



BACHELOR THESIS & COLLOQUIUM – ME184841

**COMPUTER-AIDED DEVELOPMENT OF A MEASURING AND
EVALUATION METHOD FOR DETERMINING THE SIZE OF THE
OVERALL SURFACES OF MACHINE COMPONENTS**

MUHAMMAD RIFQI RAMADHAN
NRP. 04211541000042

SUPERVISOR:
Prof. Dr.-Ing. Achmed Omar
Prof. Dr.rer.nat. Ute Schreiber

DOUBLE DEGREE PROGRAM
DEPARTMENT OF MARINE ENGINEERING
FACULTY OF MARINE TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019



BACHELOR THESIS & COLLOQUIUM – ME184841

**COMPUTER-AIDED DEVELOPMENT OF A MEASURING AND
EVALUATION METHOD FOR DETERMINING THE SIZE OF THE
OVERALL SURFACES OF MACHINE COMPONENTS**

**MUHAMMAD RIFQI RAMADHAN
NRP. 0421154100042**

SUPERVISOR:

Prof. Dr.-Ing. Achmed Omar

Prof. Dr.rer.nat. Ute Schreiber

**DOUBLE DEGREE PROGRAM
DEPARTMENT OF MARINE ENGINEERING
FACULTY OF MARINE TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019**

“This page intentionally left blank”



SKRIPSI – ME184841

**COMPUTER AIDED DEVELOPMENT DARI METODE PENGUKURAN
DAN EVALUASI UNTUK MENENTUKAN UKURAN PERMUKAAN
KESELURUHAN DARI KOMPONEN-KOMPONEN MESIN**

**MUHAMMAD RIFQI RAMADHAN
NRP. 0421154100042**

SUPERVISOR:

Prof. Dr.-Ing. Achmed Omar

Prof. Dr.rer.nat. Ute Schreiber

**PROGRAM DOUBLE DEGREE
DEPARTMEN TEKNIK SISTEM PERKAPALAN
FAKULTAS TEKNOLOGI KELAUTAN
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019**

“This page intentionally left blank”

APPROVAL FORM

**COMPUTER-AIDED DEVELOPMENT OF A MEASURING AND EVALUATION
METHOD FOR DETERMINING THE SIZE OF THE OVERALL SURFACES OF
MACHINE COMPONENTS**

BACHELOR THESIS

Submitted to Comply One of The Requirement to Obtain a Bachelor Engineering Degree

On

Marine Operation and Maintenance (MOM)

Bachelor Program Department of Marine Engineering

Faculty of Marine Technology

Institut Teknologi Sepuluh Nopember

Prepared by:

MUHAMMAD RIFQI RAMADHAN

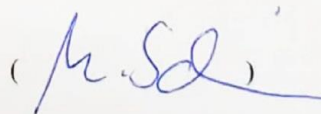
NRP. 04211541000042

Approved by Supervisor:

Prof. Dr-Ing Achmed Omar

()

Prof. Dr.rer.nat. Ute Schreiber

()

“This page intentionally left blank”

APPROVAL FORM

COMPUTER-AIDED DEVELOPMENT OF A MEASURING AND EVALUATION METHOD FOR DETERMINING THE SIZE OF THE OVERALL SURFACES OF MACHINE COMPONENTS

BACHELOR THESIS


Submitted to Comply One of the Requirements to Obtain Bachelor Engineering Degree

on

Bachelor Program Department of Marine Engineering
Faculty of Marine Technology
Institut Teknologi Sepuluh Nopember

Prepared by:
MUHAMMAD RIFQI RAMADHAN
NRP. 0421154100042

Approved by
Head of Department of Marine Engineering



Dr. Eng. Muhammad Badrus Zaman, S.T., M.T.
NIP. 197708022008011007

“This page intentionally left blank”

DECLARATION OF HONOR

I hereby who signed below declare that:

This bachelor thesis has written and developed independently without my plagiarism act, and confirm consciously that all data, concepts, design, references, and material in this report own by Marine Operation and Maintenance (MOM) in Department of Marine Engineering ITS which are the product of research study and reserve the right to use for further research study and its development.

Name : Muhammad Rifqi Ramadhan
NRP : 04211541000042
Bachelor Thesis Title : Computer-Aided Development of a Measuring and Evaluation Method for Determining the Size of the overall Surfaces of Machine Components
Department : Marine Engineering

If there is plagiarism act in the future, I will fully responsible and receive the penalty given by ITS according to the regulation applied.

Surabaya, 2019

Muhammad Rifqi Ramadhan

“This page intentionally left blank”

**COMPUTER-AIDED DEVELOPMENT OF A MEASURING AND
EVALUATION METHOD FOR DETERMINING THE SIZE OF THE
OVERALL SURFACES OF MACHINE COMPONENTS**

Name : Muhammad Rifqi Ramadhan
NRP : 04211541000042
Department : Marine Engineering
Supervisor I : Prof. Dr.-Ing. Achmed Omar
Supervisor II : Prof. Dr.rer.nat. Ute Schreiber

ABSTRACT

With the development of advanced technology in the maritime industry, it adds to the need for several technologies to support this industry. The idea is how to find out the overall surface area of a convex object that usually occurs in the maritime industry that deals with large construction volumes, the overall surface area is calculated using finite set of projections, the projections used are parallel and centralized projections, all data are simulated using MATLAB. Problems arise in how to project objects in several directions, normalizing objects if using central projections and determining minimum projections needed to determine the overall surface area. It turns out that the position of the object and the direction of projection affect the results, this problem is solved by using spherical coordinates for object rotation and projecting objects only from one particular direction, for center projection objects is scaled depending on the direction of projection, and all the overall surface area of the developed method validated by calculate simple convex objects and compare the results from mathematical formulas.

Keywords: Parallel Projections, Central Projections, Overall Surface Area

“This page intentionally left blank”

**COMPUTER AIDED DEVELOPMENT DARI METODE PENGUKURAN
DAN EVALUASI UNTUK MENENTUKAN UKURAN PERMUKAAN
KESELURUHAN DARI KOMPONEN-KOMPONEN MESIN**

Nama : Muhammad Rifqi Ramadhan
NRP : 04211541000042
Jurusan : Teknik Sistem Perkapalan
Dosen Pembimbing I : Prof. Dr.-Ing. Achmed Omar
Dosen Pembimbing II : Prof. Dr.rer.nat. Ute Schreiber

ABSTRAK

Dengan berkembangnya teknologi canggih di industri maritim, menambah kebutuhan akan beberapa teknologi untuk mendukung industri ini. Idenya adalah bagaimana mengetahui luas permukaan keseluruhan objek cembung yang biasanya terjadi pada industri maritim yang berurusan dengan volume konstruksi yang besar, luas permukaan keseluruhan dihitung dengan menggunakan proyeksi yang terbatas, proyeksi yang digunakan adalah proyeksi paralel dan terpusat, semua data disimulasikan menggunakan MATLAB. Masalah muncul dalam cara memproyeksikan objek pada beberapa arah, menormalkan objek jika menggunakan proyeksi sentral dan menentukan proyeksi minimum untuk menentukan luas permukaan keseluruhan. Ditemukan bahwa posisi objek dan arah proyeksi mempengaruhi hasil, masalah ini diselesaikan dengan menggunakan koordinat bola untuk rotasi objek dan memproyeksikan objek hanya dari satu arah tertentu, untuk proyeksi pusat objek akan diskalakan tergantung pada arah proyeksi, dan semua luas permukaan keseluruhan dari metode yang dikembangkan divalidasi dengan menghitung objek cembung sederhana dan membandingkan hasil dari rumus matematika.

Kata kunci: Proyeksi Paralel, Proyeksi Sentral (Perspektif), Luas Permukaan Keseluruhan

“This page intentionally left blank”

PREFACE

Above all, the author gave thanks and praise to the Almighty God, Allah SWT and also Prophet Muhammad SAW who has given me strength and wisdom so that the author can complete his studies and this bachelor thesis. Hopefully with the completion of this research study, authors gain more perspectives, information, and knowledge for the future career.

The author owed sincere and earnest appreciation for those who had helped, guided through and provided the author vital suggestion in the completion of this bachelor thesis.

1. Author's beloved parents, Bapak Budi Revianto, Ibu Rosnita who are being patient all this time raising author up! Thanks for everything, this is for you!
2. Author's Brother and Sister, Raihan Hilmy and Raisa Sabita.
3. My Second family in Rostock, Persatuan Pelajar Indonesia Chapter Rostock.
4. A. Tazkiyah Batari Uleng, S.Ked, for giving all the courage, advice, and inspirations.
5. Dr. Eng. M. Badrus Zaman, ST., M.T. as Chairman of Marine Engineering Department, Marine Technology Faculty of Institut Teknologi Sepuluh Nopember
6. Prof. Dr.-Ing. Achmed Omar as not only My Supervisor who through his fruitful of knowledge, encouragement, and guidance supervised the author through his bachelor thesis work and provided him with vital suggestions and advice, but also an inspiring, amazing, wonderful and helpful person for me.
7. Prof. Dr.rer.nat Ute Schreiber as Supervisor who through his fruitful of knowledge, encouragement, and guidance supervised the author through his bachelor thesis work and provided him with vital suggestions and advice.
8. SALVAGE'15 as author Batch, thanks for the memories. Thanks to Bismarck'12, Barakuda'13, Mercusuar'14, Voyage'16, Badrikara'17 to paint my collage life!
9. Jemaah Masjid Manarul Ilmi ITS, thanks for all the experience in building the author's character.
10. Barunastra RoboBoat Team ITS, as one of the author's place for character development
11. Fellow colleague the author who always there in bitter sweet conditions.

The author concerns for the imperfection in this thesis. Therefore, any criticism and suggestion are expected. The author hoped this thesis will provide benefits primarily for the readers.

Surabaya, 2019

Muhammad Rifqi Ramadhan

“This page intentionally left blank”

TABLE OF CONTENTS

APPROVAL FORM	vi
APPROVAL FORM	viii
DECLARATION OF HONOR	x
ABSTRACT	xii
ABSTRAK	xiv
PREFACE	xvi
TABLE OF CONTENTS	xviii
LIST OF FIGURES	xx
LIST OF TABLES	xxi
1 CHAPTER I INTRODUCTION	1
1.1 Background	1
1.2 Problem Statements	4
1.3 Research Limitations	4
1.4 Research Objectives	5
2 CHAPTER II THEORITICAL	7
2.1 Projection	7
2.2 Area Calculation	11
3 CHAPTER III RESEARCH METHODOLOGY	13
3.1 Research Approach	14
3.2 Define Object Parameters	14
3.3 Object Rotations	16
3.4 Projections and Area Calculations	18
3.5 Conclusion and Outlook	20

4	CHAPTER IV ANALYSIS AND RESULTS	21
4.1	Overall Area Surface Calculation from Parallel Projection	21
4.2	Validation and Error Testing of Developed Calculation Method (Parallel Projection).....	21
4.3	Summary of Overall Surface Area Calculation (Parallel Projection)	34
4.4	Overall Area Surface Calculation from Central Projection.....	35
4.5	Validation and Error Testing of Developed Calculation Method (Central Projection)	39
4.6	Summary of Overall Surface Area Calculation (Central Projection)	48
5	CHAPTER V CONCLUSION AND OUTLOOK.....	51
	REFERENCES	53
	APPENDIX	55
	Appendix A: Main Script Function	56
	Appendix B: AutoProject Function.....	61
	Appendix C: AutoProjectCentral Function.....	62
	Appendix D: Viewmtx Function	64
	Appendix E: rotate_3D Function.....	66
6	AUTHOR BIOGRAPHY	67

LIST OF FIGURES

Figure 1.1 Image processing acquisition system example	1
Figure 1.2 Original and processed images with two different views per product..	2
Figure 1.3 Simple Rectangular Box.....	3
Figure 1.4 Various projections of cube above plane.....	4
Figure 2.1 Parallel projection of a cube, left is orthographic and right is oblique.	8
Figure 2.2 Viewing volume in orthographic projection.....	8
Figure 2.3 Perspective projection types	9
Figure 2.4 One-points perspective projection.....	10
Figure 2.5 Example of distorted picture (left) and undistorted picture (right)	10
Figure 2.6 Polygon Example	11
Figure 3.1 Methodology Flowchart	13
Figure 3.2 Generated 25 Random Points between 0-25 (Left) and 0-10 (Right) with rng value at 0	15
Figure 3.3 Generated Convex Object from 25 Radnom Points between 0-25 (Left) and 0-10 (Right) with rng value at 0.....	15
Figure 3.4 Example of Object Rotation in X Axis from determined angle that ranging from 0-360 Degrees.....	16
Figure 3.5 Example Example of Object Rotation in Y Axis from determined angle that ranging from 0-360 Degrees.....	17
Figure 3.6 Spherical coordinates (r, θ , ϕ), radial distance r, azimuthal angle θ , .	17
Figure 3.7 Rotation Orientation Coordinates from 10 by 10 matrix therefore will make 100 vector of orientation.....	18
Figure 3.8 Examples of Projected Area of Figure 3.4.....	19
Figure 3.9 Examples of Projected Area of Figure 3.5.....	19
Figure 4.1 Icosahedron for AutoProject function validation.....	23
Figure 4.2 Test Object 1:Tetrahedron with edge 10	24
Figure 4.3 Test Object 2:Tetrahedron with edge 1.414.....	27
Figure 4.4 Test Object 3: Octahedron	28
Figure 4.5 Test Object 4: Hexahedron (Cube)	30
Figure 4.6 Test Object 5: Dodecahedron	31
Figure 4.7 Test Object 6: Icosahedron	33
Figure 4.8 Camera Graphics Terminology in Matlab	36
Figure 4.9 Central Projection Example, top left is main object seeing from front or Y direction and top right is the projected area from the front, bottom left is simulated central projection with AZ and EL 0, AOV 10, and camera target at the middle of the object,	38
Figure 4.10 Central Projection Example of First Object with Original Scale, Depth Calibraion of 10 and AOV at 10.....	41
Figure 4.11 Central Projection Example of First Object with Original Scale, Depth Calibraion of 10 and AOV at 25.....	44

“This page intentionally left blank”

LIST OF TABLES

Table 4.1 Object Details for Validation and Error Testing (Parallel Projection)	22
Table 4.2 Theta Details for Validation and Error Testing (Parallel Projection)	22
Table 4.3 Results of AutoProject Function Projections	23
Table 4.4 Object 1 Overall Surface Area Results	25
Table 4.5 Object 1 Error Test Results (Percentage).....	26
Table 4.6 Object 2 Overall Surface Area Results	27
Table 4.7 Object 2 Error Test Results (Percentage).....	28
Table 4.8 Object 3 Overall Surface Area Results	29
Table 4.9 Error Test Results (Percentage).....	29
Table 4.10 Object 4 Overall Surface Area Results	30
Table 4.11 Object 4 Error Test Results (Percentage).....	31
Table 4.12 Object 5 Overall Surface Area Results	32
Table 4.13 Object 5 Error Test Results (Percentage).....	32
Table 4.14 Object 6 Overall Surface Area Results	33
Table 4.15 Object 6 Error Test Results (Percentage).....	34
Table 4.16 Error Test Result Summary (Highest Accuracy).....	34
Table 4.17 Error Test Result Summary (Lowest Accuracy)	35
Table 4.18 Angle of View Value Representation	36
Table 4.19 Object Overall Area Surface at Original.....	39
Table 4.20 Object Overall Area Surface at 5 Times Scaled.....	40
Table 4.21 Object Overall Area Surface at 10 Times Scaled.....	40
Table 4.22 Scale Testing, at AOV 10, object original scale, depth at original, -5, and -10	41
Table 4.23 Scale Testing, at AOV 10, object 5 times scaled, depth at original, -5, and -10	42
Table 4.24 Scale Testing, at AOV 10, object 10 times scaled, depth at original, -5, and -10.....	43
Table 4.25 Scale Testing, at AOV 25, object original scale, depth at original, -5, and -10	44
Table 4.26 Scale Testing, at AOV 25, object 5 times scaled, depth at original, -5, and -10	45
Table 4.27 Scale Testing, at AOV 25, object 5 times scaled, depth at original, -5, and -10	46
Table 4.28 Summary of Scale Estimation with Angle of View/Phi 10.....	47
Table 4.29 Summary of Scale Estimation with Angle of View/Phi 25.....	47
Table 4.30 Scale Estimation Tested on Main Object with Angle of View/Phi 10	48
Table 4.31 Scale Estimation Tested on Main Object with Angle of View/Phi 25	48

“This page intentionally left blank”

CHAPTER I INTRODUCTION

1.1 Background

Knowledge of surface area of mechanical components can help people to work more precisely, beside the use of physical measurements tool there are also digital measurements method with image processing, it can be done by working with pixel data or basically the images matrix data because every digital image has a corresponding matrix of color and color intensities [9].

Image processing itself have several ways to calculate an area surface, image processing methods first step was made when a photographic method was proposed to estimate surface area and volume of eggs. The top and bottom of the eggs were considered as segments of spheres and the sections between were considered elementary cylinders. Although it had good results, this method was also time consuming making it impractical for large number of samples [11].

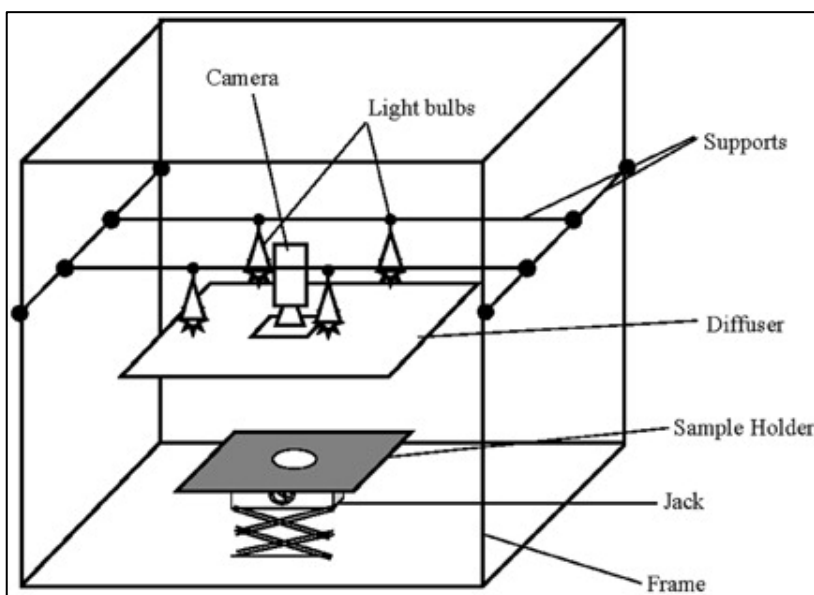


Figure 1.1 Image processing acquisition system example

(Source: C.M. Sabliov, et al. 2005)

There was also another image processing to determine surface area and volume for axisymmetric agricultural products that determine with three steps: 1) image acquisition, 2) image processing, and 3) volume and surface area computation [10].

