e-003,
2.3667795089349508e-001, -2.4717190485214140e-002,
9.7127370398635526e-001]

T: !!opencv-matrix rows: 3 cols: 1 dt: d

data: [-1.6513336420120980e-002, -8.6837368483585572e-002,
3.3825264950098716e-002]

E: !!opencv-matrix rows: 3 cols: 3 dt: d

data: [-2.2705715610916343e-002, -3.1608629866937597e-002,

-8.4677164044561201e-002, 3.6701955228560519e-002,
-2.4204101113789967e-003, 7.9966821846278475e-003,
8.3137681075272959e-002, -2.1644944528155189e-002,
-2.0809642156399567e-002]

F: !!opencv-matrix rows: 3 cols: 3 dt: d

data: [3.7453211558210249e-005, 5.2138621030839202e-005,
6.0750700854855968e-002, -6.0540091196877388e-005,
3.9924807265499921e-006, 5.1778583172435878e-003,
-7.1465084801595039e-002, 3.4774895397202100e-004, 1.]

It means extrinsic parameters between both cameras are in Table 2

Table 2 Experimental Extrinsic Parameters between both Cameras.

Parameters	**Values**
Rotation parameters (R Matrix)	R_{11} = 9.6950, R_{12} = -5.9489, R_{13} = -2.3775 R_{21} = 6.3657, R_{22} = 9.9792, R_{23} = 9.8834 R_{31} = 2.3667, R_{32} = -2.4717, R_{33} = 9.7127
Translation parameters (T Matrix)	T_{11} = -1.6513 T_{12} = -8.6837 T_{13} = 3.3825

1.2 Depth Sensor Calibration

In methods section, shows 2 equations to interpolate the depth values between the raw depth data (11 bits) to depth distance (meter scales). Note that Kinect returns the raw depth data which contains integer value between 0 to 2047. It must be transformed into the depth image using the following equations. And we found the depth distances are negative value at more than 1080 of the raw depth data. It means we should use around 0 up to 1080 of the raw depth data.

$$Depth = \frac{1.0}{f(d)} \qquad (24)$$

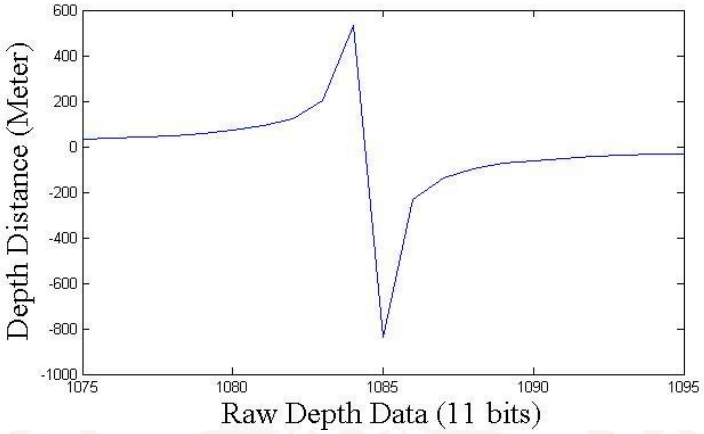


Figure 35 This graph is shown relation between raw depth data (11 bits) and depth distance (meter scales) using above equations.

However, the depth distances from the raw data depend on the standard deviations. The calculated standard deviations plotted with the depth distance from the plane to the depth sensor, Ruchanurucks (2008). The random errors can be increase from a few millimeters at 0.5 m distance up to a few centimeters at the faraway range of the depth sensor. In fact, the random errors increase with the square distance from the depth sensor as follows Figure (above-mentioned):

$$F(Z) = 1.5 * 10^{-5} * Z^2 \qquad (41)$$

F(Z) is the error value of each point in the same plane.

Table 3 Show the error values in centimeter scales.

