7-29-2020

Image Space Coverage Model for Deployment of Multi-Camera Networks

Eslam Samir Eissa
University of Windsor

Image Space Coverage Model for Deployment of Multi-Camera Networks

by

Eslam Samir Eissa

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada

2020

Image Space Coverage Model for Deployment of Multi-Camera Networks

by

Eslam Samir Eissa

APPROVED BY:

A. Jalal

Department of Mechanical, Automotive and Materials Engineering

---

B. Balasingam

Department of Electrical and Computer Engineering

X. Chen, Advisor

Department of Electrical and Computer Engineering

June 12, 2020

Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

When it comes to visual sensor networks deployment and optimization, modeling the coverage of a given camera network is a vital step. Due to many complex parameters and criteria that governs coverage quality of a given visual network, modeling such coverage accurately and efficiently represents a real challenge.

This thesis explores the idea of simplifying the mathematical interpretation that describes a given visual sensor without incurring a cost on coverage measurement accuracy. In this thesis, coverage criteria are described in image space, in contrast to some of the more advanced models found in literature, that are formulated in 3D space, which in turn will have a direct impact on efficiency and time cost.

In addition, this thesis also proposes a novel sensor deployment approach that examines the surface topology of the target object to be covered by means of a mesh segmentation algorithm, which is that a different way to tackle the problem other than the exhaustive search methods employed in the examined literature.

There are two main contributions in this thesis. Firstly, a new coverage model that takes partial occlusion criterion into account is proposed, which is shown to be more accurate and more efficient than the competition. Next, a new sensor deployment method was presented that takes the target object shape topological properties into account, an approach that is to the best of our knowledge, was not attempted in literature before at the time of publication.

This thesis attempts to support all of claims made above, the proposed model is validated and compared to an existing state of art coverage model. In addition, simulations and experiments were carried out to demonstrate the accuracy and time cost efficiency of the proposed work.

Dedication

To my Family and Friends

Acknowledgements

I would like to thank my advisor Dr. Xiang Chen, for his patient guidance and continued support that he provided throughout my time as his student. I have been very lucky to have supervisor who cared so much about quality and high standards.

I'd also thank the members of my committee, Dr. Ahmad Jalal and Dr. Balakumar Balasingam, for the valuable feedback they provided toward my research, and the time they generously spent reviewing my work.

I'd also like to thank my fellow students Tong Zhang and Zike Lei for their helpful comments and insights they gave me towards finishing my thesis.

My most sincere gratitude and appreciation goes to my family and friends that supported me and encouraged me to pursue graduate studies and push forward even when faced by adversary.

Finally, I'd like to thank University of Windsor for giving me the chance to study aboard and also for its generous financial contributions towards this research.

Contents

List of Tables

List of Figures

List of Algorithms

Introduction

CHAPTER 1

Introduction

1.1 Background

Computer vision is the enterprise of processing and extracting information from a given visual sensor. It attempts to build autonomous systems that can carry out some of the tasks that a human vision system can do. Computer vision has a wide array of applications that include but are not limited to: scene reconstruction, object recognition, 3D pose estimation and visual servoing.

Computer vision tasks are generally related to extracting useful information from a stream of data that comes from some sort of visual sensor, sometimes called a field sensor in literature. The most commonly used visual sensor today is the camera, due to its various inherent advantages, such as: low cost, light weight and rich information output. Almost all cameras that are used nowadays are digital cameras or RGB cameras (because they use the RGB coloring model).

Light is reflected from the observed object through the camera's aperture onto to a digital image sensor that is made up of an array of photosensitive cells. Each cell corresponds to a pixel in the output image. The camera lens focuses the incoming light onto the image sensor. While the aperture controls the amount of light coming through, the shutter controls the duration of time the light is hitting the sensor surface.

Because a single camera generally has a limited field of view, multiple cameras are sometimes used in conjunction to cover larger surfaces. They are sometimes called multiple camera networks (MCNs) or visual sensor networks (VSNs) in common literature. A multiple camera network such as a stereoscopic vision system takes inspiration from the human vision system, and how it uses two view angles (corresponding to two eyes) to gain additional depth information about the inspected object. Expanding on this principle, adding more views would lead to gaining more depth information as well as covering more surface area.