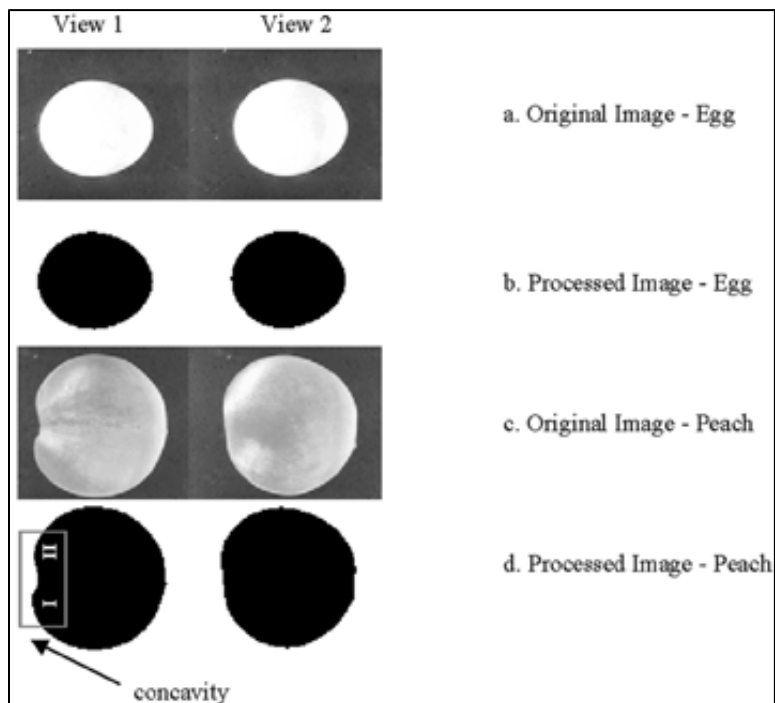


Figure 1.2 Original and processed images with two different views per product

(Source: C.M. Sabliov, et al. 2005)

The idea is to get the object picture, convert to binary image, save as bitmap files, then process in MATLAB to calculate image pixels to get surface area, and using one representative products for calibration to know the length of each pixel.

There is another idea to calculate overall surface of an 3D object by using projections, where the 3D object is projected in several sides then the object surface area is calculated. It will be easy if the object is only simple box, cube, pyramid, etc. Because to calculate the overall surface area is just by using basic formula of the object itself.

For the overall surface area of the rectangular box (see Figure 1.3), there are 2 side creating large rectangle (the front and behind side), 2 side create more smaller rectangle (the top and bottom side), and 2 side create the smallest rectangle (the left and right side). Basically, to get the overall surface area, all the rectangle sides are calculated using rectangle formula and then add in total to known the overall surface area.

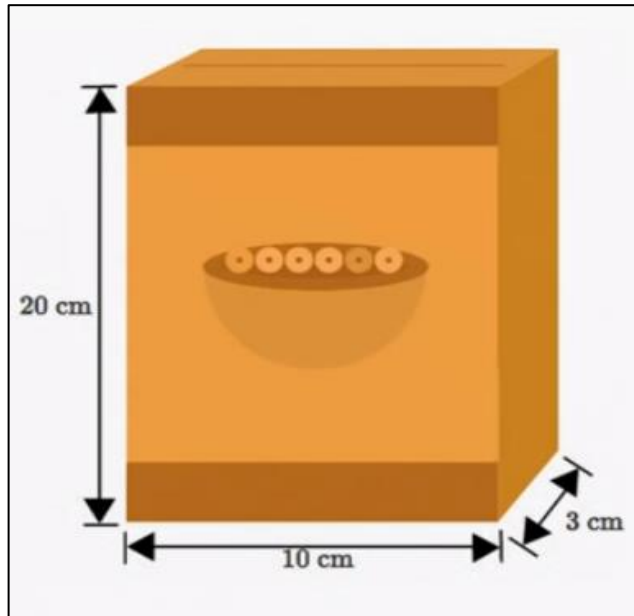


Figure 1.3 Simple Rectangular Box

(Source: khanacademy.org)

$$SA = 2LW + 2LH + 2WH \quad (1.1)$$

Where,

SA = Surface Area

L = Length of the box

H = Height of the box

W = Width of the box

But the things will get complicated if the object is not a basic object especially in marine industry like ship hull, propellers, tanks, doors, windows, etc. The idea in this bachelor thesis to calculate the overall surface area is by projecting objects to certain axis the projection method is parallel and central/perspective projection, then rotate to several direction to get different projection and calculate the projected area of the object to get estimation of the overall surface area of the object. All of the projections will be simulated and calculated using MATLAB, the accuracy of this method is depending on how many projections used, the object rotation, and for the central projection the angle of view and camera distance from the object is also matters to know scale that must be used for correction due to object distortion.

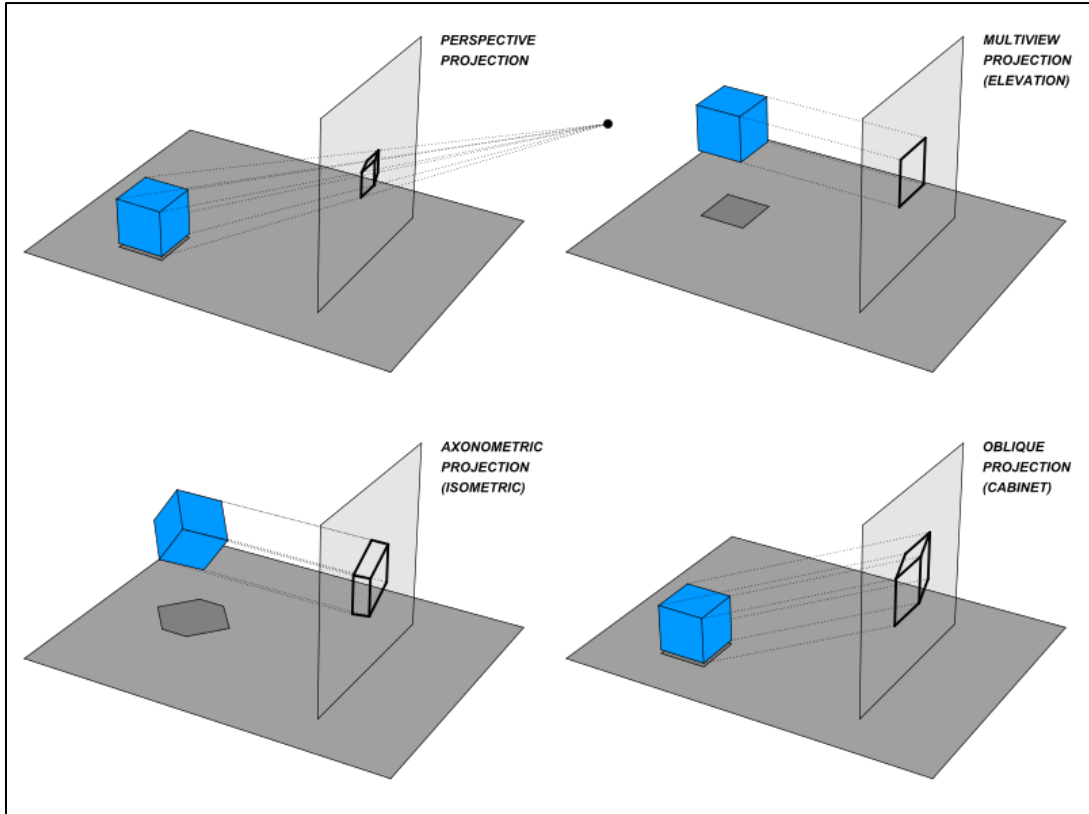


Figure 1.4 Various projections of cube above plane

(Source: wikipedia.org)

1.2 Problem Statements

Based on the background that has been described above, the problems of this bachelor thesis are:

- How to estimate the minimum number and perspectives of the necessary image recordings of the object in case of parallel projection?
- How to make evaluation procedure to determine the overall surface area of the component on the basis of the finite set of projection surfaces?
- How to make correction of the measured projection surfaces due to central projection in the case of digital image recordings and make development of a practicable scaling possibility to determine the area size?

1.3 Research Limitations

The limitations of this bachelor thesis are:

- The object that used in calculation must be convex object not concave.

1.4 Research Objectives

The objectives in this bachelor thesis are:

- a. To estimate the necessary image recordings of the object in case of a parallel projection.
- b. To make development of an evaluation procedure to determine the overall surface area of the component on the basis of the finite set of projection surfaces.
- c. To make correction of the measured projection surfaces due to central projection of digital image recordings and make a development to practicable scaling possibility for determining the area size.

“This page intentionally left blank”

CHAPTER II THEORITICAL

In completing this bachelor thesis, various theories needed which will support the study, starting from the concept of projections in parallel and central/perspective projections. Another thing is about how to process the projected data to obtain the overall surface area of an object.

2.1 Projection

A projection is the transformation of points and lines in one plane onto another plane by connecting corresponding points on the two planes with parallel lines. [13]. This research mostly talking about 3D projection, which is a method of projection that mapping three-dimensional points to a two-dimensional plane, 3D projections contain 2 main method of projection, the parallel projection and central/perspective projection.

- **Parallel Projection**

A type of projection that displays an object in its true shape and size. The parallel projection is formed simply by extending the parallel lines from each vertex on the object until it intersects the plane of the screen. There are 2 types of parallel projection when the projection rays are perpendicular to the projection plane is called orthographic and when the projection rays are not perpendicular to the projection plane but strike the projection plane at an angle other than ninety degrees is called oblique [8]. The parallel projection in this research is using orthographic and simply produce by removing one of the axes, so if an object is defined in x, y, and z axis, removing the z axis will simulating a parallel projection from the top or z axis. And to get the object projections from other direction, the object is rotated but the view is still from the top because the object only projected at x and y axis, and this will be explained further in chapter analysis and results.

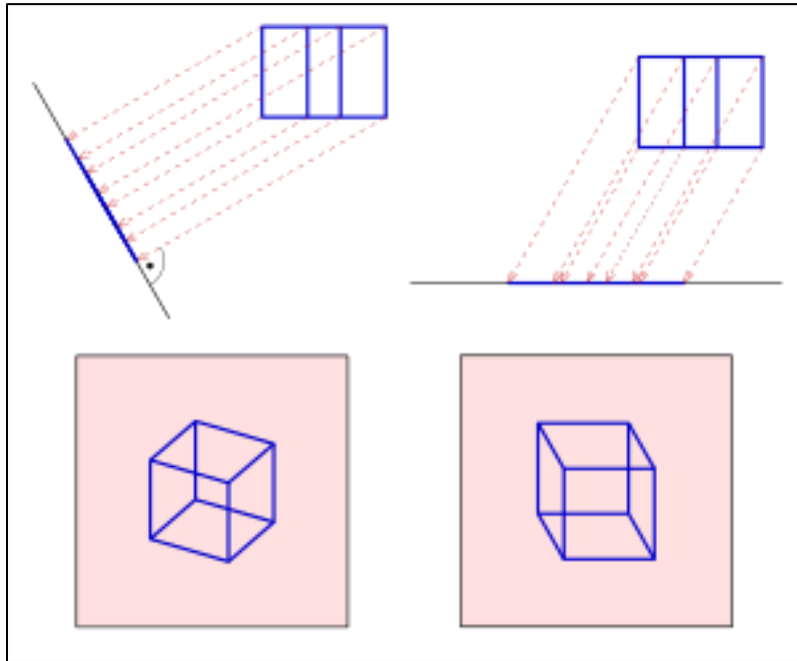


Figure 2.1 Parallel projection of a cube, left is orthographic and right is oblique
(Source: wikipedia.org)

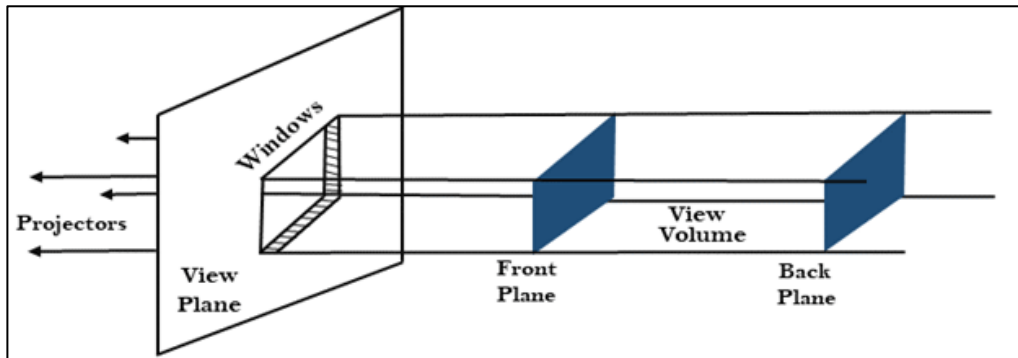
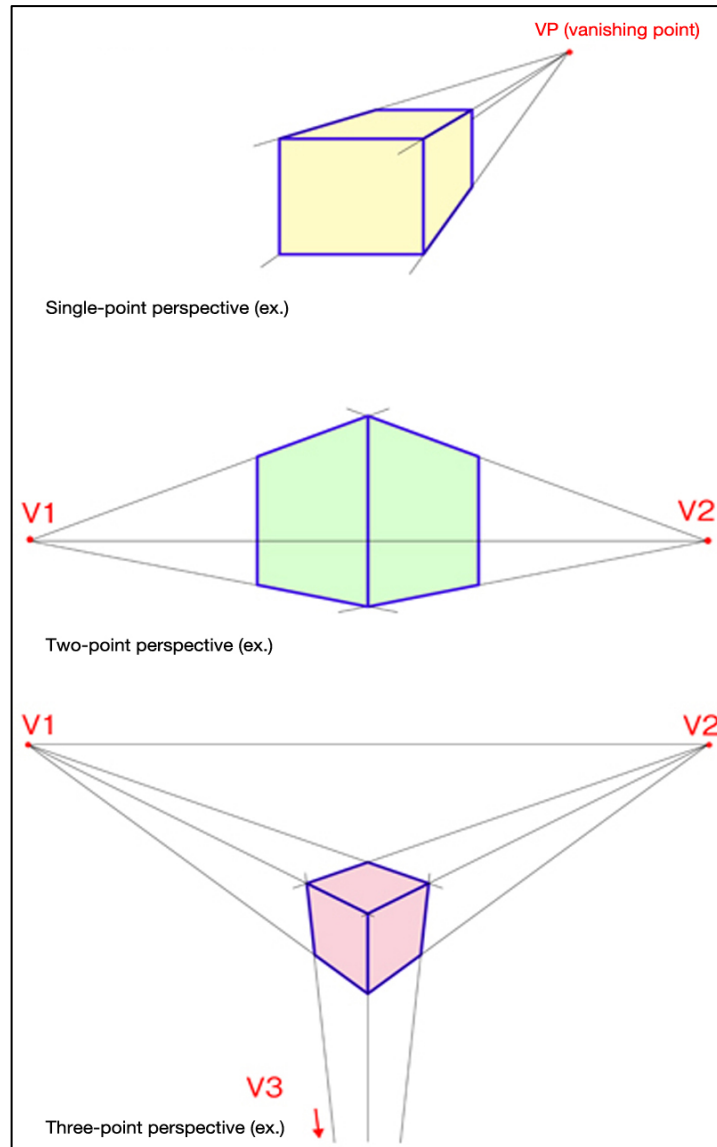


Figure 2.2 Viewing volume in orthographic projection
(Source: javatpoint.com)

- Central Projection

Also known as perspective projection, is a type of projection that a three-dimensional object is projected on a picture plane. This has an effect that distant object will appear smaller than nearer objects. Photographic lenses and the human eye are work in the same way, perspective projection is categorized into one-point, two-point, and three-point perspective that depends on the orientation of the projection plane towards the axes of the depicted object. [3].

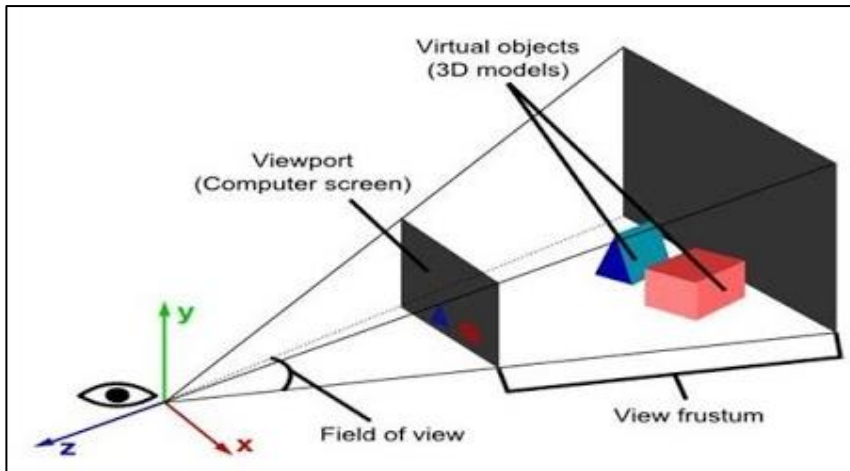


*Figure 2.3 Perspective projection types
(Source: art-design-glossary.musabi.jp)*

In this research only the one-point perspective is used because that is how the camera works and in MATLAB the camera can be simulated therefore the one-point perspective projection can be simulated.