The depth distance in meter scales	The error value in centimeter scales
0.50	0.0375
1	0.15
2	0.6
3	1.35
4	2.40
5	3.75

(a) (b) (c)

Figure 36 Testing the distance of depth camera (a) IR image is captured other camera (b) Depth image less than 50 centimeters (c) Depth image more than 50 centimeters.

2. Processing

Fisrt experiment, we design to test sample plannar objects which must be known area as shown Table 1. Then we apply to test the burn surface area per a view. It is captured by depth and color cameras in the same time.

2.1 The tri-state segmentation

To efficiently segmentation program, we consider color image only. Thus, the color image is divided using Watershed Algorithm and Chebyshev's Inequality into 3 areas, there are 2 steps. First, segment between the skin and the background. Second, further segment the skin into normal skin and the burn areas. Now we are ready to prepare image for reconstruction.

(a) (b)

Figure 37 The size of 1st sample object is 36.5 cm x 11 cm = 401.5 cm^2 of front view.

(a) (b)

Figure 38 The size of 2nd sample object is 38.5 cm x 16.5 cm = 635.25 cm^2 of front view.

(a) (b)

Figure 39 The size of 3rd sample object is 38.5 cm x 11.5 cm = 442.75 cm^2 of front view.

(a) (b)

Figure 40 The size of 4th sample object is 27 cm x 20.5 cm = 553.5 cm^2 of front view.

(a) (b)

Figure 41 The size of 5th sample object is 27 cm x 6.5 cm = 175.5 cm^2 of front view.

(a) (b)

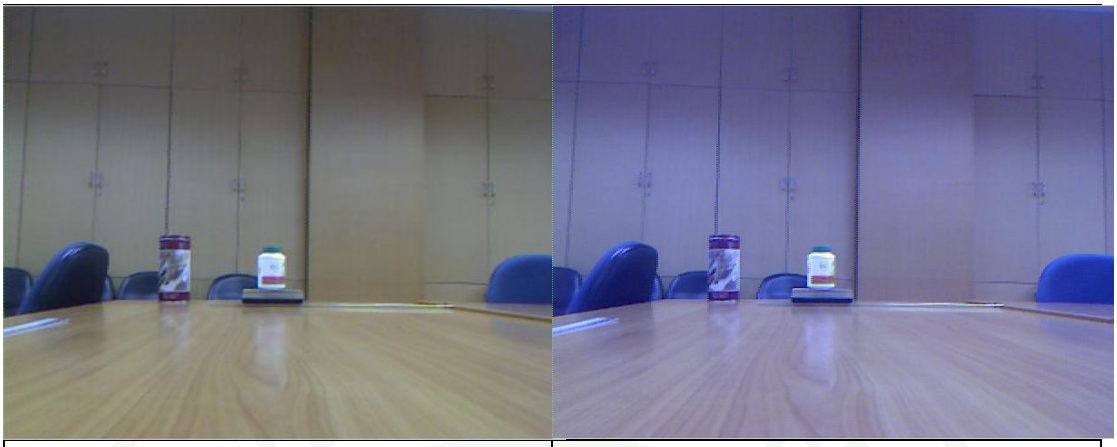
Figure 42 The size of 6th sample object is 18.5 cm x 21 cm = 388.5 cm^2 of front view.

(a) (b)

Figure 43 The size of 7th sample object is 5.5 cm x 21 cm = 115.5 cm^2 of front view.