1.2 Motivation

Visual sensors or field sensors as they're sometimes called in literature, are a class of sensors that can cover more than one point in space [1] . They usually deliver their data in multi-dimensional format, a camera for example provides its data in a form of 2D image, while a LIDAR, on the other hand outputs a 3D point cloud representation of its surroundings. Visual sensors are considered indispensable when it comes to computer and machine vision applications, therefore one can see the importance of visual sensor network planning and optimization.

Camera networks have a various application in many fields. When it comes to industry and manufacturing, they're typically used for automated inspection and object recognition tasks. They're are also used extensively in robot navigation and autonomous vehicles. For example, Tesla's autopilot technology uses a network of eight cameras to provide 360 degrees of visibility around the vehicle and give it the ability to maneuver around busy road conditions in cities [2] .

Camera networks are also heavily used in environment conservation and wildlife protection activities. Gonzalez et al. used cameras mounted on UAVs to survey threatened and invasive species for purposes of wildlife monitoring and conservation [3] . Following on the same path, Casbeer et al. also used UAV camera networks to monitor and survey forests for wild fires [4] .

In the area of 3D reconstruction, Moons et al. have used inputs from multiple cameras to build a 3D CAD representation of inspected objects from images that were taken from various angles [5] . Wu et al. proposed a method to carry out real-time reconstruction of the human posture using a network of cameras [6] . Moving on the field of surveillance, Angella et al. proposed a non model-based framework to optimize camera networks for surveillance purposes [7] . Fu el al. designed a method that uses particle swarm optimization to maximize coverage of 2D plane surfaces using a network of cameras [8] .

Finally, sensor networks play a vital role in industrial automation in the sense that they provide machines a sense of visual perception to carry out repetitive task such as quality inspection and tag identification [9] .

1.3 Preposition

In this thesis, we attempt to solve the problem of optimal deployment of multi camera systems, by proposing two main contributions: a novel mathematical model that quantifies the coverage strength of a given camera, and a new optimal camera deployment method that examines the inspected object

surface topology and then use a method of mesh segmentation to divide the object into different segments in the sense that, each segment or region have similar topological properties.

The main difference between the proposed coverage model and the ones in literature is about the approach that the model takes to process the coverage criteria. The proposed coverage model processes the coverage of the observed object in image space. It works by projecting the object into the camera's image plane first, then process the object for coverage, whereas other models in literature, they process coverage in 3D space. One clear advantage for the case of 2D image-based coverage is reduced computational complexity and time cost.

In the proposed model, we introduce a new occlusion criterion, in which partial or "graded" occlusion per unit triangle is taken into account, unlike other models in the literature that evaluates the occlusion criterion as a binary value, where '0' or '1' values are given to each triangle, corresponding to the triangle being completely occluded or completely visible relative to a given camera.

Regarding the proposed camera placement method, it uses a 3D mesh segmentation algorithm to separate the inspected object into several regions, where each region would have similar average surface orientation, then a candidate camera pose would be generated for each region. The main advantage of such approach is that it avoids the exhaustive search methods employed by other methods in the literature.

1.4 Thesis Outline

This thesis begins with Chapter 2, which is a literature review. It begins with an explanation of the notion of coverage modeling, which goes through coverage model dimensionality and task definition, followed by a brief definition of the concept of valuation and measurement of model criteria. Review of literature is then conducted on each of the individual coverage criteria, then analysis and comparison are made between the examined models.

Part I, which includes Chapters 3,4 and 5, explains the theory behind the Image Space Coverage Model, while Part II, which include only Chapter 6, offers an application on the model, in which the proposed method is used to solve a camera deployment problem.

The Image Space Coverage Model is proposed in Chapter 3. This chapter begins with explanation of some prerequisite concepts that are related to computer vision such as image formation and projection. After that the full formulation of the model is presented.

In Chapter 4, A simulation using the *Khepra*[1] tool was conducted and the accuracy and time cost of the model are compared to a model that represent the state of the art.

In Chapter 5, experimental validation of the proposed model is conducted using fiducial marker detection as task and series of tests is carried out to validate the proposed model.

Chapter 6 goes through the shape segmentation-based camera deployment, along with simulations and comparisons. Chapter 7 concludes this thesis and provides possible directions for future work. Appendix A provides documentation and information about the *Khepra Simulation Environment*, a tool that was developed specifically for the work in this thesis.

[1] More information is provided about Khepra in Appendix A

Previous Work

CHAPTER 2

Literature Review

2.1 Overview

Modeling and optimization of visual sensor networks are considered an area of active research; therefore, we can find a large number of publications and writings of quality work that span several decades back. In this chapter we'll review some of them. This chapter will be divided in two sections, the first section will focus on previous work done on coverage quantification models, while the second section will cover sensor deployment methods and optimization techniques.