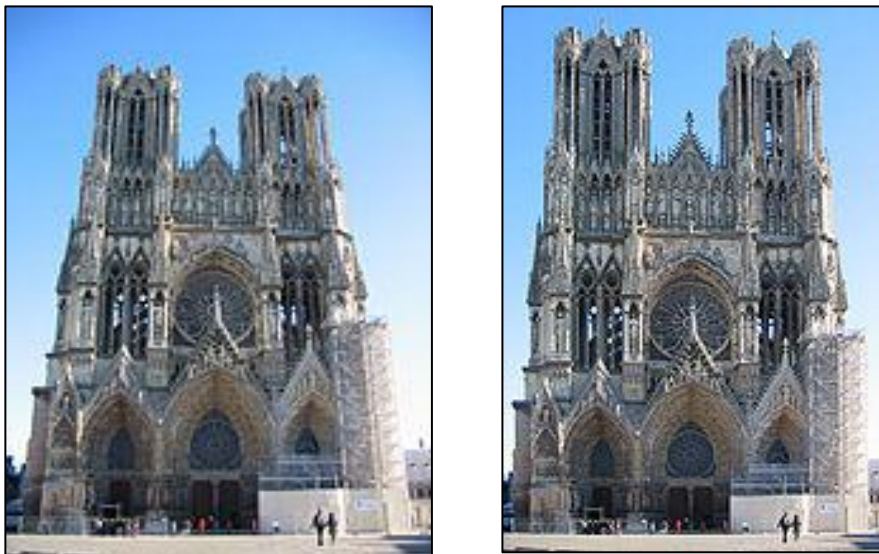
One-point perspective is affected by the camera/eye position, angle/field of view, and camera/eye look-at or also called target. In MATLAB all of these parameters can be adjusted in a built-in function, the function is *viewmtx* (Appendix D) that will give a view transformation matrix to transform the original object position to

projection plane/screen [7], and will be explained further in chapter analysis and results.



*Figure 2.4 One-points perspective projection
(Source: youtube.com)*

Also, in perspective projection there will be a distortion due to position of camera/eye, the angle of view and camera/eye target/look-at because the projection plane is not parallel to lines that are required to be parallel in the object. This distortion will make the projection more inaccurate, to minimize this distortion the camera target must be at the center of the object and angle of view must be proportional with the camera position. If the angle of view is high then the camera position must be closer to the object and vice versa [5].



*Figure 2.5 Example of distorted picture (left) and undistorted picture (right)
(Source: wikipedia.org)*

Distortion must be kept minimum to get more accurate data about length and width of the object in that projection. There is an algorithm that developed for perspective distortion control that can correct the image distortion [6]. but it only valid if the projection is already in an image from photographic lenses/camera, and in this research the simulated projection result is in matrix, and therefore the distortion can only be fixed by calibrating the camera target, angle of view, and camera position.

2.2 Area Calculation

After the object is projected, then the area of projection should be calculated, all of the projection are created from MATLAB built-in function the *convhull* function that create a convex hull of the projected object [12] and because projections are in 2D and the shape will be a polygon due to *convhull* the area can be calculated using another built-in function the *polyarea* that calculate area of polygon. [12]

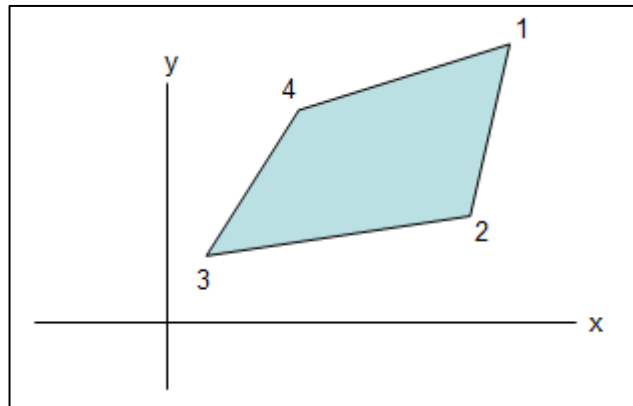


Figure 2.6 Polygon Example

(Source: mathopenref.com)

The *polyarea* is using the same principle in calculating area of a polygon in coordinate geometry, by formula:

$$area = \left| \frac{(X_1 Y_2 - Y_1 X_2) + X_2 Y_3 - Y_2 X_3) \dots + (X_n Y_1 - Y_n X_1)}{2} \right| \quad (2.1)$$

Where X_1 is the coordinate of vertex 1 and Y_n is the coordinate of the nth vertex etc.

After all the projection area is calculated, then to get the overall/total surface area, all of the projection area is averaged and multiple by 4 because “for every convex body the average area of its parallel projections into the plane always a quarter its surface is, or in other words, the expectation value for the randomly chosen projection direction for the ratio between the area of the projection and the content of the surface of the origin body is 1:4” (Cauchy, 1841) [1]. And also, the average

projected area over all orientations of any ellipsoid is $1/4$ the total surface area, this theorem also holds for any convex solid [13].

For the central projection it will be using the same method but there will be a scale value for correction of the overall area surface calculation due to projection distortion. This scale value is depending on object maximum length/width and the parameters in central projection, that is the angle of view and eye/camera target/look at.

CHAPTER III RESEARCH METHODOLOGY

To achieve the objectives that described in Chapter 1, a further explanation of research methodology is needed, in Figure 3.1 is a brief description of the developed research methodology to complete this bachelor thesis.

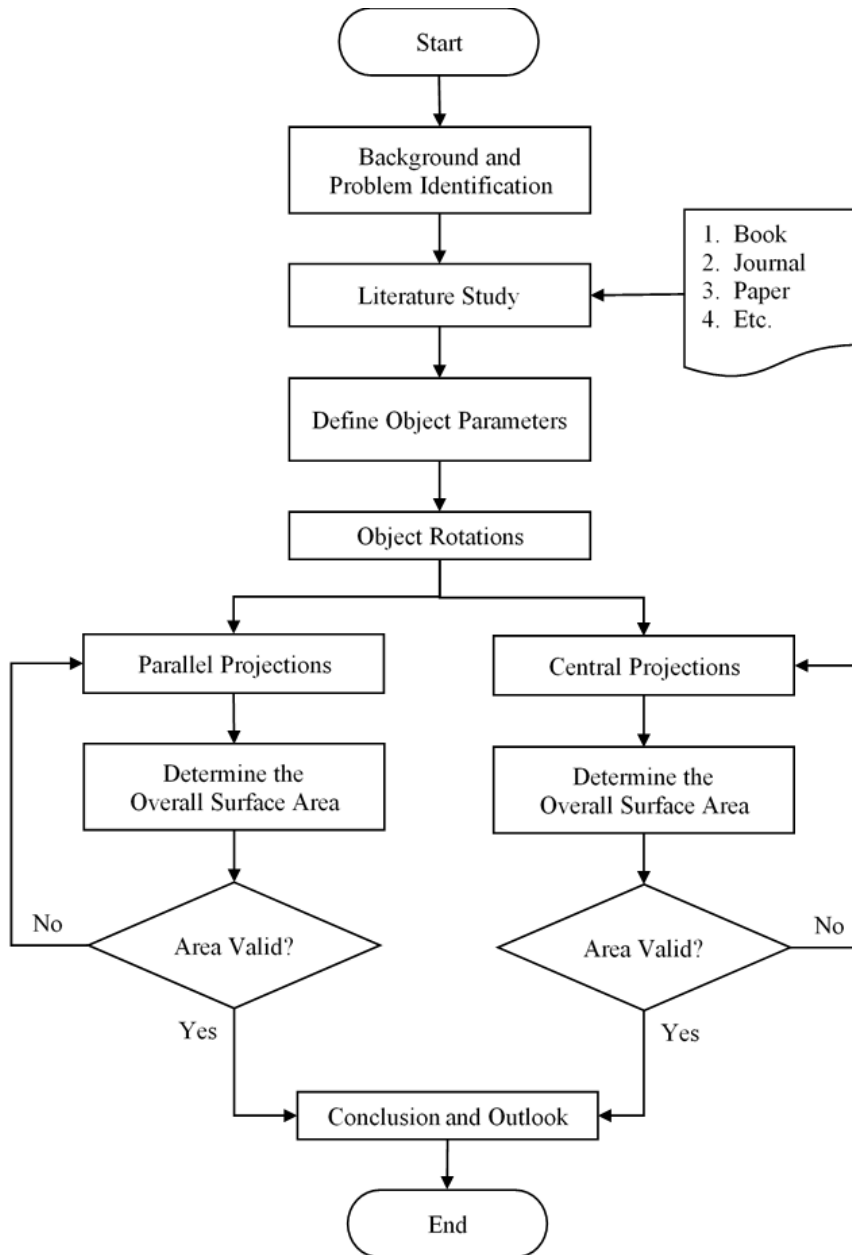


Figure 3.1 Methodology Flowchart

3.1 Research Approach

The approach of this research is by doing simulation in MATLAB (see [12] to know the version used) to simulate the projection and also calculate that area of projection to get the total surface area of an object. Before doing simulation, it is needed to understand the fundamental theories of projections and area calculation that can be found from various sources like book, journal, paper, etc. The theories are then linked to the goals to be achieved.

All of the simulations are needed to be validated and make corrections if there is a high amount of error, and also to get the best results, the simulation will be repeated many times.

3.2 Define Object Parameters

First step in this bachelor thesis after literature study is to define the object parameters of geometry types and morphology of components that to be measured. The convex object is selected as geometry types and also become the limitation of this research.

The object will be created in random convex shape by creating a virtual workspace for this research. The workspace in this research is box shaped by defining the same maximum and minimum value in x, y, and z domain.

For making the object, random points are used, there are generated random points that created in 3-dimensions and by using the *scatter3* built-in function in MATLAB that will plot the generated random points [12]. If less points are used the object will be more randomly shaped and if more points are used the object will be more box shaped. The objective of determining the amount of generated points is associated with object complexity. To hold the value of generated random points in every run for more easy reading, the *rng* function is used, that a built-in function to control the generated random number and can produce a predictable sequence of numbers [12]. Changing the *rng* value will also change the object shape.

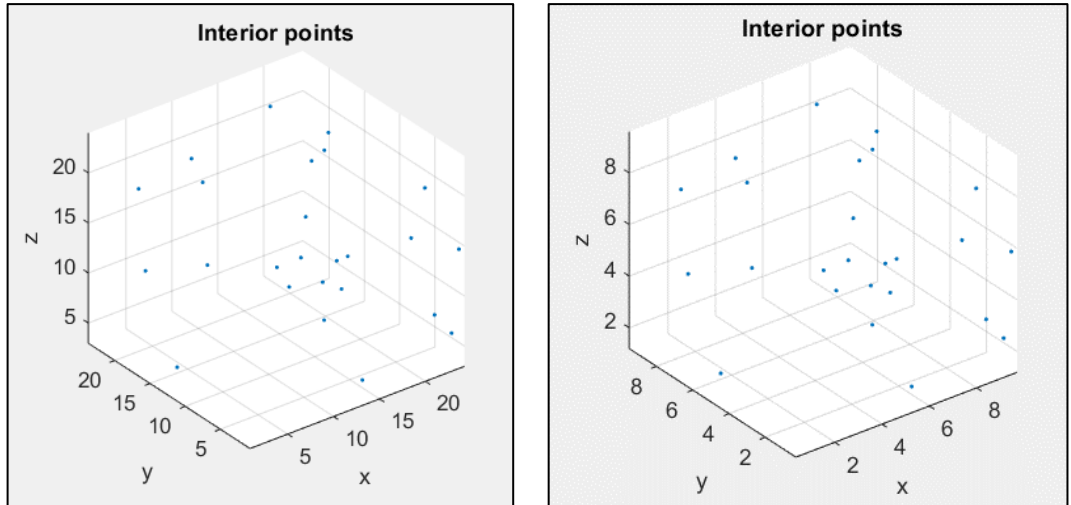


Figure 3.3 Generated 25 Random Points between 0-25 (Left) and 0-10 (Right) with *rng* value at 0

After the points is created, next is connecting the points to create a convex object using the *convhull* function, a built-in function that returns the convex hull of a set points in 2D/3D space and for plotting the generated convex object *trisurf* function is used that will make a triangular surface plot [12].

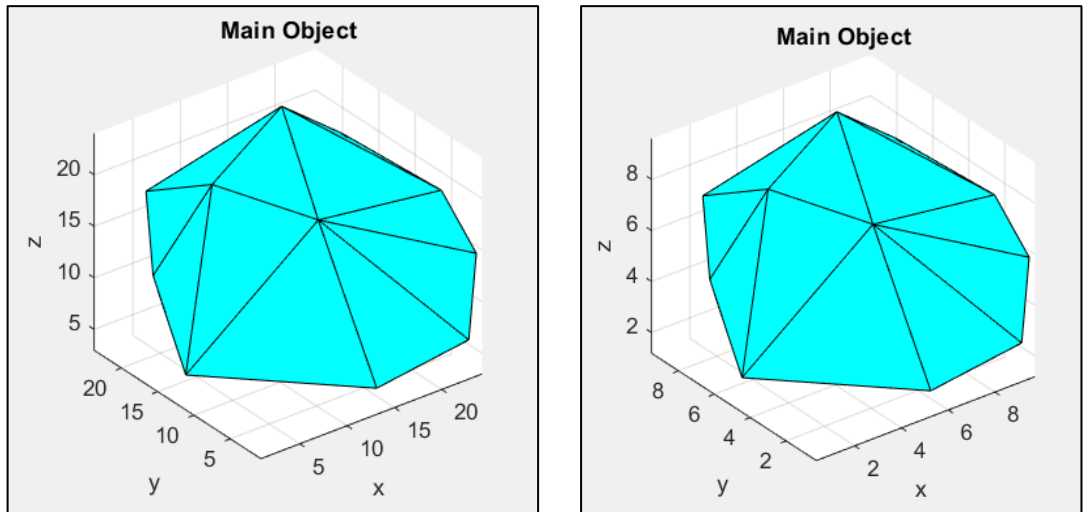


Figure 3.2 Generated Convex Object from 25 Radnom Points between 0-25 (Left) and 0-10 (Right) with *rng* value at 0

3.3 Object Rotations

For calculating the overall area surface, is it necessary to rotate the object in several directions to capture the object from different side. For this *rotate_3D* function is used (Appendix E), is a function that can compute the rotation of a vector in 3D space [2].

The angles that used for rotation is the special angles in the trigonometry, there are: 30, 45, 60, 90, 120, 135, 150, 180, 210, 225, 240, 270, 300, 315, 330, and 360. In total there are 16 rotation for simulating the rotation using *rotate_3D* function, and beside determine the rotation, the rotation axis is also need to be determined.

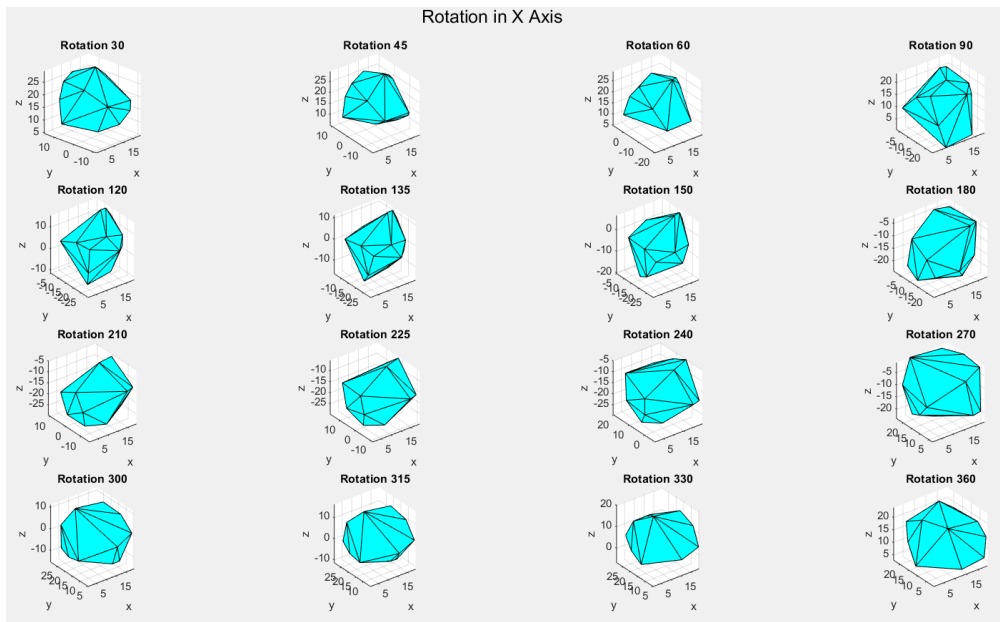


Figure 3.4 Example of Object Rotation in X Axis from determined angle that ranging from 0-360 Degrees

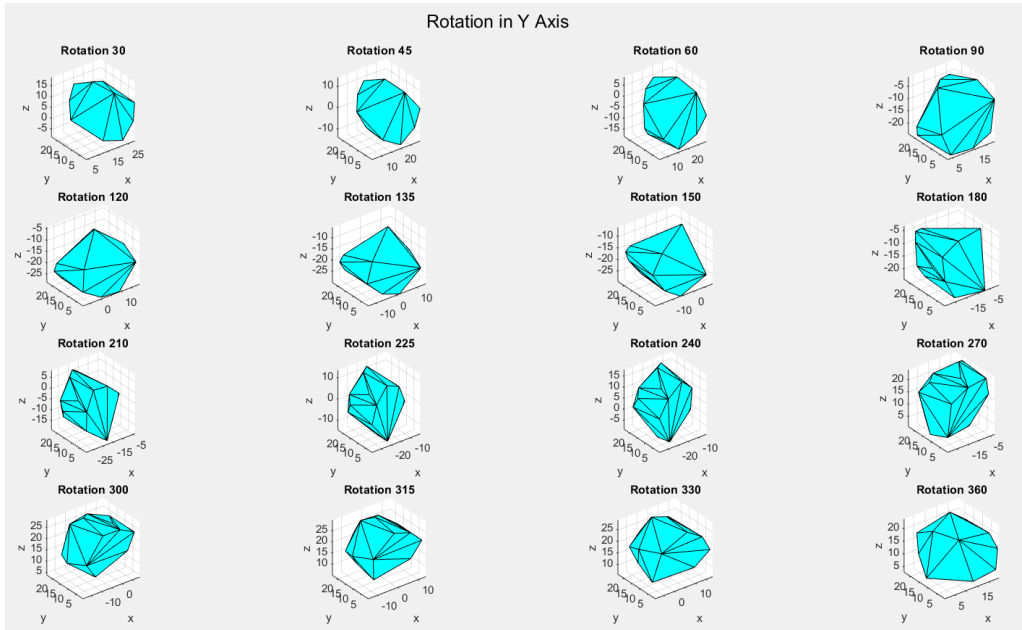


Figure 3.6 Example Example of Object Rotation in Y Axis from determined angle that ranging from 0-360 Degrees

To get all the object side the rotations in X-Axis and Y-Axis only is not enough, and for applying the Cauchy theorem for total surface area calculations the orientations must be in ellipsoid, therefore spherical coordinates system is used, spherical coordinates system is defined in radial distance, azimuthal angle and polar angle [4].