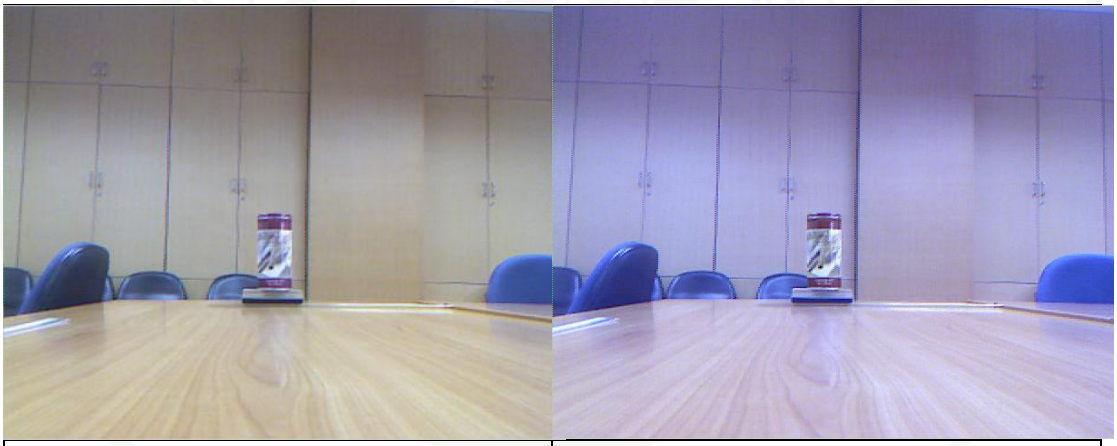
(a) (b)

Figure 44 The size of 8th sample object is 29.5 cm x 4.5 cm = 132.75 cm^2 of front view.



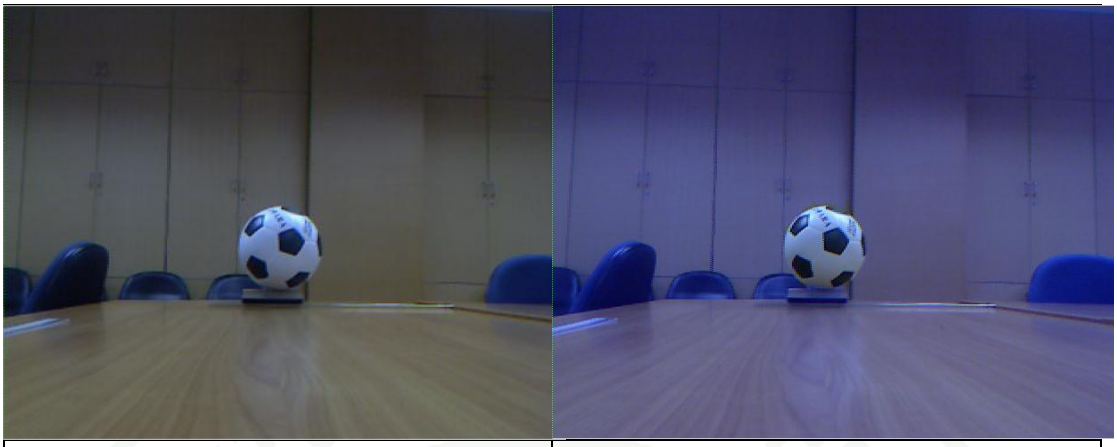
(a) (b)

Figure 45 The size of 9th sample object is 113.4 cm^2 of front view.



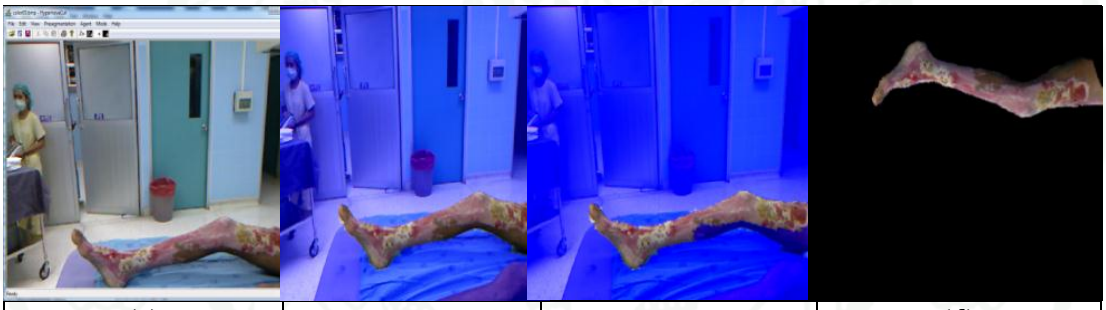
(a) (b)

Figure 46 The size of 10th sample object is 282.6 cm^2 of front view.