Literature review of coverage modeling will follow its evolution in a chronological order, where earlier works are presented first, then later developments are discussed and highlighted.

2.2 Coverage Modeling

In order for us to be able to automate the process of camera view planning and deployment, first we must have a way to judge if a given camera view is considered to be good or bad. Of course, we're aware that what is good or bad is subjective to each person and heavily depends on the task at hand. The task is defined by what we are trying to do and what is our end goal. Are we trying to maximize coverage resolution? Or we're trying to minimize occlusion? From here comes the need to model our coverage requirements and needs.

A coverage model is a mathematical model that quantifies what the vision system can see or cover with respect to our requirements. So, if we require to maximize resolution and minimize focus blur, we would use a model that rewards views with high resolution properties while penalizing views with blurry properties. Maviranc and Chen have provided a sublime survey on this particular subject [9] .

2.2.1 Dimensionality of a Coverage Model

Because the physical universe is modeled by three-dimensional Euclidean space, it makes sense to model vision as a three-dimensional construct. However, some researchers, for the sake of simplification, used

two-dimensional models, where they assumed that all sensors and inspected subjects are located on a common plane and all occluding agents are made up of high vertical barriers. Such formulation is reminiscent of the well-known art gallery problem [9] .

An important aspect that accounts for the behavior of a given coverage model, is its *geometry*. The geometry of a coverage model is the representation of the space covered by a given sensor. A common example of such representation would be the volume of three-dimensional Euclidean space that is covered by a given sensor's view. Such models are generally called *Geometric coverage models* in literature, and this thesis shall focus on them, as opposed to other types of coverage modeling, such as *Topological modeling*.

2.2.2 Definition of the Inspected Task

Coverage measurement methods are usually divided into two categories, model based and non-model based. Model based means that the inspected object shape or *task* (as sometimes called in literature) is known beforehand, which usually comes in the form of a 3D CAD file, while in non-model methods, the shape is unknown. In such cases, typically the inspected area or volume space is divided into a discrete number of points and a coverage strength value is given to each point.

A 3D CAD file is made up of vertices and triangles. Triangles represents the smallest atomic unit that defines surfaces inside the file, each triangle is defined by three vertices positioned in three-dimensional Euclidean space. The positions of these vertices define triangle orientation and size.

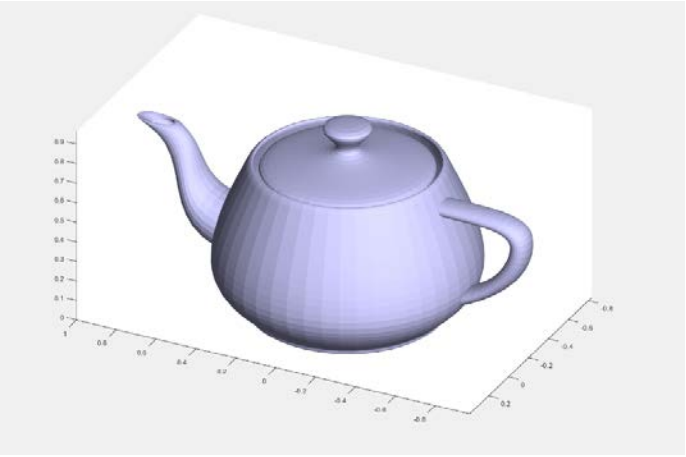


Figure 2.1: A 3D CAD file of the well-known Utah teapot

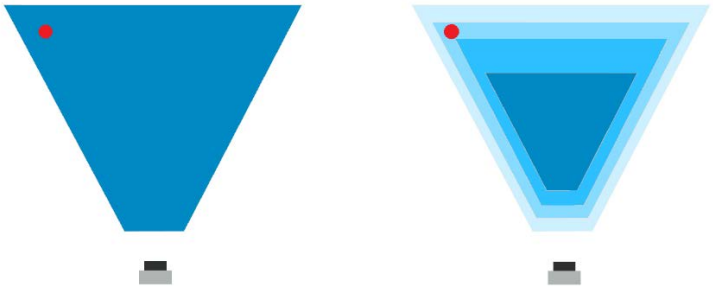
It is evident that choosing a suitable method depends on the application at hand and whether shape of the task exists beforehand or not. While non model-based models seem to be more versatile because they don't need a model to function, their shortcoming is that they depend on the amount discretization

or sampling used. High sampling will lead to more time cost, unlike model-based approaches, whose time cost depends on the complexity of the inspected object's CAD file.