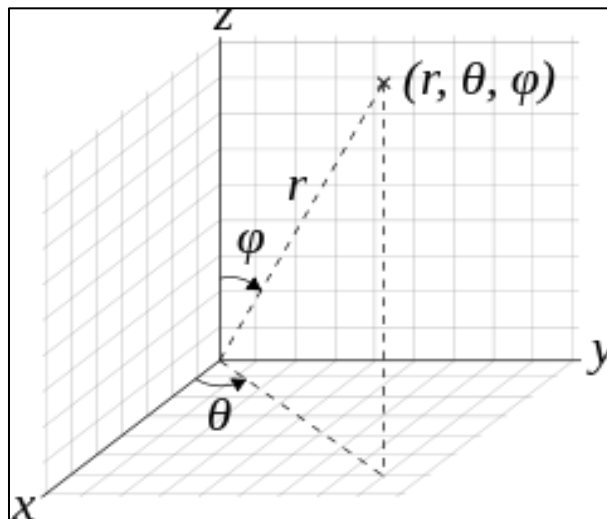


Figure 3.5 Spherical coordinates (r, θ, ϕ) , radial distance r , azimuthal angle θ , and polar angle ϕ .

(Source: wikipedia.org)

just as its name the spherical coordinates system is like making a sphere from dots matrix, and in each dot vector will become the axis pole of the rotation instead of only X and Y as a pole, but because the rotation need to be in cartesian coordinates, the spherical coordinates data is transform into cartesian by using built-in function *sph2cart* [12].

The more rows and columns added to the sphere will make more points for the orientation of the rotation and vice versa, therefore the rotation orientation coordinates used in this research is as shown in the figure below, that make 100 vector of orientation coordinates that distributed equally in a sphere shape.

3.4 Projections and Area Calculations

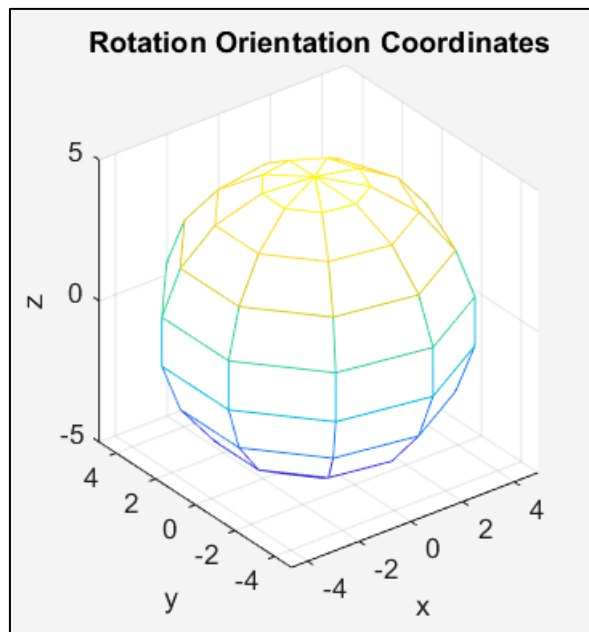


Figure 3.7 Rotation Orientation Coordinates from 10 by 10 matrix therefore will make 100 vector of orientation

After the object is rotated it will be projected in each of rotation. The projection will be creating a 2D polygon shape in therefore the area can be calculated with built-in function *polyarea*. This is examples of projected area from Figure 3.4 and 3.5 in Z direction or from the top and the area calculation using *polyarea*.

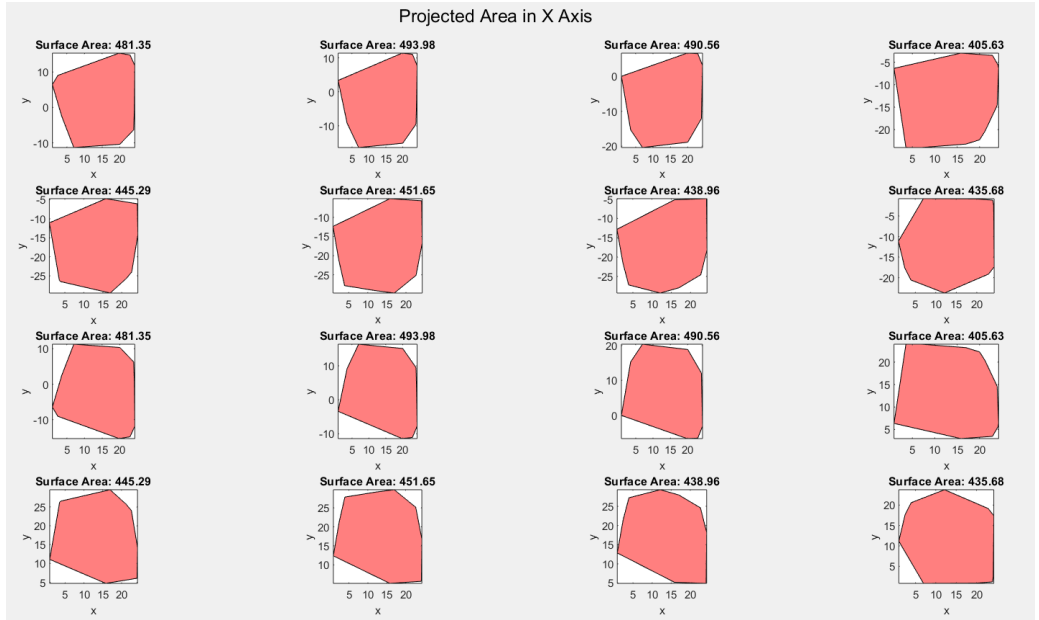


Figure 3.9 Examples of Projected Area of Figure 3.4

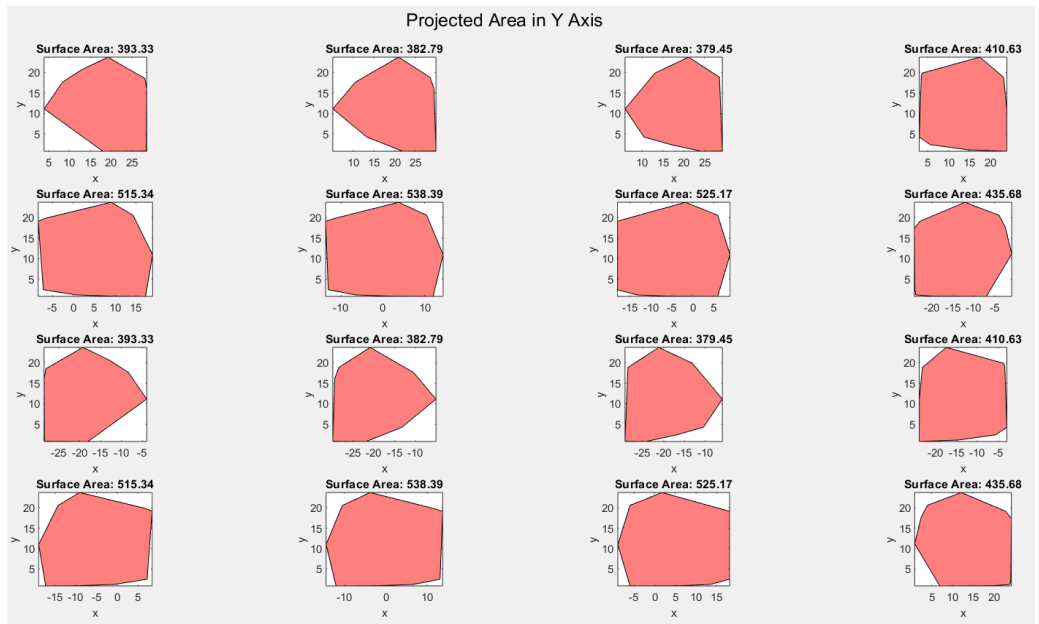


Figure 3.8 Examples of Projected Area of Figure 3.5

For the overall area calculation and how to get and process the projections data will be explained further in chapter analysis and results.

3.5 Conclusion and Outlook

The overall result of this research will be presented in this stage. Some cases, example conditions, and validation of developed method which has been analyzed previously will be served here as summary of this research and representative completions of the objectives. Also, there will be outlook based on what has been discussed and analyzed in this research and comments about the future research including some recommendations.

CHAPTER IV ANALYSIS AND RESULTS

4.1 Overall Area Surface Calculation from Parallel Projection

In parallel projection the object will be projected into the Z axis so it is like seeing the object from above. The object must only be plotted in one of the Axis because if more than one axis, is it not necessary to do the object rotation and will make the projection more complicated.

For calculating the overall surface area *AutoProject* function is used (Appendix B). This is a custom function that combining the rotation from *rotate_3D* and projection area calculation with *polyarea* function, it will return each area projection from all the rotation.

The input will be points in 3D space and also the orientation axis of the rotation that defined in variable u , then the function will rotate the object in certain angles. The orientation used are described in Chapter 3.3 and Figure 3.7, because the object is projected from Z axis the top and the bottom of the sphere will be creating the same projection so is not counted (the row 1 and 10).

In this function the theta that used to rotate in each u or the orientation are 15 thetas, 30, 45, 60, 90, 120, 135, 150, 180, 210, 225, 240, 270, 300, 315, and 330 Degrees. This rotation orientation coordinates and determined theta will be validated and tested in the next sub-chapter.

All of this projection will create 1200 projection in total, is counted by 8 rows (the top and bottom rows are not counted) multiplied by 10 columns and multiplied again by 15 (the theta points), then to get the overall surface area from this projections data, all of the projection area is averaged and multiple by 4 following the Cauchy theorem.

The object that created from chapter 3.2 with 25 points ranging from 0-10 and mg value of 0 will have total surface area of 296.22

4.2 Validation and Error Testing of Developed Calculation Method (Parallel Projection)

Validation is the important part of this research because with this the errors of the calculation can be known, for the validation process a simple convex object is needed because the real overall surface area can be easily calculated manually. There will be 6 objects for this error testing and for the test, parameter from the rotation orientation coordinates and the angle (theta) of the rotation.

The object to be used in testing is Platonic Solids objects, and the detail for the object, rotation orientation coordinates and theta used in the test will be detailed in the table below:

Table 4.1 Object Details for Validation and Error Testing (Parallel Projection)

No	Object	Formula	a	Overall Area Surface
1	Tetrahedron	$\sqrt{3}a^2$	10	173.21
2	Tetrahedron	$\sqrt{3}a^2$	1.414	3.46
3	Octahedron	$2\sqrt{3}a^2$	1.414	6.93
4	Hexahedron (Cube)	$6a^2$	1	6
5	Dodecahedron	$3\sqrt{25 + 10\sqrt{5}}a^2$	0.764	12.05
6	Icosahedron	$5\sqrt{3}a^2$	1	8.66

Table 4.2 Theta Details for Validation and Error Testing (Parallel Projection)

No	Theta	Degrees Used
1	Theta2	45 and 90
2	Theta3	90, 180, and 270
3	Theta4	30, 45, 60, and 90
4	Theta4b	45, 90, 135, and 180
5	Theta7	45, 90, 135, 180, 225, 270, and 315
6	Theta8	30, 45, 60, 90, 120, 135, 150, and 180
7	Theta15	30, 45, 60, 90, 120, 135, 150, 180, 210, 225, 240, 270, 300, 315, and 330 Degrees

For the rotation orientation coordinates matrix that used in the error test is from 6x6-15x15 but at the first object the matrix is 4x4 – 25x25, 30x30, 40x40, 50x50, 100x100, 200x200, and 300x300. The reason will be explained further, and for Theta15 is using everything described in Chapter 3.3 except the 360 Degrees, is removed because it doesn't make any move just like 0 Degrees.

Before the object test, the *AutoProject* function that create rotation orientation coordinates is tested, to proof that the rotation is happening in every dots of rotation orientation coordinates created. To proof it an icosahedron object with high

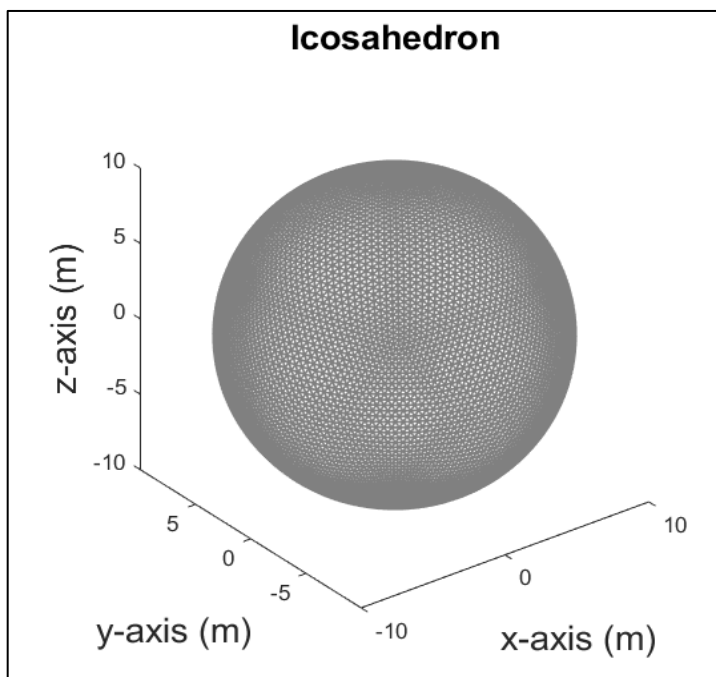


Figure 4.1 Icosahedron for *AutoProject* function validation

frequency is created, high frequency means the object will be more like a ball [14], so the calculated projection created from the rotation in each rotation orientation coordinates must be similar. The rotation orientation coordinates used in this test is 10 (Created from 10x10 matrix) therefore will create 100 orientation of rotation, and the rotation in each orientation will be used theta15 parameter, and the result are in the table below:

Table 4.3 Results of *AutoProject* Function Projections

	1	2	3	4	5	6	7	8	9
2	314.05	314.05	314.05	314.06	314.06	314.06	314.05	314.06	314.05
3	314.05	314.06	314.06	314.05	314.05	314.05	314.05	314.05	314.05
4	314.05	314.06	314.05	314.05	314.05	314.05	314.05	314.04	314.05
5	314.06	314.05	314.05	314.06	314.05	314.05	314.06	314.05	314.06
6	314.06	314.05	314.05	314.06	314.05	314.05	314.06	314.05	314.06
7	314.05	314.06	314.05	314.05	314.05	314.05	314.05	314.04	314.05
8	314.05	314.06	314.06	314.05	314.05	314.05	314.05	314.05	314.05
9	314.05	314.05	314.05	314.06	314.06	314.06	314.05	314.06	314.05

*Row = Rotation Orientation Coordinates Row (Row 1 and 10 are not included because it is the rotation from top and bottom of object and don't interfere with results because the object is projected from the top.

*Column = Rotation Orientation Coordinates Column

Because the total data is 1200 projections, above table only contain data from columns 1-9, there are still 141 columns that have similar results, and it is shown that the *AutoProject* function is valid because the projected data only ranging around 0,2.

After the *AutoProject* function validation, then the validation of why using 100 orientation and rotate with the angle in Theta15 for each orientation will be shown below.

a. First Object: Tetrahedron with edge 10

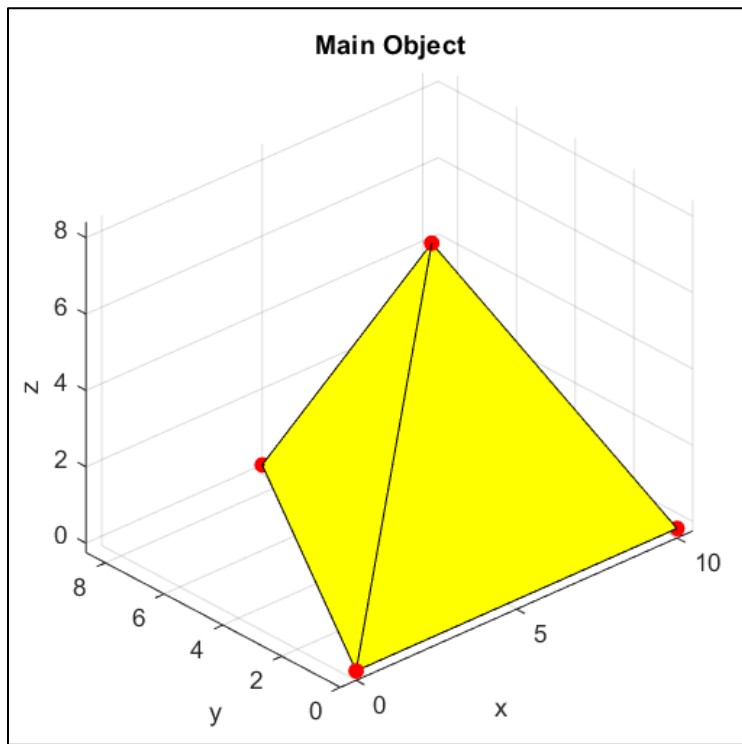


Figure 4.2 Test Object 1: Tetrahedron with edge 10

The first object taken for this validation is a tetrahedron with the edge length of 10, then the overall surface area is 173.21 the result is shown in the table below:

Table 4.4 Object 1 Overall Surface Area Results

Size of Orientation Coordinates	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
4	180.20	170.44	182.40	181.88	173.87	182.50	173.50
5	173.55	167.61	173.47	168.52	170.07	169.95	170.77
6	172.34	172.75	172.63	174.80	174.49	175.24	174.80
7	170.80	173.11	171.66	174.86	174.22	174.60	174.22
8	171.28	171.12	171.49	171.87	171.81	172.13	172.08
9	171.28	170.38	171.36	172.00	172.11	172.21	172.22
10	171.03	170.41	171.20	172.50	172.51	172.40	172.60
11	171.05	171.30	171.41	172.68	172.38	172.53	172.33
12	170.93	170.59	171.36	171.83	171.70	172.09	171.99
13	171.09	170.09	171.37	171.88	171.97	172.15	172.16
14	170.95	170.70	171.29	172.16	172.04	172.17	172.07
15	171.06	170.80	171.34	171.99	171.87	172.13	172.04
16	171.08	170.83	171.37	171.94	171.94	172.09	171.99
17	171.15	170.41	171.34	171.82	171.89	171.96	171.96
18	171.16	170.89	171.32	172.13	172.06	172.19	172.11
19	171.22	170.94	171.37	172.06	171.98	172.18	172.12
20	171.09	170.90	171.29	172.15	172.09	172.17	172.10
21	171.08	170.69	171.29	172.04	172.05	172.14	172.13
22	171.09	170.87	171.27	172.16	172.05	172.26	172.20
23	171.07	170.95	171.27	172.18	172.08	172.26	172.19
24	171.07	170.87	171.27	172.18	172.12	172.18	172.12
25	171.06	170.83	171.25	172.12	172.09	172.23	172.19
30	171.07	170.91	171.24	172.10	172.04	172.19	172.15
40	171.04	170.92	171.20	172.13	172.09	172.20	172.16
50	171.05	170.94	171.19	172.14	172.10	172.19	172.16
100	171.04	170.97	171.16	172.15	172.10	172.19	172.17
200	171.03	170.98	171.14	172.15	172.11	172.19	172.18
300	171.03	170.98	171.14	172.15	172.11	172.19	172.18



: The highest accuracy in the theta group





: The lowest accuracy in the theta group

For more easier reading of the error the result is converted into an error percentage, the error is calculated by comparing the overall area calculation results from Table 4.4 and the overall surface area from mathematical formula.