(a) (b)

Figure 47 The size of 11th sample object is 3215.36 cm^2 of front view.



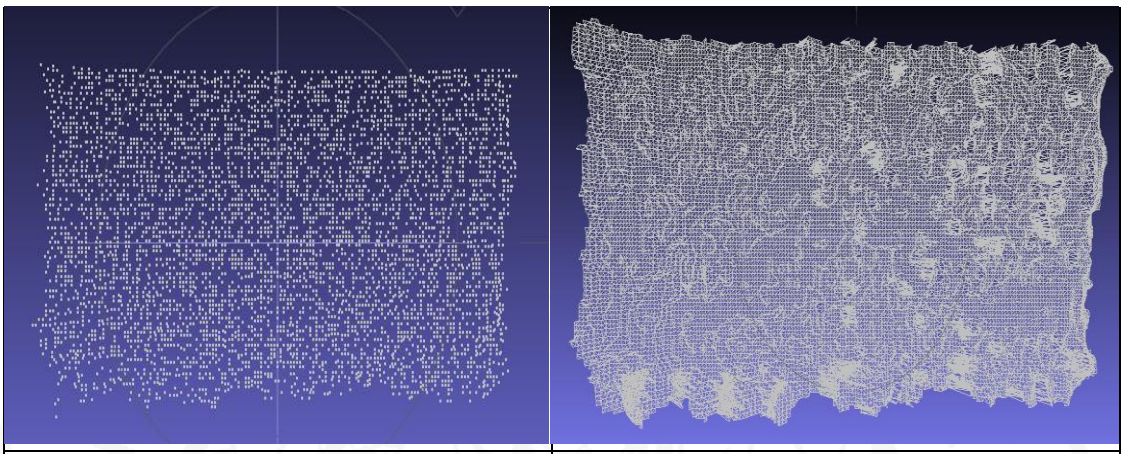
(a) (b) (c) (d)

Figure 48 The result images of segmentation program (a) Original image (b) The skin area is separated between skin area of patient and environment. (c) The burn area is separated between burn area and skin area. (d) Prepare image for mapping program

3. Post-Processing

Experiment is divided into two parts. First is the average filter's evaluation. Second is burn area estimation. First, we collect point cloud of planar objects and compare the resulting area of our method with and without average filter. We calculate the resulting area of sample rectangular between the ground truth and our

programs. In this experiment, a set of point cloud is captured in range between 0.5 m to 1m as shown in Figure 6. The result is shown in this table, it is clear that average filter enhances the correctness of surfaces dramatically.



(a) (b)

Figure 49 (a) Point cloud of sample object is normalized using average filter. (b) Triangular mesh area of planar object is generated using marching cubes algorithm

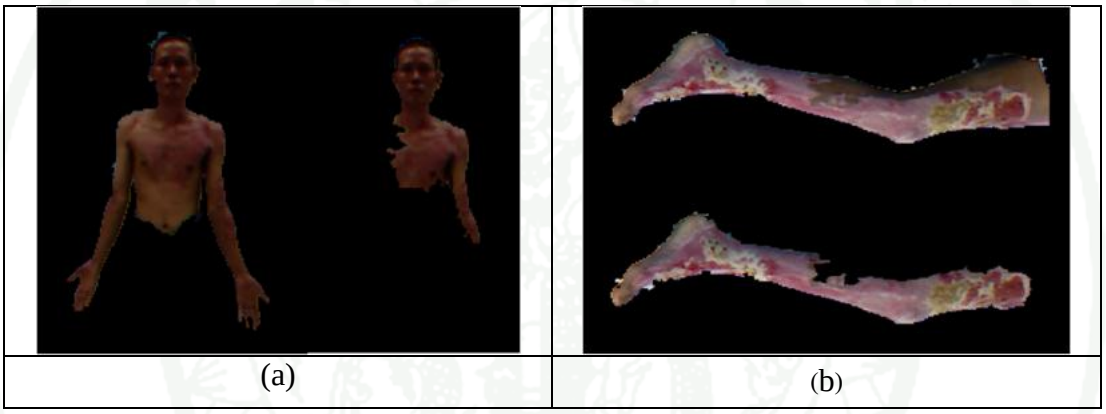
Table 4 Comparison results of ground truth and average filter techniques.