2.2.3 Model Criteria Valuation

The next definition is bivalent vs graded or real-valued: when measuring coverage performance of a given point, the coverage value can be either bivalent or real-valued: Bivalent models assigns binary value of (0 or 1), which means either as covered or not-covered. While Real-Valued models assign a real value number to each inspected point.

An example is shown in Figure 2.2 where these two shapes represent a camera's field of view- the bivalent model on the left has assigned a value of 1 to the red point just because it lies inside the Field of View, while the real-valued model on the right gives priority to objects on the middle of the Field of View, and penalize objects that lies on the borders, so it gives a partial coverage value to it because it lies on the edges.



(a) Bivalent (b) Graded

Figure 2.2: Right red point assigned partial coverage (40%) value

One could argue that real-valued models are generally more accurate and are a better representation of real-world applications.

2.2.4 Coverage Measurement Criteria

When it comes to evaluating coverage performance of a given vision system, just like evaluating performance of any system, it has to be done against some metrics or criteria. Various criteria have been proposed and used by researchers over the past two decades. We will look at five basic criteria that were commonly used in literature.

2.2.4.1 Field of View (FOV)

The region of three-dimensional Euclidean space of $\mathbb{R}^3$ which is said to theoretically visible to a given visual sensor, it's commonly described as *the view frustrum*, with its apex positioned at the optical center of the sensor. The frustrum dimensions are defined by two angles, which are the horizontal and vertical apex angles. These two angles depend on *focal length* and *sensor dimensions* of a given camera.

Field of View is typically considered to be the most important criterion because it defines what the camera can view. From computational efficiency point of view, it makes sense to calculate Field of View coverage first then the rest of the criteria, so that we only evaluate what is inside the field of view only and not waste valuable computation time on element that might not be inside the vision field.

Two-dimensional representations were used for the sake of simplification, such representations were modeled as fan shaped or a pie chart sector. It was used by Ma and Liu [10] , Ai and Abouzeid [11] and Jiang et al. [12] . Horster and Lienhart simplified it even more by used a triangle shape [13] .

Moving on to the third dimension, Erdem at al. used a frustrum or pyramid that was defined by two apex angles [14] . Malik and Bajscy, Maviranc et al. and Alarcon also followed the same steps [15] . Cowan and Kovesi and Tarabanis et al. have managed to simplify the previous model by using a regular pyramid that was defined by the one apex angle only, which was the smaller one [18] [19] . Other less common geometric representations were used, such as that of a cone, which was used by Piciarelli et al., where the cone apex angle was equal to the smaller of the two apex angles [20] .

When it comes to coverage calculation, Mavrianc et al. and Zhang et al [16] would check if the subject physically lies within the 3D Euclidean space of the viewing frustrum. Alarcon, in his PhD dissertation, used a tensor framework to model the position and dimensions of the viewing frustrum, and then measure the distance of the sensor pose from an assumed optimal pose. The measure, which he dubbed *vision distance*, uses both Euclidean norm and Frobenius norm to measure translational and rotational difference from the optimal pose [17] .

2.2.4.2 Resolution

Defined as the minimum resolution that is needed to sufficiently cover a given task or subject; it is defined by the amount of the photosensitive cells that exists on the image sensor of the camera.

Some researchers proposed a distance limit as a resolution constraint, in which any subject that falls beyond the distance limit is considered uncovered. In the two-dimensional models of Ma and Liu [10] and Jiang el al. [12] , resolution constraint was put as distance limit on the sector, whereas Cowan and Kovesi [18] used a cap cut out sphere to model the resolution constraint. Maviranc and Chen see that

using a circular or a spherical model here is overcomplicates the matter, as the projected image is always going to be planar, and they see that using a triangle as a model is a better option [9] .

Erdem and Sclaroff, Malik and Bajcsy and Marviranc et al., all consider the resolution as a function of depth along the optical axis [14] [15] [16] . Zhang el al. proposed a new resolution measure in which, the angle of optical axis with respect to the inspected surface is accounted for, eliminating the need to add such angle as a separate criterion, like what other models did [22] . The new measure examines resolution as pixel/mm.