Table 4.5 Object 1 Error Test Results (Percentage)

Size of Orientation Coordinates	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
4	4.03	1.60	5.31	5.00	0.38	5.36	0.16
5	0.19	3.23	0.15	2.71	1.81	1.88	1.41
6	0.50	0.26	0.34	0.92	0.74	1.17	0.92
7	1.39	0.06	0.89	0.95	0.58	0.80	0.58
8	1.11	1.20	0.99	0.77	0.81	0.62	0.65
9	1.12	1.63	1.07	0.70	0.64	0.58	0.57
10	1.26	1.62	1.16	0.41	0.41	0.47	0.35
11	1.25	1.10	1.04	0.31	0.48	0.39	0.51
12	1.32	1.51	1.07	0.80	0.87	0.64	0.70
13	1.23	1.80	1.07	0.77	0.72	0.61	0.60
14	1.30	1.45	1.11	0.61	0.67	0.60	0.66
15	1.24	1.39	1.08	0.70	0.77	0.62	0.68
16	1.23	1.38	1.06	0.74	0.73	0.65	0.70
17	1.19	1.61	1.08	0.80	0.76	0.72	0.72
18	1.19	1.34	1.09	0.62	0.66	0.59	0.63
19	1.15	1.31	1.06	0.67	0.71	0.59	0.63
20	1.22	1.33	1.11	0.61	0.64	0.60	0.64
21	1.23	1.46	1.11	0.68	0.67	0.62	0.63
22	1.22	1.35	1.12	0.61	0.67	0.55	0.59
23	1.23	1.30	1.12	0.60	0.65	0.55	0.59
24	1.24	1.35	1.12	0.59	0.63	0.60	0.63
25	1.24	1.37	1.13	0.63	0.65	0.57	0.59
30	1.23	1.33	1.14	0.64	0.67	0.59	0.61
40	1.25	1.32	1.16	0.62	0.65	0.58	0.60
50	1.25	1.31	1.17	0.62	0.64	0.59	0.61
100	1.25	1.30	1.18	0.61	0.64	0.59	0.60
200	1.26	1.29	1.19	0.61	0.64	0.59	0.60
300	1.26	1.28	1.20	0.61	0.64	0.59	0.60



 : The highest accuracy in the theta group
 : The lowest accuracy in the theta group

It is shown that the highest and lowest accuracy are spread around rotation orientation coordinates 6-15, therefore in next object only 6-15 will be used, the rotation orientation coordinates 4 and 5 are not included because inconsistency and with that amount the created rotation orientation coordinates are not forming a sphere or ellipsoid therefore is not valid for applying the Cauchy Theorem.

b. Second Object: Tetrahedron with edge 1.414

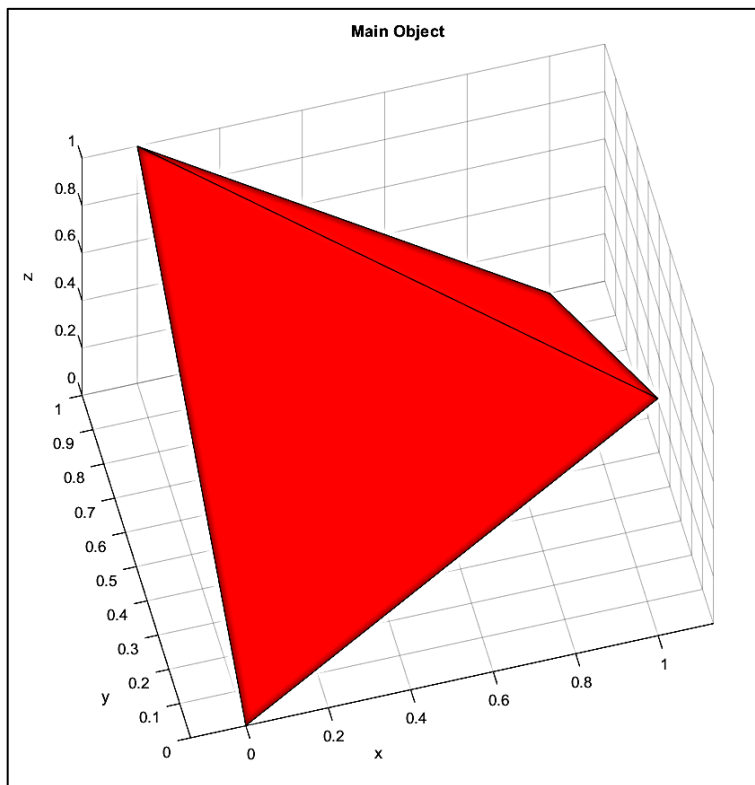


Figure 4.3 Test Object 2: Tetrahedron with edge 1.414

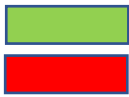
The second object taken for this validation is a tetrahedron with the edge length of 1.414, then the overall surface area is 3.46 the result is shown in the table below:

Table 4.6 Object 2 Overall Surface Area Results

Size of Orientation Coordinates	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
6	3.4204	3.4414	3.4579	3.4053	3.4054	3.4275	3.429
7	3.4075	3.4642	3.4687	3.4265	3.4122	3.4434	3.4379
8	3.4313	3.4617	3.488	3.4283	3.4199	3.4437	3.4408
9	3.4537	3.4391	3.5044	3.4064	3.4173	3.4446	3.4522
10	3.4735	3.4999	3.5132	3.4515	3.4466	3.4628	3.4612
11	3.477	3.4949	3.5195	3.4509	3.4493	3.4723	3.473
12	3.4823	3.5081	3.5246	3.4719	3.4675	3.4882	3.4873
13	3.4777	3.5111	3.5234	3.4804	3.4736	3.4936	3.4913
14	3.4849	3.5084	3.5286	3.4775	3.473	3.4953	3.4944
15	3.4878	3.5146	3.5304	3.4826	3.4768	3.4979	3.4963

Table 4.7 Object 2 Error Test Results (Percentage)

Size of Orientation Coordinates	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
6	1.14	0.54	0.06	1.58	1.58	0.94	0.90
7	1.52	0.12	0.25	0.97	1.38	0.48	0.64
8	0.83	0.05	0.81	0.92	1.16	0.47	0.55
9	0.18	0.60	1.28	1.55	1.23	0.45	0.23
10	0.39	1.15	1.54	0.25	0.39	0.08	0.03
11	0.49	1.01	1.72	0.26	0.31	0.36	0.38
12	0.64	1.39	1.87	0.34	0.22	0.82	0.79
13	0.51	1.48	1.83	0.59	0.39	0.97	0.90
14	0.72	1.40	1.98	0.51	0.38	1.02	0.99
15	0.80	1.58	2.03	0.65	0.49	1.10	1.05



: The highest accuracy in the theta group

: The lowest accuracy in the theta group

c. Third Object: Octahedron

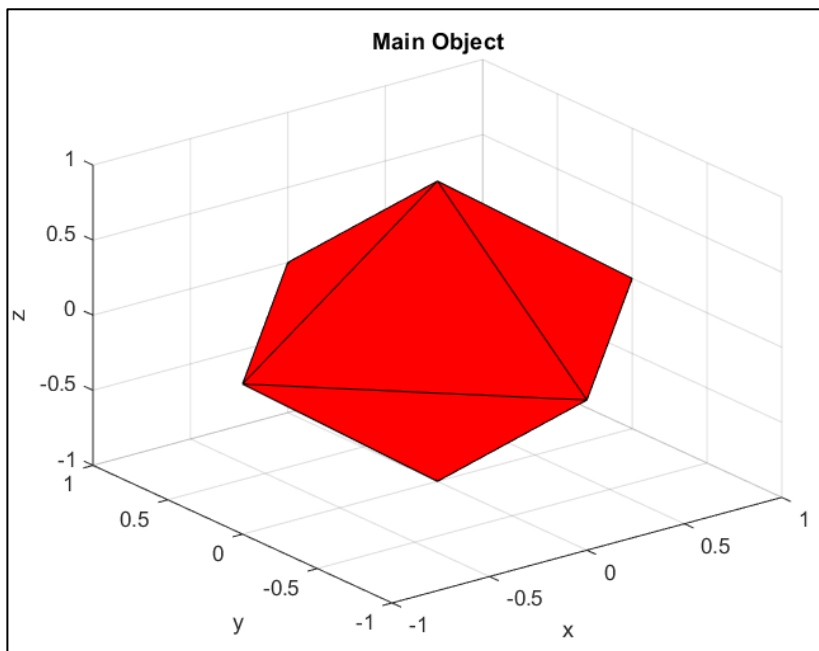


Figure 4.4 Test Object 3: Octahedron

The third object taken for this validation is an octahedron with the edge length of 1.414, then the overall surface area is 6.93 the result is shown in the table below:

Table 4.8 Object 3 Overall Surface Area Results

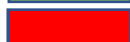
Size of Orientation Coordinates	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
6	6.8408	6.8829	6.9159	6.8106	6.8107	6.855	6.858
7	6.815	6.9284	6.9374	6.8529	6.8244	6.8869	6.8758
8	6.8626	6.9234	6.9761	6.8565	6.8938	6.8874	6.8816
9	6.9075	6.8782	7.0088	6.8129	6.8345	6.8892	6.9044
10	6.9471	6.9999	7.0263	6.903	6.8931	6.9256	6.9225
11	6.954	6.9898	7.039	6.9018	6.8987	6.9446	6.946
12	6.9647	7.0161	7.0491	6.9438	6.9351	6.9764	6.9745
13	6.9553	7.0223	7.0468	6.9608	6.9471	6.9872	6.9826
14	6.9698	7.0168	7.0571	6.955	6.9459	6.9907	6.9888
15	6.9756	7.0291	7.0608	6.9652	6.9537	6.9958	6.9925

Table 4.9 Error Test Results (Percentage)

Size of Orientation Coordinates	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
6	1.29	0.68	0.20	1.72	1.72	1.08	1.04
7	1.66	0.02	0.11	1.11	1.52	0.62	0.78
8	0.97	0.10	0.67	1.06	0.52	0.61	0.70
9	0.32	0.75	1.14	1.69	1.38	0.59	0.37
10	0.25	1.01	1.39	0.39	0.53	0.06	0.11
11	0.35	0.86	1.57	0.41	0.45	0.21	0.23
12	0.50	1.24	1.72	0.20	0.07	0.67	0.64
13	0.37	1.33	1.69	0.44	0.25	0.83	0.76
14	0.57	1.25	1.83	0.36	0.23	0.88	0.85
15	0.66	1.43	1.89	0.51	0.34	0.95	0.90



: The highest accuracy in the theta group



: The lowest accuracy in the theta group

d. Fourth Object: Hexahedron (Cube)

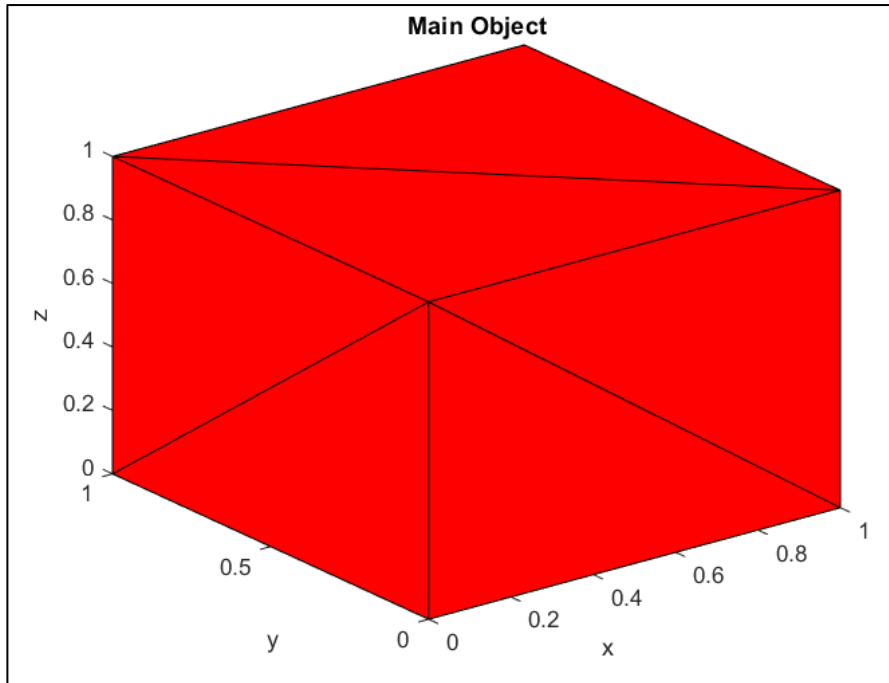


Figure 4.5 Test Object 4: Hexahedron (Cube)

The fourth object taken for this validation is a hexahedron with the edge length of 1, then the overall surface area is 6 the result is shown in the table below:

Table 4.10 Object 4 Overall Surface Area Results

Size of Orientation Coordinates	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
6	6.0223	5.9766	5.9889	6.1134	6.1314	6.1301	6.1396
7	5.971	5.8625	5.9321	5.9944	6.0292	6.0167	6.0344
8	5.9626	5.9331	5.9181	5.99	5.9979	5.9933	5.9972
9	5.9213	5.7398	5.8815	5.8638	5.9222	5.9206	5.9516
10	5.9195	5.8981	5.8742	5.9906	6.003	5.9884	5.994
11	5.8964	5.854	5.8533	5.9396	5.9565	5.9432	5.9513
12	5.8897	5.8721	5.8447	5.9395	5.9474	5.9405	5.9442
13	5.8809	5.7953	5.8331	5.8868	5.9169	5.9075	5.923
14	5.8694	5.855	5.8242	5.9366	5.9461	5.929	5.9329
15	5.859	5.8327	5.814	5.9092	5.9208	5.9131	5.9188

Table 4.11 Object 4 Error Test Results (Percentage)

Size of Orientation Coordinates	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
6	0.37	0.39	0.19	1.89	2.19	2.17	2.33
7	0.48	2.29	1.13	0.09	0.49	0.28	0.57
8	0.62	1.12	1.37	0.17	0.04	0.11	0.05
9	1.31	4.34	1.98	2.27	1.30	1.32	0.81
10	1.34	1.70	2.10	0.16	0.05	0.19	0.10
11	1.73	2.43	2.45	1.01	0.72	0.95	0.81
12	1.84	2.13	2.59	1.01	0.88	0.99	0.93
13	1.99	3.41	2.78	1.89	1.39	1.54	1.28
14	2.18	2.42	2.93	1.06	0.90	1.18	1.12
15	2.35	2.79	3.10	1.51	1.32	1.45	1.35



: The highest accuracy in the theta group

: The lowest accuracy in the theta group

e. Fifth Object: Dodecahedron

The fifth object taken for this validation is a tetrahedron with the edge length of 0.764, then the overall surface area is 12.05 the result is shown in the table below:

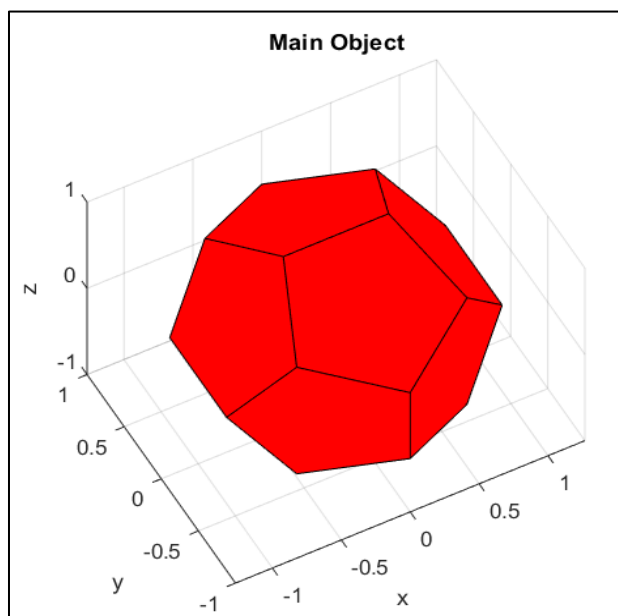


Figure 4.6 Test Object 5: Dodecahedron

Table 4.12 Object 5 Overall Surface Area Results

Size of Orientation Coordinates	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
6	12.187	12.0962	12.1583	12.089	12.0977	12.0737	12.0768
7	12.1632	12.064	12.1444	12.036	12.0523	12.0429	12.0509
8	12.1234	12.0905	12.1216	12.1157	12.1117	12.1061	12.1036
9	12.1025	12.0013	12.111	12.049	12.0677	12.0771	12.0877
10	12.1295	12.1057	12.1115	12.0951	12.0938	12.0701	12.0679
11	12.12	12.0454	12.1015	12.0444	12.0656	12.0547	12.0653
12	12.1016	12.0681	12.0855	12.0717	12.0738	12.0601	12.0603
13	12.0986	12.0086	12.0832	12.0261	12.0462	12.0441	12.0547
14	12.0958	12.0785	12.0767	12.0823	12.0831	12.0703	12.0698
15	12.0933	12.0692	12.0753	12.0727	12.0762	12.064	12.0651