Surface Area Distances (in front of views)		**Result of Surface Area (cm^2)**			**%Error of Surface Area**	
Width (cm)	Length (cm)	Actual Area	Marching Cubes	Marching Cubes (Normalized)	Marching Cubes	Marching Cubes (Normalized)
5.11713	8.28747	42.40806136	68.1104	45.3756	60.6072002	6.997581459
4.66838	7.91935	36.97053515	60.3286	39.4187	63.1802184	6.621935108
4.76699	9.06871	43.23044988	52.9238	46.8473	22.4225058	8.36441078

From the table.1, the sizes of sample rectangular are used to estimate actual surface area for comparing with the surface area that is calculated from a set of point cloud. Using marching cubes algorithm, triangular meshes of object is generated to

calculate the surface area that is shown in marching cubes table. Also, an average filter is used to normalize the result of surface area that is shown in marching cubes (Nomalized) table. According to error of surface area, the resulting area with average filter is more accurate than without average filter and is similar to the actual surface area. It is clear that average filter (normalization) enhances the correctness of surfaces dramatically.

Second, we segment the burn area and the skin area as shown in Figure 34. An example of such area's point cloud is shown in Figure 35.



(a) (b)

Figure 50 (a-b) The segmentation of burn area and the skin area.

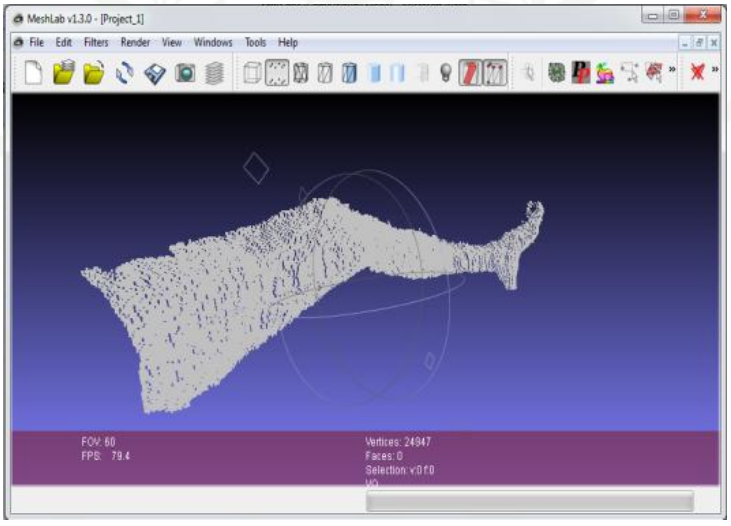


Figure 51 The rest point cloud is deleted that shows in MeshLab Program

Figure 52 Extract black-white-black-white features 48 points on chessboard that upper images are captured from the depth camera and lower images are captured from the color camera (There are 16 images: 8 depth images and 8 color images).

CONCLUSION

This research presents a comprehensive Kinect calibration and enhancement method. For calibration, first, we calibrate depth from raw depth information. Second, we calibrate Euclidean coordinate using the depth information and intrinsic parameters.