2.2.4.3 Focus (Depth of Field)

Minimum amount of image sharpness that is need to sufficiently cover a given subject. Image blurriness is defined by a distance range of the subject from the optical center, such range is termed *depth of field.*

In the work of Park el al., focus is taken into account as near and far focus distance limits are incorporated into the coverage formulation [23] . Wang et al. in addition to Maviranc et al. also follow on the same steps [24] [16] .

2.2.4.4 Angle

Refers to the angle at which the camera is facing a given surface. It's defined as the angle between the optical axis and the normal direction of the inspected surface. It is natural to assume that a lower value of this angle would result in a better coverage, because more pixels would be utilized to cover the surface.

This particular metric has been utilized in literature according to the task in question. For example, for the task of face tracking, Shen et al. incorporated the angle criterion in their model [25] . Maviranc et al. Chen also added the criterion in their model, dubbed the *Coverage Strength Model,* aiming to achieve a more accurate coverage measurement [16] .

Zhang et al. managed to eliminate the angle metric by proposing a new resolution measure, that takes view angle into account. A step which made the coverage formulation more compact [22] .

2.2.4.5 Occlusion

The problem of occlusion detection is considered to be a very challenging one. Occlusion detection has its root from the well-known visibility problem. If we're given a group of obstacles in Euclidean space, how can we determine if two points in space are visible to each other? One common way is to check if the line segment that connects both of them does not intersect any obstacles [26] . The visibility

problem is considered to be one of the basic computational geometry problems and has many applications in computer graphics, motion planning and other fields.

Some models completely ignore this metric, such as the model proposed by Malik and Bajcsy, which assume an empty room with coverage priority placed on the center [15] . Two-dimensional approaches on the other hand, treat occluding obstacles as vertical high barriers or walls. Erdem and Sclaroff presented an algorithm to evaluate such occlusion [14] .

Moving on to three-dimensional cases, Angella el al. advises a discrete occlusion checking method, where accuracy of such checking depends solely on discretization of the inspected volume [7] . Zhang el al. employed a different criterion in his model-based framework, where they use a triangle-ray intersection algorithm to judge if the line segment that connects triangle vertices and the camera's optical center is intersecting any other triangles [21] . Although such approach is considered to be the one to yield the most accurate results, it's comes with huge time cost due to the nature of 3D triangle-ray intersection checking, a reality that pushed Zhang el al. to publish a parallel based occlusion checking algorithm in their survey paper [21] .

2.2.5 Analysis and Comparison

In this section, we'll compare some selected models from literature and highlight the development of these models across the time. These particular models were selected because of two aspects: First, they represent the development the state of the art. Second, they're similar to the proposed model, as they are all three-dimensional based and they all use graded criteria.

By examining Table 2.1, we can see the development of these models. The second column labeled *Geometry,* refers to criteria formulation not model dimensionality. Model geometry is directly related to time cost, as simpler geometry representation will lead to less time cost and a more efficient model, and vice versa. All of the examined models have three-dimensional formulations. The model proposed by Tarabanis et al. [19] takes two criteria only into account and resolution is the only graded criterion. While Scott adds to the above the focus criterion [27] . Mavrianc et al. proposed a model that takes all criteria into account, with field of view only is a graded criterion[16] . Alarcon's model takes self-occlusion (self-occlusion is defined here) only into account, while having two criterions as graded [17] . Zhang et al. model represent the state of art and it takes all criteria into account and two of them are graded [21] .

Table 2.1: Comparison summery between state of the art

Model	Geometry	FOV	Resol.	Focus	Occlusion	Angle	Graded?
Tarabanis et al. [19]	3D based	✓	✓	✘	✘	✘	Resolution
Scott [27]	3D based	✓	✓	✓	✘	✘	Resolution
Maviranc et al. [16]	3D based	✓	✓	✓	✓	✓	FOV
Alarcon [17]	3D based	✓	✓	✓	Self Only	✓	Resolution, Angle
Zhang el al. [21]	3D based	✓	✓	✓	✓	✓	Resolution, Angle

2.3 Camera Deployment and Optimization

The problem of coverage optimization is about achieving a maximum coverage of the observed task using minimum number of sensors. Coverage optimization problem is reminiscent of the well-known art gallery problem [9] , which stems from the real-life scenario of monitoring an art museum with the minimum number of guards. Camera deployment or optimal camera placement as sometimes is called in literature, is the process of determining the optimal intrinsic and extrinsic parameters of the cameras used. Intrinsic parameters refer to the sensor internal parameters, such as focal length, sensor size, resolution, etc. while extrinsic parameters refer to the camera pose in space [28] .