Table 4.13 Object 5 Error Test Results (Percentage)

Size of Orientation Coordinates	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
6	1.14	0.38	0.90	0.32	0.40	0.20	0.22
7	0.94	0.12	0.78	0.12	0.02	0.06	0.01
8	0.61	0.34	0.59	0.55	0.51	0.47	0.44
9	0.44	0.40	0.51	0.01	0.15	0.22	0.31
10	0.66	0.46	0.51	0.37	0.36	0.17	0.15
11	0.58	0.04	0.43	0.05	0.13	0.04	0.13
12	0.43	0.15	0.29	0.18	0.20	0.08	0.09
13	0.40	0.34	0.28	0.20	0.03	0.05	0.04
14	0.38	0.24	0.22	0.27	0.27	0.17	0.16
15	0.36	0.16	0.21	0.19	0.22	0.12	0.13



: The highest accuracy in the theta group



: The lowest accuracy in the theta group

f. Sixth Object: Icosahedron

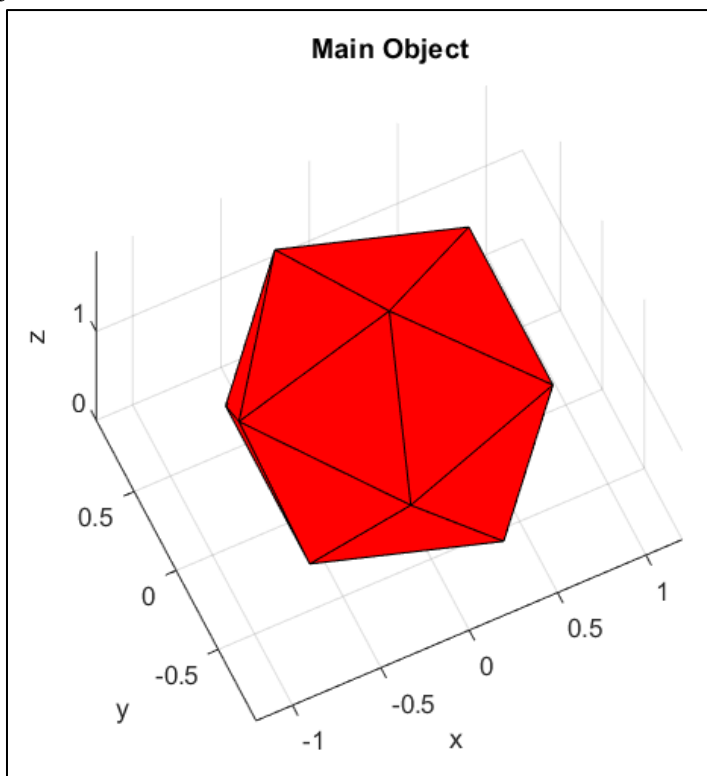


Figure 4.7 Test Object 6: Icosahedron

The sixth object taken for this validation is an icosahedron with the edge length of 1, then the overall surface area is 8.66 the result is shown in the table below:

Table 4.14 Object 6 Overall Surface Area Results

Size of Orientation Coordinates	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
6	8.7694	8.7637	8.7173	8.7063	8.6995	8.668	8.6623
7	8.7468	8.6608	8.6942	8.6478	8.67	8.6366	8.6462
8	8.733	8.7105	8.683	8.7044	8.7098	8.6904	8.692
9	8.708	8.7059	8.665	8.712	8.7098	8.6857	8.6829
10	8.6968	8.6747	8.6595	8.6904	8.6946	8.6784	8.6796
11	8.6868	8.6938	8.6567	8.7025	8.699	8.6821	8.6791
12	8.697	8.6783	8.6572	8.6786	8.6823	8.6598	8.6603
13	8.6911	8.674	8.6534	8.6714	8.673	8.6571	8.6569
14	8.6877	8.661	8.6507	8.6683	8.6725	8.6545	8.6555
15	8.6863	8.6631	8.6486	8.6727	8.6753	8.657	8.6571

Table 4.15 Object 6 Error Test Results (Percentage)

Size of Orientation Coordinates	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
6	1.26	1.20	0.66	0.53	0.46	0.09	0.03
7	1.00	0.01	0.39	0.14	0.12	0.27	0.16
8	0.84	0.58	0.27	0.51	0.58	0.35	0.37
9	0.55	0.53	0.06	0.60	0.58	0.30	0.26
10	0.42	0.17	0.01	0.35	0.40	0.21	0.23
11	0.31	0.39	0.04	0.49	0.45	0.26	0.22
12	0.43	0.21	0.03	0.21	0.26	0.00	0.00
13	0.36	0.16	0.08	0.13	0.15	0.03	0.04
14	0.32	0.01	0.11	0.10	0.14	0.06	0.05
15	0.30	0.04	0.13	0.15	0.18	0.03	0.03



: The highest accuracy in the theta group



: The lowest accuracy in the theta group

4.3 Summary of Overall Surface Area Calculation (Parallel Projection)

After all the test on the 6 objects, the summary of the which theta and rotation orientation coordinates have the best accuracy in each object are shown in table below:

Table 4.16 Error Test Result Summary (Highest Accuracy)

	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
Object 1	6	7	6	11	10	11	10
Object 2	9	8	6	10	12	10	10
Object 3	10	7	7	12	12	10	10
Object 4	6	6	6	7	8	8	8
Object 5	15	11	15	9	7	11	7
Object 6	15	7 & 14	10	14	7	12	12

Table 4.17 Error Test Result Summary (Lowest Accuracy)

	theta2	theta3	theta4	theta4b	theta7	theta8	theta15
Object 1	7	13	300/10*	7	12	6	6
Object 2	7	15	15	6	6	15	15
Object 3	7	15	15	6	6	6	6
Object 4	15	9	15	9	6	6	6
Object 5	6	10	6	8	8	8	8
Object 6	6	6	6	9	8 & 9	8	8

**In theta4 the lowest is rotation orientation coordinates 300x300 but because the rest of the test not using 300 so the second lowest is taken, which is 10.*

Based on the summary result of the accuracy, in theta2-4 the average highest accuracy is in rotation orientation coordinates 6 but also in the same time the lowest accuracy is rotation orientation coordinates 6, but in theta4b-15 the average highest accuracy is 10 and the lowest accuracy is in 6.

There is some different result in Object 1 and 2 even the object is alike only different in the position and length of the edge, after several test the cause of this, is the position of tetrahedron, but after all the results is almost the same in entire rotation orientation coordinates and theta.

After all of the test the error amount only ranging between 0.01% to 2.03% except in the hexahedron or cube the highest error is at 4.34% in theta3 rotation orientation coordinates 9. Therefore, it can be concluded that the variable that always keep the consistency of high accuracy is at rotation orientation coordinates 10 and using theta15.

4.4 Overall Area Surface Calculation from Central Projection

Object parameters used for central projection is the same with parallel projection, and the overall surface area calculation is also similar but there will be scale correction due to distortion created in central projection. If the parallel projection can be simulated by simply remove one of the axes, in central projection the object must be transformed by using *viewmtx* function, a built-in function that computes a view the transformation matrix for central projection simulation.

The *viewmtx* function computes a 4-by-4 orthographic or perspective transformation matrix that projects four-dimensional homogeneous vectors onto a

two-dimensional view surface [7], there are 4 parameters in this function, there are:

- AZ = Azimuth (Horizontal Rotation)
- EL = Elevation (Vertical Rotation)
- AOV/PHI = Angle of View
- XC = Position of camera target point or look-at

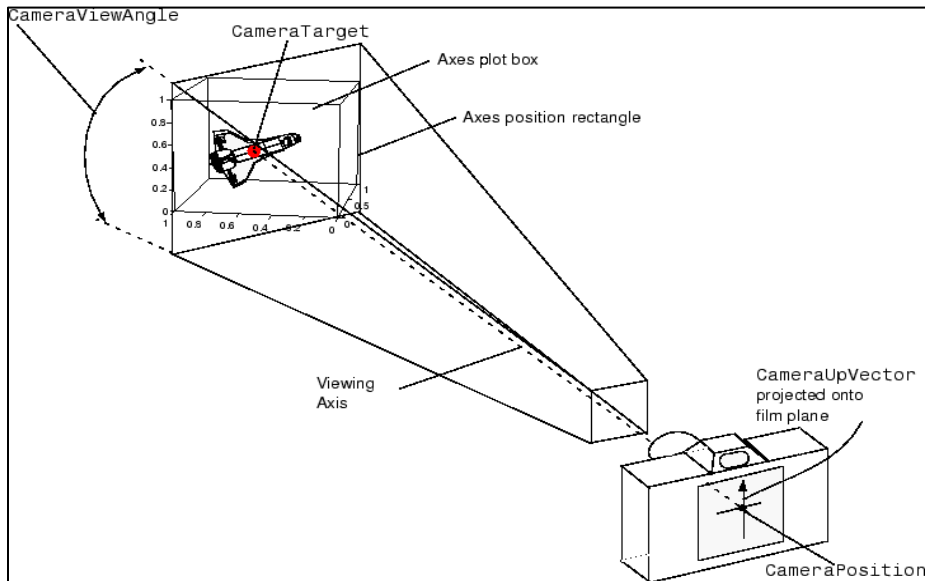


Figure 4.8 Camera Graphics Terminology in Matlab

(Source: mathworks.com)

AZ and EL will determine the camera viewing position, if AZ is 0 and EL is 90 then the camera is in the top of the object if AZ is 0 and EL is 0 then the camera will be in the front of the object.

For AOV is the camera view angle as shown in Figure 4.8, and is not only determine the horizontal degrees but also the vertical degrees of view angle.

Table 4.18 Angle of View Value Representation

Phi (AOV)	Description
0	Orthographic Projection
10	Similar to Telephoto Lens
25	Similar to Normal Lens
40-60	Similar to Human Eyes
60	Similar to Wide-Angle Lens

Last is XC, as shown in Figure 4.8 XC is camera target, that must be stay at the center of the view plane if is it not in the center of the view plane the projection will be getting significant distortion. And in MATLAB the view plane are places in the origin of object coordinate.

An example of implementation is if AZ and EL are 0 and AOV is 10 if there are a line that representant in x, y, z point and have value of 0,0,1 and 1,0,1 then the XC must be in 0.5, -1, 0.5 the image will get less distorted because the camera is in front of the object and camera target also in front of the object shown by the -1 value of y axis, the more less the amount the more further away from the object and 0.5 in both x and z axis are showing that the target is pointing in the middle of the object.

In this research the *viewmtx* parameters are:

- AZ = 0
- EL = 0
- AOV = 10
- TargetPoint (XC) = described below

```
DphClbrn = 10;
ClbrnMAX = max (Rpoints) / 2;
ClbrnMIN = min (Rpoints) / 2;
Clbrn = ClbrnMAX+ClbrnMIN;
```

```
ClbrnX = Clbrn(1,1);
ClbrnY = ClbrnMIN(1,2)-DphClbrn;
ClbrnZ = Clbrn(1,3);
```

The Rpoints are any by 3 matrix that represent the points of an object, for better understanding see the figure below:

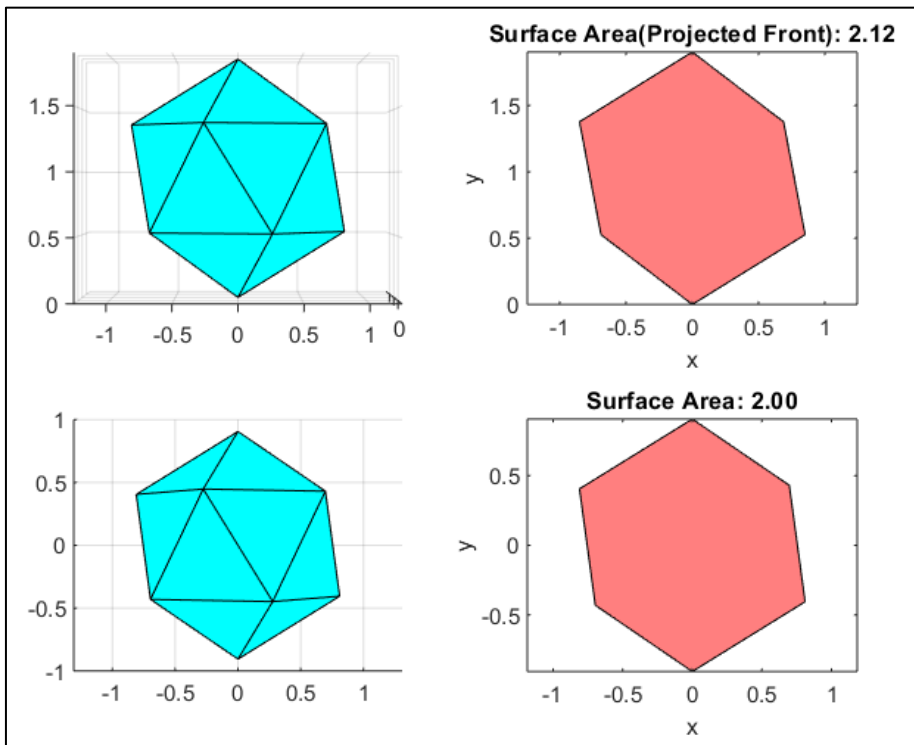


Figure 4.9 Central Projection Example, top left is main object seeing from front or Y direction and top right is the projected area from the front, bottom left is simulated central projection with AZ and EL 0, AOV 10, and camera target at the middle of the object,

If in parallel projection *AutoProject* function is used for simulate the object rotation and projection, in central projection *AutoProjectCentral* is used (Appendix C). It rotates the object in the same parameters using 100 orientation vector and theta15 for rotation parameter, but this time because the object is seeing from the front all of the orientation is used and will create 1500 projections from 10 rows multiplied by 10 columns for orientation vector and multiplied again by 15 for rotation parameter in each orientation. And also, in *AutoProjectCentral* the *viewmtx* function will be used for simulating the central projection.

For overall surface area calculation, the result will be multiplied by certain scale value, that will correct the result due to distortion in central projection and make the result close to parallel projection. The object that created from chapter 3.2 with 25 points ranging from 0-10, *rng* value of 0 and have total surface area of 296.22 in parallel projection. In central projection with *viewmtx* parameters as described above will result 296.78 with scale value of 6.03 and give different of 0.19% from the parallel projection value. The scale value is depending on the central projection parameters or in this the object maximum length/height and *viewmtx* parameters.

4.5 Validation and Error Testing of Developed Calculation Method (Central Projection)

The object for validation and error testing in this central projection will be similar to parallel projection by using Platonic Solids objects, the testing parameters are written below:

- Object Scale = 1 (Original), 5, and 10
- Angle of View = 10 and 25
- Target (Depth) Calibration = 0 (Original), -5, and -10

Object scale is value that scaling the object, if 1 then the object is in original value, angle of view 10 is similar to telephoto and 25 is similar to normal lens in camera (Table 4.18), and depth calibration is a value that reduce the y axis value in the camera position because the object is projected from the front or at the direction of y axis, the more less the amount the more further away the camera and will resulting object to be smaller but it will reduce distortion resulting more accurate results.

Before the testing begin, this are object overall area surface when object at original scale, 5 times scaled and 10 times scaled, area is calculated with mathematical formula:

Table 4.19 Object Overall Area Surface at Original

No	Object	Formula	a	Overall Area Surface
1	Tetrahedron	$\sqrt{3}a^2$	1	1.73
2	Tetrahedron	$\sqrt{3}a^2$	1.414	3.46
3	Octahedron	$2\sqrt{3}a^2$	1.414	6.93
4	Hexahedron (Cube)	$6a^2$	1	6
5	Dodecahedron	$3\sqrt{25 + 10\sqrt{5}}a^2$	0.764	12.05
6	Icosahedron	$5\sqrt{3}a^2$	1	8.66

Table 4.20 Object Overall Area Surface at 5 Times Scaled

No	Object	Formula	$a*5$	Overall Area Surface
1	Tetrahedron	$\sqrt{3}a^2$	5	43.3
2	Tetrahedron	$\sqrt{3}a^2$	7.07	86.58
3	Octahedron	$2\sqrt{3}a^2$	7.07	173.15
4	Hexahedron (Cube)	$6a^2$	5	150
5	Dodecahedron	$3\sqrt{25 + 10\sqrt{5}a^2}$	3.82	301.27
6	Icosahedron	$5\sqrt{3}a^2$	5	216.51

Table 4.21 Object Overall Area Surface at 10 Times Scaled

No	Object	Formula	$a*10$	Overall Area Surface
1	Tetrahedron	$\sqrt{3}a^2$	10	173.21
2	Tetrahedron	$\sqrt{3}a^2$	14.14	346.31
3	Octahedron	$2\sqrt{3}a^2$	14.14	692.61
4	Hexahedron (Cube)	$6a^2$	10	600
5	Dodecahedron	$3\sqrt{25 + 10\sqrt{5}a^2}$	7.64	1205.08
6	Icosahedron	$5\sqrt{3}a^2$	10	866.03

a. Angle of View/Phi at 10

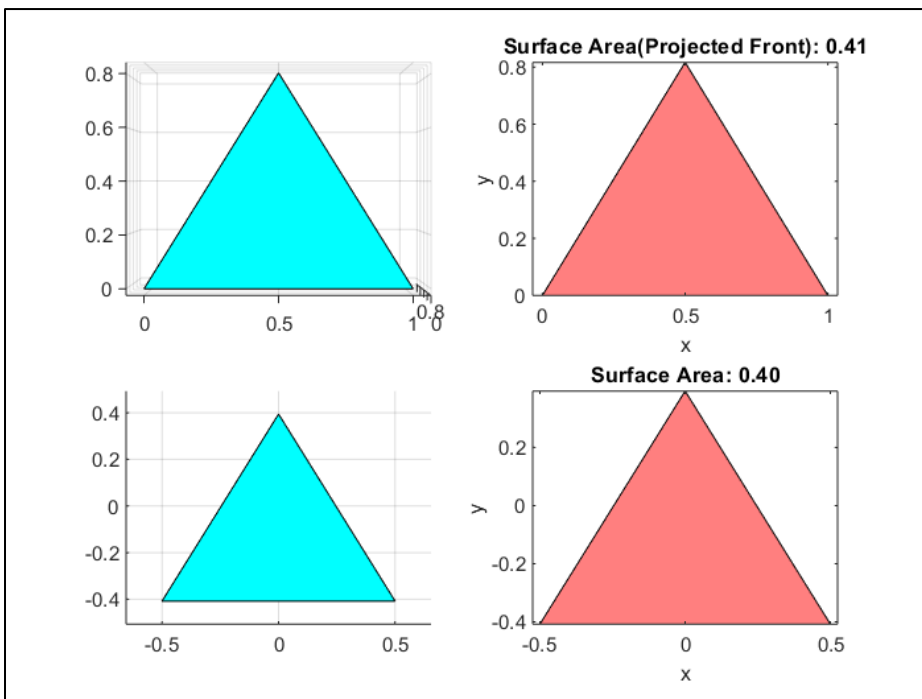


Figure 4.10 Central Projection Example of First Object with Original Scale, Depth Calibration of 10 and AOV at 10

The figure that created for example are using parameters described from Chapter 4.4, below is the result of scale needed in certain depth that the error (different value) from mathematical formula must be below 5%. After the scale is obtained, it is averaged and recalculate the object error using averaged scale.