Even after good calibration is performed, limitations of Kinect are discussed. First, error in depth information increases by a factor of square distance from the sensor. This error would result in an unorganized point cloud. Second, Kinect's sensing is weak if the surface is shiny or transparent. Also, capturing under sunlight is not recommended.

Our method improves the unorganized point cloud from Kinect by an average filter. Experiment is done with the target objects in the range between 0.5 to 1 meters. Error rate of triangular mesh area reduces to a great deal. The remaining error, in Table 1, is assumed to be that due to inaccuracy of Kinect itself as mentioned in, Ruchanurucks *et al.* (2008)

LITERATURE CITED

Bouvrie, B. 2011. **Improving RGBD Indoor Mapping.** Academic Press, the Netherlands.

Burn Care. 2012. **Nopparat Rajathanee Hospital.** Academic Press, Thailand. Available Source: http://www.nopparat.go.th/, July 30, 2012

Cline, H. E. and W. E. Lorensen. 1987. **Marching cubes: a high resolution 3D surface construction algorithm.** *Proceeding SIGGRAPH '87 Proceedings of the 14th annual conference on Computer graphics and interactive techniques*.

Conley, K. 2011. Kinect_Node, Wikipedia. Available Source: http://www.ros.org/wiki/kinect_node.

Jianhui, Z., Y. Zhiyong, D. Yihua, Z. Yuanyuan and L. Chengjiang. 2009. **Improvements on IPD Algorithm for Triangular Mesh Reconstruction.** *International Conference on Multimedia Information Networking and Security*, 2009.

Jones, B. and R. Sodhi. 2010. **Kinect-projector calibration.** CS 498 Computational Photography, University of Illinois at Urbana-Champaign.

Khongma, A., M. Ruchanurucks, T. Phatrapornnant, Y. Koike and P. Rakprayoon. 2012. **Kinect Calibration and Quality Improvement for Triangular Mesh Reconstruction**. The International Conference on Information and Communication Technology for Embedded Systems. Bangkok, Thailand.

Khoshelham, K. 2011. **Accuracy analysis of kinect depth data.** *International Conference on Society for Photogrammetry and Remote Sensing*. p. 6.

Lenz, M., H. Bischof and M. Ruther. 2011. **µNect: On using a gaming RGBD camera in micro-metrology applications.** *Computer Vision and Pattern Recognition Workshops.* pp. 52 - 59.

Šolony, M. 2011. **Scene Reconstruction from Kinect Motion.** Electrical Engineering, Information and Communication Technologies (EEICT). Doctoral Degree Programme (2), FIT BUT.

Ruchanurucks, M., K. Ogawara, and K. Ikeuchi. 2008. **Integrating Region Growing and Classification for Segmentation and Matting.** Image Processing, 2008. ICIP 2008. 15th IEEE International Conference. Tokyo.

Teardown. 2010. **Microsoft Kinect Teardown.** Available Source: http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/1.

Wedro, B. C. 2012. **Burn Percentage in Adults: Rule of Nines.** Available Source: http://www.emedicinehealth.com/burn_percentage_in_adults_rule_of_nines/article_em.htm, July 30,2012.

Wilson, J. W. 1986. **PROBLEM SOLVING WITH HERON'S FORMULA.** *the Northwest Mathematics Conference*.

Zlatka, M. 2009. **Triangulation methods.** *Triangulation Methods.* pp. 9-10.

CIRRICULUM VITAE

NAME : Miss Amornrat Khongma

BIRTH DATE : February 14, 1988

BIRTH PLACE : Bangkok, Thailand

EDUCATION :

YEAR	**INSTITUTE**	**DEGREE/DIPLOMA**
2010	Kasetsart Univ.	B.Eng. (Electrical Engineering)
2012	Kasetsart Univ.	M.Eng. (Information and Communication Technology for Embedded Systems)

POSITION/TITLE : -

WORK PLACE : -

SCHOLARSHIP/AWARDS : TAIST ICTES Master Degree Scholarship