Table 4.22 Scale Testing, at AOV 10, object original scale, depth at original, -5, and -10

Depth Original, Object Original Scale			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	1.03	4.07	1	2.21
2	1.04	4.18	2	3.26
3	1.05	4.72	3	4.72
4	1.06	4.14	4	5.05
5	1.06	4.22	5	5.12
6	1.05	4.12	6	4.12
Mean	1.05	4.24	Mean	4.08
Depth -5 , Object Original Scale			Scale Mean is Used	

Object	Scale	Error	Object	Error
1	2.63	4.89	1	4.17
2	2.63	5	2	4.28
3	2.66	4.84	3	5.2
4	2.68	4.93	4	5.99
5	2.66	4.84	5	5.2
6	2.64	4.93	6	4.57
Mean	2.65	4.91	Mean	4.90
Depth -10 , Object Original Scale			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	4.98	4.96	1	4.77
2	4.96	5	2	4.42
3	5	4.92	3	5.11
4	5.05	4.97	4	6.1
5	4.99	4.93	5	4.93
6	4.97	4.84	6	4.45
Mean	4.99	4.94	Mean	4.96

Table 4.23 Scale Testing, at AOV 10, object 5 times scaled, depth at original, -5, and -10

Depth Original, Object 5 Times Scaled			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	1.13	4.48	1	1.26
2	1.18	4.5	2	1.26
3	1.28	4.45	3	8.93
4	1.2	4.36	4	2.76
5	1.3	4.34	5	10.22
6	1.21	4.7	6	3.92
Mean	1.22	4.47	Mean	4.73
Depth -5 , Object 5 Times Scaled			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	2.87	4.79	1	1.18
2	2.99	4.99	2	3.09
3	3.14	4.79	3	7.52
4	3.05	4.84	4	4.84

5	3.2	4.73	5	9.19
6	3.07	4.96	6	5.58
Mean	3.05	4.85	Mean	5.23
Depth -10 , Object 5 Times Scaled			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	5.35	4.95	1	0.15
2	5.54	4.86	2	3.48
3	5.72	4.91	3	6.58
4	5.64	4.94	4	5.28
5	5.81	4.88	5	7.99
6	5.65	4.98	6	5.48
Mean	5.62	4.92	Mean	4.83

Table 4.24 Scale Testing, at AOV 10, object 10 times scaled, depth at original, -5, and -10

Depth Original, Object 10 Times Scaled			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	1.03	4.14	1	28.34
2	0.76	4.4	2	3.14
3	1.23	4.78	3	40.39
4	0.54	4.43	4	36.28
5	1.03	4.07	5	28.29
6	0.04	1.93	6	1788
Mean	0.77	3.96	Mean	320.74
Depth -5 , Object 10 Times Scaled			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	2.95	4.94	1	1.51
2	3.03	4.81	2	1.04
3	3.39	4.95	3	11.68
4	3.07	4.75	4	2.27
5	3.38	4.84	5	11.31
6	3.08	4.98	6	2.82
Mean	3.15	4.88	Mean	5.11
Depth -10 , Object 10 Times Scaled			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	5.61	4.84	1	2.29
2	5.87	4.93	2	2.34

3	6.3	4.88	3	8.95
4	5.97	4.89	4	3.93
5	6.41	4.89	5	10.53
6	6.04	4.93	6	5.09
Mean	6.03	4.89	Mean	5.52

b. Angle of View/Phi at 25

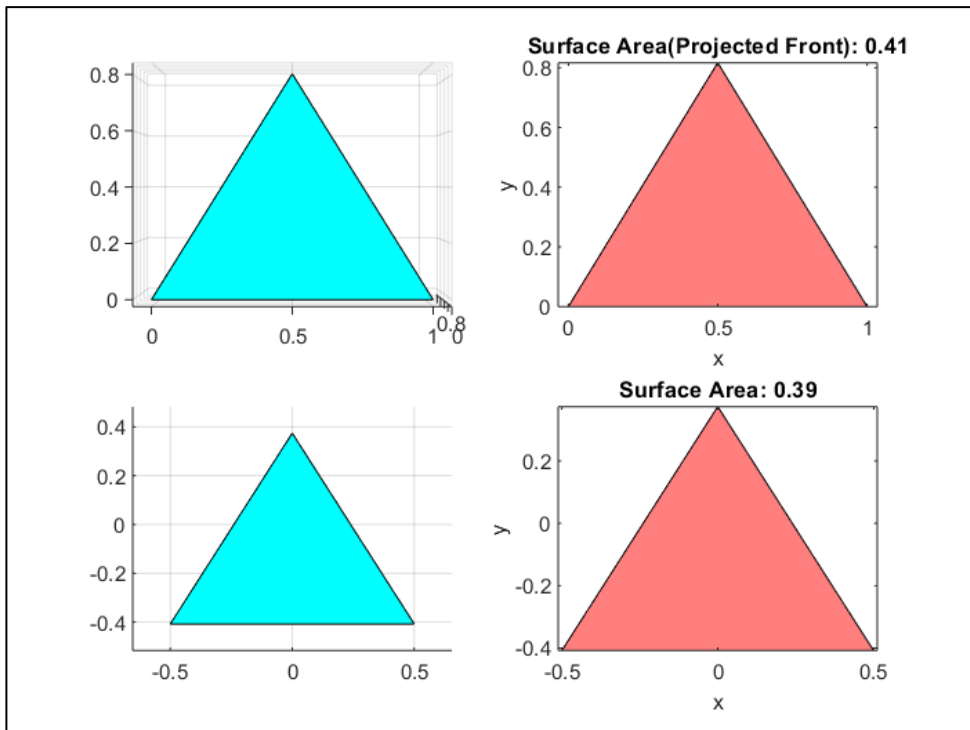


Figure 4.11 Central Projection Example of First Object with Original Scale, Depth Calibration of 10 and AOV at 25

Using the same method like angle of view or phi at 10, and this is the results:

Table 4.25 Scale Testing, at AOV 25, object original scale, depth at original, -5, and -10

Depth Original, Object Original Scale			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	1.08	4.68	1	0.61
2	1.12	4.83	2	3.13
3	1.17	4.4	3	6.85
4	1.15	4.18	4	5.02

5	1.19	4.32	5	8.34
6	1.15	4.6	6	5.43
Mean	1.14	4.50	Mean	4.90
Depth -5 , Object Original Scale			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	6.73	4.87	1	3.03
2	6.8	4.89	2	4.05
3	6.9	4.93	3	5.48
4	6.92	5	4	5.82
5	6.93	4.9	5	5.86
6	6.86	4.88	6	4.88
Mean	6.86	4.91	Mean	4.85
Depth -10 , Object Original Scale			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	17.15	4.96	1	4.23
2	17.16	4.99	2	4.33
3	17.33	4.97	3	5.24
4	17.48	4.95	4	6.04
5	17.33	4.97	5	5.24
6	17.22	4.95	6	4.62
Mean	17.28	4.97	Mean	4.95

Table 4.26 Scale Testing, at AOV 25, object 5 times scaled, depth at original, -5, and -10

Depth Original, Object 5 Times Scaled			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	0.82	4.41	1	45.21
2	0.11	2.35	2	317
3	1.07	4.19	3	57.91
4	0.08	4.93	4	457
5	0.66	4.99	5	32.2
6	0.07	4.91	6	502.3
Mean	0.47	4.30	Mean	235.27
Depth -5 , Object 5 Times Scaled			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	7.41	4.95	1	2.36
2	7.74	4.99	2	2.04
3	8.38	4.98	3	9.52

4	7.86	5	4	3.55
5	8.51	4.93	5	10.85
6	7.96	4.94	6	4.7
Mean	7.98	4.97	Mean	5.50
Depth -10 , Object 5 Times Scaled			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	18.68	5	1	1.11
2	19.5	4.99	2	3.14
3	20.43	4.98	3	7.54
4	19.86	5	4	4.9
5	20.81	4.96	5	9.21
6	20.02	4.96	6	5.62
Mean	19.88	4.98	Mean	5.25

Table 4.27 Scale Testing, at AOV 25, object 5 times scaled, depth at original, -5, and -10

Depth Original, Object 10 Times Scaled			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	0.02	4.88	1	139.3
2	0.14	4.93	2	72.08
3	0.01	4.9	3	188.18
4	0.001	0.34	4	2576
5	0.01	4.66	5	293
6	0.04	4.87	6	10.46
Mean	0.04	4.10	Mean	546.50
Depth -5 , Object 10 Times Scaled			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	6.78	4.92	1	24.84
2	5.16	4.93	2	1.25
3	8.13	4.96	3	37.34
4	3.59	4.74	4	42.23
5	6.83	4.87	5	25.34
6	1.67	4.7	6	205.88
Mean	5.36	4.85	Mean	56.15
Depth -10 , Object 10 Times Scaled			Scale Mean is Used	
Object	Scale	Error	Object	Error
1	19.25	4.98	1	1.49
2	19.77	4.97	2	1.17

3	22.11	4.99	3	11.65
4	20.02	4.98	4	2.42
5	22.05	4.96	5	11.38
6	20.14	5	6	3.01
Mean	20.56	4.98	Mean	5.19

c. Summary Result of Scale Estimation

Table 4.28 Summary of Scale Estimation with Angle of View/Phi 10

Object Scale	Depth	Mean Scale	Mean Error	Mean Tested Error
Original Scale	Depth Original	1.05	4.24	4.08
	Depth -5	2.65	4.91	4.90
	Depth -10	4.99	4.94	4.96
5 Times Scaled	Depth Original	1.22	4.47	4.73
	Depth -5	3.05	4.85	5.23
	Depth -10	5.62	4.92	4.83
10 Times Scaled	Depth Original	0.77	3.96	320.74
	Depth -5	3.15	4.88	5.11
	Depth -10	6.03	4.89	5.52

Table 4.29 Summary of Scale Estimation with Angle of View/Phi 25

Object Scale	Depth	Mean Scale	Mean Error	Mean Tested Error
Original Scale	Depth Original	1.14	4.50	4.90
	Depth -5	6.86	4.91	4.85
	Depth -10	17.28	4.97	4.95
5 Times Scaled	Depth Original	0.47	4.30	235.27
	Depth -5	7.98	4.97	5.50
	Depth -10	19.88	4.98	5.25
10 Times Scaled	Depth Original	0.04	4.10	546.50
	Depth -5	5.36	4.85	56.15
	Depth -10	20.56	4.98	5.19

In summary, if the mean tested error value is not much different with mean error then the scale is valid for that parameters, and if the result are showing large different like in angle of view 10, depth original, object 10 times scaled and at angle of view 25, depth original, object 5 times scaled the scale is not valid to be used.

4.6 Summary of Overall Surface Area Calculation (Central Projection)

In this part the scale estimation result will be used in main object that defined in this research, and because the object overall surface area already calculated using parallel projection, the result can be compared in the table below.

Table 4.30 Scale Estimation Tested on Main Object with Angle of View/Phi 10

Object Scale	Depth	Mean Scale	Original Object Error
Original Scale	Depth Original	1.05	2.34
	Depth -5	2.65	3.8
	Depth -10	4.99	4.16
5 Times Scaled	Depth Original	1.22	1.09
	Depth -5	3.05	0.63
	Depth -10	5.62	1.43
10 Times Scaled	Depth Original	0.77	19.38
	Depth -5	3.15	3.15
	Depth -10	6.03	0.19

Table 4.31 Scale Estimation Tested on Main Object with Angle of View/Phi 25

Object Scale	Depth	Mean Scale	Original Object Error
Original Scale	Depth Original	1.14	1.01
	Depth -5	6.86	3.15
	Depth -10	17.28	3.86
5 Times Scaled	Depth Original	0.47	12496.97
	Depth -5	7.98	0.32
	Depth -10	19.88	0.7
10 Times Scaled	Depth Original	0.04	143.57
	Depth -5	5.36	16.33
	Depth -10	20.56	0.14

The green mark is shown the lowest error at certain scale, the error are ranging from 0.14 to 2.34% and in this result is proved that if the object is getting larger the depth needs to be lesser or getting the target further away from the object and the depth reduction is linear to object scale, the scale is having a possibility to create a pattern.

In Angle of View 10, the scale range between object at original and 5 times scaled is 2 and at 10 times scaled is 3, the estimation scale if the object is 15 times scaled and depth at -15 the scale is 10 by adding 4 from scale 6 at object 10 times scaled and depth at -10.

In Angle of View 25, the scale range between object at original and 5 times scaled is 6.84 and at 10 times scaled is 12.58, the estimation scale if the object is 15 times scaled and depth at -15 the scale is 39.06 by adding 18.5 from scale 20.56 at object 10 times scaled and depth at -10.

“This page intentionally left blank”

CHAPTER V CONCLUSION AND OUTLOOK

This section summarizes all the important findings, conclusions of the study, and recommendations derived from its results and conclusions. The research was to make a measuring and calculation method for determining the size of overall area surface of a convex machine component with projections. The objective was to estimate the minimum number of projections needed for calculating overall area surface, make a calculation method for processing projections data to get the overall area surfaces with minimum error, and also to make distortion corrections if the projection is a central projection.

Based on the research that has been conducted, carry out simulation in MATLAB, and error testing of the developed calculation method, the conclusions of this bachelor thesis are listed below:

- Getting a large amount of projections data doesn't always make the calculation more accurate, only some data at certain axis rotation are important. And in this research the projections data is 1200 at parallel projection and 1500 at central projection that proved to have minimum amount of error.
- To get the overall surface area of a convex object the Cauchy theorem is helpful but the object must be projected in orientations that created an ellipsoid type of rotation.
- Parallel projection is the most ideal projection for calculating area because there is no distortion of object, but this projection is cannot be done in the real world, therefore central projection is used.
- In central projection to minimize the distortion the target needs to be positioned at the right position, also the angle of view is important because to keep distortion at minimum and the object must be fitted in radius angle of view, the nearer the object into the camera the wider angle of view is needed.
- The developed calculation method for parallel projection have an error of 0.01 to 4.34% and for central projections ranging between 0.14 to 2.34%

After simulations and testing of several developed calculation, some topics emerged that later can be submitted as a future idea to prepare another research. The suggestions that can be given for the development of this research include:

- The amount of projections data still can be reduced by reducing the rotations, the random rotation angles that outside of special degree are still not tested.
- There is still no method for correcting the distorted object in central projection, the distortion can only be minimized, there are method for correcting the distortion but the input is already a 2D image not matrix projection from an object. There is a possibility to make a correction method by comparing the result with a matrix of the object that are not distorted.
- There is a possibility for making a scale value estimation based on object length/width, angle of view, and camera target/look-at for making overall surface area corrections in central projection.

“This page intentionally left blank”

REFERENCES

- [1] Cauchy, A.-L. (1841). *Note on various théorèmes à la rectification des courbes et à la quadrature des surfaces*, *Compte Rendu Acad. Sci.*, Vol. 13.
- [2] Douillet, N. (2018, January 15). Any 3D rotation version 1.4.0.0, Matlab Central File Exchange.
- [3] Foley, J. (1997). *Computer Graphics, Chapter 6*. Boston: Addison-Wesley.
- [4] Hazewinkel, & (Ed.), M. (1994). *Encyclopedia of Mathematics*. Netherlands: Springer Science+Business Media B.V.
- [5] Hess, J. P. (2017, January 24). *Digging Deeper Into Perspective Distortion*. Retrieved from Filmmaker IQ: <https://filmmakeriq.com/lessons/digging-deeper-perspective-distortion/>
- [6] Kiran, A. G., & Murali, S. (2013). Automatic Rectification of Perspective Distortion From a Single Image Using Plane Homography. *International Journal on Computational Sciences & Applications (IJSCA)*.
- [7] Mathworks Inc. (2006). *viewmtx*. Retrieved from Matlab Documentation: <https://www.mathworks.com/help/matlab/ref/viewmtx.html>
- [8] Maynard, P. (2005). Drawing distinctions: the varieties of graphic expression, P 22. Cornell University Press.
- [9] Prabhune, O., Sabale, P., Sonawane, D. N., & Prabhune, C. (2017). Image Processing and Matrices. *2017 International Conference on Data Management, Analytics and Innovation (ICDMAI)*. Pune, India: IEEE.
- [10] Sabliov, C. M., Boldor, D., Keener, K. M., & Farkas, B. E. (2002). Image Processing Method to Determine Surface Area and Volume of Axi-Symmetric Agricultural Products. *International Journal of Food Properties*.
- [11] Safwat, M. A. (1971). Theoretical Prediction of Volume, Surface Area, and Center of Gravity for Agricultural Products. *Transactions of the ASAE*.
- [12] The Mathworks Inc. (2019). MATLAB version 9.6.0.1150989 (R2019a) Update 4. Natick, Massachusetts.
- [13] Weisstein, E. W. (n.d.). *Projection*. Retrieved from A Wolfram Web Resource: <http://mathworld.wolfram.com/Projection.html>
- [14] Zechmann, E. (2008, March 12). Make Icosahedron version 1.0.0.0, MATLAB Central File Exchange.

“This page intentionally left blank”

APPENDIX

Appendix A: Main Script Function

```

clear; clc;
%% Create Object Parameters
Data = 25; % Set the Number of Random Data
Rmin = 0; % Set minimum data value
Rmax = 10; % Set maximum data value
rng(0) % To Variate and Hold Random Data Generated in each Run
Rpoints = (Rmax-Rmin)*rand(Data,3) + Rmin; % Generate Random
% 3-D points

Object1 = convhull(Rpoints(:,1),Rpoints(:,2),Rpoints(:,3));

% Plot the scattered points:
figure(1);
subplot(2,2,1);
scatter3(Rpoints(:,1),Rpoints(:,2),Rpoints(:,3),'');

axis equal;
title('Interior points');
xlabel('x'); ylabel('y'); zlabel('z');

% Plot the 3-D convex hull:
subplot(2,2,2);
trisurf(Object1,Rpoints(:,1),Rpoints(:,2),Rpoints(:,3),'FaceColor','c');

axis equal;
title('Main Object (From Top/Z Axis)');
view(2);
xlabel('x'); ylabel('y'); zlabel('z');

%% Using Polyarea for calculating Projected Surface Area from Z
% Axis:
Poly_x= Rpoints(:,1); %take the x axis
Poly_y= Rpoints(:,2); %take the y axis

ParProjHull = convhull(Poly_x,Poly_y);
ParArea = polyarea(Poly_x(ParProjHull),Poly_y(ParProjHull));

subplot(2,2,3);
plot(Poly_x,Poly_y, 'k' );
fill(Poly_x(ParProjHull),Poly_y(ParProjHull), 'r','facealpha', 0.5);

axis image;
title(sprintf('Surface Area(Projected Above): %.2f', ParArea));

```



```

xlabel('x'); ylabel('y');

%% Define The Rotation Orientation Coordinates
theta = linspace(0, 2*pi,10);
phi = linspace(-pi/2, pi/2,10);
[theta, phi] = meshgrid(theta,phi);
rho = 5;
[SphrX,SphrY,SphrZ] = sph2cart(theta,phi,rho);

subplot(2,2,4);
mesh(SphrX,SphrY,SphrZ);

axis equal;
title('Rotation Orientation Coordinates');
xlabel('x'); ylabel('y'); zlabel('z');

sgtitle('Main Data 1')

%% Projecting from Rotation Orientation Coordinates (Parallel
Projection)
AreaProjectPar = [];
for i = 2:9 % The top and the bottom of orientation coordinates is not
           % included because the object will be projected from Z axis,
           % therefore the top and bottom will generate the same amount
           % of area surfaces
    for j = 1:10
        u = [ SphrX(i,j) SphrY(i,j) SphrZ(i,j)]';
        AreaProjectPar(i,:,j) = AutoProject(Rpoints,u);
    end
end
% The resulting first row of AreaProjectPar is 0 due to the first run of
% AreaProjectPar is (2,:,1) therefore the first row is skipped and
% resulting 0 in result

ParProjReshape = reshape(AreaProjectPar,[9 150]);

% Optional Code for Writing the Results in Excel for Better Reading
% xlswrite('AreaProjecPar.xlsx',ParProjReshape);

ParProj = ParProjReshape(2:end,1:end);
Parprojmean = mean(ParProj,'all');
OverallSurfaceAreaPar = Parprojmean*4; % multiples 4 by Cauchy
Theorem

%% Object Re-Plotting for Central Projection

```

```

% The object will be projected into the Y axis instead of Z axis in
% Central Projection, so the object is re-plotted just to have a better
% view from the front and the figure view is changed from orthographic
% to perspective

figure(2);
subplot(2,2,1);
trisurf(Object1,Rpoints(:,1),Rpoints(:,2),Rpoints(:,3),'FaceColor','c');

axis equal; camproj('perspective');
title('Main Object (From Front/Y Axis)');
view(0,0); xlabel('x'); ylabel('y'); zlabel('z');

%% Using Polyarea for calculating Projected Surface Area from Y Axis:
Poly_x2=Rpoints(:,1); %take the X axis
Poly_y2=Rpoints(:,3); %take the Z axis

ParProjHull2 = convhull(Poly_x2,Poly_y2);
ParArea2 = polyarea(Poly_x2(ParProjHull2),Poly_y2(ParProjHull2));

subplot(2,2,2);
plot(Poly_x2,Poly_y2, 'k');
fill(Poly_x2(ParProjHull2),Poly_y2(ParProjHull2), 'r','facealpha', 0.5);

axis image;
title(sprintf('Surface Area(Projected Front): %.2f, ParArea2));
xlabel('x'); ylabel('y');

%% Central Projection Parameters

AZ = 0; % Azimuth
EL = 0; % Elevation
% AZ and EL 0 means seeing the object from Y direction/Front

AOV = 10; % Angle of View, the higher angle of view will make the
% need of Camera Target getting further away from the object, also %high
% angle of view will make the object more distorted.

DphClbrn = 10; % Calibration for Camera Target,
% the higher the number, the further away from object

% Calibration Parameters
ClbrnMAX = max(Rpoints)/2;
ClbrnMIN = min(Rpoints)/2;
Clbrn = ClbrnMAX+ClbrnMIN;

```

```

ClbrnX = Clbrn(1,1);
ClbrnY = ClbrnMIN(1,2)-DphClbrn;
ClbrnZ = Clbrn(1,3);

TargetPoint = [ClbrnX,ClbrnY,ClbrnZ]; % Camera Target/Look-At

% Central Projection for Example Figure
CenPar = viewmtx(AZ,EL,AOV,TargetPoint);
% Transformation Matrix to Simulate the Central Projection

SZ = size(Rpoints,1);
Rpoints4d = [Rpoints(:,1),Rpoints(:,2),Rpoints(:,3),ones(SZ,1)'];
% Transform points into homogenous transformation matrix

Rpoints2d = CenPar*Rpoints4d; % Apply the Transformation Matrix

CenProjX(:) = Rpoints2d(1,:)./Rpoints2d(4,:);
CenProjY(:) = Rpoints2d(2,:)./Rpoints2d(4,:);
CenProjZ(:) = Rpoints2d(3,:)./Rpoints2d(4,:);

NewRpoints = [CenProjX;CenProjY;CenProjZ]';
% Transform points back to original from homogenous transformation

Object2 = convhull(NewRpoints(:,1),NewRpoints(:,2),NewRpoints(:,3));

subplot(2,2,3);
trisurf(Object2,NewRpoints(:,1),NewRpoints(:,2),...
        NewRpoints(:,3),'FaceColor','c');

title('Central Projection (From Front)'); view(2);
axis equal;
xlabel('x'); ylabel('y'); zlabel('z');

%% Using Polyarea for calculating Projected Surface Area:

CenProjHull = convhull(CenProjX,CenProjY);
CenArea = polyarea(CenProjX(CenProjHull),CenProjY(CenProjHull));

subplot(2,2,4);
plot(CenProjX,CenProjY, 'k' );
fill(CenProjX(CenProjHull),CenProjY(CenProjHull), 'r','facealpha', 0.5);

axis equal;
title(sprintf('Surface Area: %.2f', CenArea));
xlabel('x'); ylabel('y');

```

```

sgtitle('Main Data 2')

%% Projecting from Rotation Orientation Coordinates
(Central/Perspective Projection)

AreaProjectCen = [];
for i = 1:10 % The top and the bottom of orientation coordinates is now
    % included because the object will be projected from Y axis,
    % therefore the top and bottom will generate the different
    % amount of area surfaces
    for j = 1:10
        u = [ SphrX(i,j) SphrY(i,j) SphrZ(i,j)]';
        AreaProjectCen(i,:,j) = AutoProjectCentral(Rpoints,u,AZ,EL,...
            AOV,DphClbrn);
    end
end

CenProjReshape = reshape(AreaProjectCen,[10 150]);
% Optional Code for Writing the Results in Excel for Better Reading
% xlswrite('AreaProjecCen.xlsx',CenProjReshape);

CenProjMean = mean(CenProjReshape,'all');

Scale = 6.03; % Depends on object and central projection parameters
OverallSurfaceArea2 = (CenProjMean*4)*Scale ;
err = OverallSurfaceArea2/OverallSurfaceAreaPar;
Diff_Percent = abs((1-err)*100) ;

fprintf('Parallel Projection Results : %.2f \n', OverallSurfaceAreaPar)
fprintf('Central Projection Results : %.2f \n', OverallSurfaceArea2)
fprintf('Scale : %.2f \n', Scale)
fprintf('Different Percentage : %.2f \n', Diff_Percent)

```

Appendix B: AutoProject Function

```

function AreaProject = AutoProject(points,u)
% Function for AutoProject in Certain Axis at Certain Degree

%% Theta Variables
theta30 = pi/6;
theta45 = pi/4;
theta60 = pi/3;
theta90 = pi/2;

theta120 = 2*pi/3;
theta135 = 3*pi/4;
theta150 = 5*pi/6;
theta180 = pi;

theta210 = 7*pi/6;
theta225 = 5*pi/4;
theta240 = 4*pi/3;
theta270 = 3*pi/2;

theta300 = 5*pi/3;
theta315 = 7*pi/4;
theta330 = 11*pi/6;

%% Legends
%R      = Rotation
%RX     = Rotation Transpose
%Coor_X = Replot Points to X axis
%Coor_Y = Replot Points to Y axis
%C      = Make Convex Hull for Projection Calculation
%AreaProject = Calculate Area of Projection
%% Main Script
mode = 'any';
theta = [theta30, theta45, theta60, theta90, theta120, theta135, theta150,...
        theta180, theta210, theta225, theta240, theta270, theta300, theta315,...
        theta330];

for i = 1:length(theta)
    R = rotate_3D(points',mode,theta(i),u); RX = R';

    Coor_X= RX(:,1); Coor_Y= RX(:,2);
    C = convhull(Coor_X,Coor_Y);
    AreaProject(i) = polyarea(Coor_X(C),Coor_Y(C));
end
end

```

Appendix C: AutoProjectCentral Function

```

function AreaProject =
AutoProjectCentral(points,u,AZ,EL,AOV,DphClbrn)
% Function for AutoProject in Certain Axis at Certain Degrass

%% Theta Variables
theta30 = pi/6;
theta45 = pi/4;
theta60 = pi/3;
theta90 = pi/2;

theta120 = 2*pi/3;
theta135 = 3*pi/4;
theta150 = 5*pi/6;
theta180 = pi;

theta210 = 7*pi/6;
theta225 = 5*pi/4;
theta240 = 4*pi/3;
theta270 = 3*pi/2;

theta300 = 5*pi/3;
theta315 = 7*pi/4;
theta330 = 11*pi/6;

%% Legends
% R      = Rotation
% RX     = Rotation Transpose
% Clbrn  = Find the Midle Position of the object
% DphClbrn = Calibration for Camera Target, the higher the number,
%         the further away from object
% XC     = Camera Target/Look-At
% A      = Transformation Matrix to Simulate the Central Projection
% AreaProject = Calculate Area of Projection

%% Main Script
mode = 'any';
theta = [theta30, theta45, theta60, theta90, theta120, theta135, theta150,...
        theta180, theta210, theta225, theta240, theta270, theta300, theta315,...
        theta330];

for i = 1:length(theta)
    R = rotate_3D(points',mode,theta(i),u);

```

```
RX = R';

ClbrnMAX = max(RX)/2;
ClbrnMIN = min(RX)/2;
Clbrn = ClbrnMAX+ClbrnMIN;

ClbrnX = Clbrn(1,1);
ClbrnY = ClbrnMIN(1,2)-DphClbrn;
ClbrnZ = Clbrn(1,3);

XC = [ClbrnX,ClbrnY,ClbrnZ];

A = viewmtx(AZ,EL,AOV,XC);
SZ = size(RX,1);
x4d = [RX(:,1),RX(:,2),RX(:,3),ones(SZ,1)]';
x2d = A*x4d;

x2(:) = x2d(1,:)./x2d(4,:);
y2(:) = x2d(2,:)./x2d(4,:);

ProjHull = convhull(x2,y2);
AreaProject(i) = polyarea(x2(ProjHull),y2(ProjHull));

end

end
```

Appendix D: Viewmtx Function

```

function a=viewmtx(az,el,phi,target)
narginchk(2,4);

if nargin==2
    phi = 0;
elseif nargin>2
    if phi>0
        d = sqrt(2)/2/tan(phi*pi/360);
    else
        phi = 0;
    end
end

% Make sure data is in the correct range.
el = rem(rem(el+180,360)+360,360)-180; % Make sure -180 <= el <=
180
if el>90
    el = 180-el;
    az = az + 180;
elseif el<-90
    el = -180-el;
    az = az + 180;
end
az = rem(rem(az,360)+360,360); % Make sure 0 <= az <= 360

% Convert from degrees to radians.
az = az*pi/180;
el = el*pi/180;

if nargin>3
    target = target(:); % Make sure its a vector.
    if length(target)~=3
        error(message('MATLAB:viewmtx:InvalidInput'));
    end
else
    target = 0.5 + sqrt(3)/2*[cos(el)*sin(az);-cos(el)*cos(az);sin(el)];
end

% View transformation matrix:
% Formed by composing two rotations:
% 1) Rotate about the z axis -AZ radians
% 2) Rotate about the x axis (EL-pi/2) radians

```



```
T = [ cos(az)      sin(az)      0      0
      -sin(el)*sin(az)  sin(el)*cos(az)  cos(el) 0
        cos(el)*sin(az) -cos(el)*cos(az)  sin(el) 0
        0              0              0      1 ];
```

```
if nargin==2 || phi==0, a = T; return, end % Return orthographic
transformation.
```

```
f = d; % Default focal length.
```

```
% Transformation to move origin of object coordinate system to
TARGET
```

```
O1 = [eye(4,3),T*[-target;1]];
```

```
% Perspective transformation
```

```
P = [1 0 0 0;
      0 1 0 0;
      0 0 1 0;
      0 0 -1/f d/f];
```

```
% The perspective transformation above works because the graphics
% system divides by the homogenous length, w, before mapping to the
screen.
```

```
% If the homegeous vector is given by V = [x,y,z,w] then the
transformed
```

```
% point is U = [x/w y/w].
```

```
% Using f = d places the image plane through the origin of the object
% coordinate system, thus projecting the object onto this plane. Note
only
```

```
% the x and y coordinates are used to draw the object in the graphics
window.
```

```
% Form total transformation
```

```
a=P*O1*T;
```

Appendix E: rotate_3D Function

```

function R = rotate_3D(V, mode, theta, u)
assert(nargin == 4, 'Error : too few or too many input arguments.
rotate_3D takes exactly four input arguments');
assert(size(V,1) <= 3 && size(V,1) > 0, 'Error : V dimension must be
strictly positive and less or equal to three. ');
assert(norm(u) ~ 0, 'Error : u must not equal to null vector. ');
assert(size(u,1) == 3, 'Error : u dimension must exactly equals to three. ');
assert(imag(theta) == 0, 'Error : theta must be real number. ');
u = u/norm(u);

if (size(V,1) < 3)
    V = cat(1, V, zeros(3-size(V,1), size(V,2)));
end
%% Rotation matrix construction %%
switch(mode)
case {'x', 'X'} % X axis rotation matrix ; u = i = [1 0 0]'
    Rm = [1      0      0;
          0 cos(theta) -sin(theta);
          0 sin(theta)  cos(theta)];
case {'y', 'Y'} % Y axis rotation matrix ; u = j = [0 1 0]'
    Rm = [cos(theta)  0  sin(theta);
          0           1  0;
          -sin(theta) 0  cos(theta)];
case {'z', 'Z'} % Z axis rotation matrix ; u = k = [0 0 1]'
    Rm = [cos(theta) -sin(theta) 0;
          sin(theta)  cos(theta) 0;
          0           0          1];
case {'any', 'ANY'} % Any u axis rotation matrix
    Rm = [u(1,1)^2+cos(theta)*(1-u(1,1)^2) (1-
cos(theta))*u(1,1)*u(2,1)-u(3,1)*sin(theta) (1-
cos(theta))*u(1,1)*u(3,1)+u(2,1)*sin(theta);
          (1-cos(theta))*u(1,1)*u(2,1)+u(3,1)*sin(theta)
u(2,1)^2+cos(theta)*(1-u(2,1)^2) (1-cos(theta))*u(2,1)*u(3,1)-
u(1,1)*sin(theta);
          (1-cos(theta))*u(1,1)*u(3,1)-u(2,1)*sin(theta) (1-
cos(theta))*u(2,1)*u(3,1)+u(1,1)*sin(theta) u(3,1)^2+cos(theta)*(1-
u(3,1)^2)];
    otherwise
        error('Bad mode argument : mode must be "x"/"X", "y"/"Y", "z"/"Z",
or "any"/"ANY". ');
    end
end
R = Rm * V;
End

```

AUTHOR BIOGRAPHY



The Author's name is Muhammad Rifqi Ramadhan, born on 05 February 1997 in Jakarta. Derived from a family with a Father named Budi Revianto and Mother named Rosnita. In 2015 author proceed to pursue bachelor degree at Department of Marine Engineering (Double Degree Program with Hochschule Wismar), Faculty of Marine Technology, Institut Teknologi Sepuluh Nopember Surabaya specializes in Marine Operation, Management and Maintenance. During the study period, author did some activities like Jemaah Masjid Manarul Ilmi ITS (2016 – 2018) and Barunastra Roboat Team ITS (2016-2018)

Muhammad Rifqi Ramadhan

m.rifqir@gmail.com

Motto: Dream Big, Work Hard, and Make It Happen