In research, efforts have been made to optimize the process of camera deployment, either by turning it into a minimization optimization problem to minimize the number of cameras used, or maximization problem to maximize coverage. In this section, we'll survey some these works.

Cowan and Kovesi used their coverage model to obtain geometric constraint from the coverage requirements [18] . Tarabanis also followed on the same path, in the sense of generating solutions from constraints [19] . Park el al. managed to generate a discrete solution space of candidate camera poses according some visual criteria, then search the solution space for optimal solutions [23] .

Scott proposed the idea of generating a solution space, in which one possible candidate view is generated for each triangle surface in the CAD file of the inspected task, for which it is guaranteed to be an optimal view for that particular triangle. A visibility matrix that describes the coverage of each candidate view in the solution space is constructed, and then a greedy algorithm used on the matrix to maximize coverage [27] . Alarcon follows on the same steps of using a visibility matrix, but manages to

extend the solution space by detecting convex regions in the object and generate additional views for each region [17] . As illustrated in Figure 2.3, which depicts camera deployment performed on a convex shape, which comes in the form of a yellow pyramid. Using Scott's solution space generation, four views are generated for each triangle surface, denoted by the blue cameras. Alarcon's method takes the average the four poses to get additional optimal view, which is shown as the red camera. After that Alarcon looks for a solution by means of a particle swarm optimization (PSO) algorithm [17] . Maviranc also made use of a particle swarm optimization in his PhD dissertation for coverage optimization [26] .

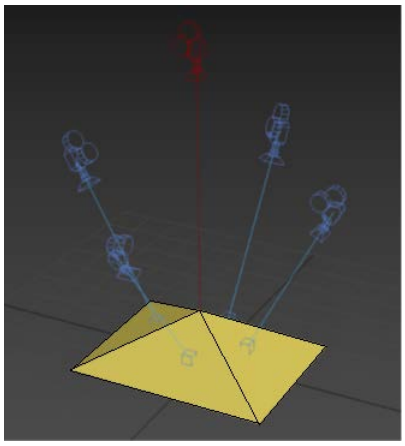


Figure 2.3: Alarcon's extended solution space: an additional optimal view (red camera) is added by averaging poses of the blue cameras

Chen and Li have made use of a genetic algorithm to solve the problem [29] , Jiang et al. also followed their steps [12] . Malik and Bajcsy combined the use of a genetic algorithm and a gradient decent algorithm to perform optimization for stereo camera deployment applications [15] .

In [22] , Zhang et al. used a recursive convex optimization algorithm to find maximize coverage of model triangles. Zhang et al. also wrote an excellent survey on the subject of coverage optimization, where they tried several optimization algorithms for coverage applications and compared their performance and time cost [21] . They have found that a binary integer programming (BIP) algorithm provides the best performance at the cost of time. Greedy algorithm is the fastest approach at the cost of performance. They found that genetic algorithm (GA), particle swarm optimization (PSO) and differential evolution (DE) methods achieve a balance between performance and time cost. They concluded that differential evolution (DE) preforms the best among the three.

Part I
Theory Behind Coverage Modeling

CHAPTER 3

A Model of Visual Coverage

3.1 Overview

In this chapter, we propose a general, high-fidelity model-based coverage model. This model was developed while keeping the observations made in Chapter 2 in mind. The focus in developing this model is to try to simplify model representation without incurring a cost on modeling accuracy.

In an attempt to make the model more accurate in representing real world scenarios and applications, occlusion criterion was reintroduced as graded real-valued metric and an algorithm to compute partial occlusion is proposed. New resolution measure is also proposed, where it deals by measuring directly the effective number of pixels that cover a given task by the camera.

3.2 Comparison with State of the Art and Contributions

The main contribution of this model is the reimagination of its geometry formulation as image based instead of 3D based formation. In this section, we'll compare some selected models from literature with the proposed model and highlight expected improvements. We revisit Table 2.1 in Table 3.1, where we can see a comparison between previous models and the proposed one. In addition to including all the main coverage criteria, the proposed model treats occlusion as a graded criterion, which is considered to be one of the main features and contributions of this work. The proposed model, which is dubbed "*Image Space Coverage Model",* is a three-dimensional model with "image based" or two-dimensional geometry, hence the name.

Table 3.1: Comparison summery between proposed model and selected ones

Model	Geometry	FOV	Resol.	Focus	Occlusion	Angle	Graded?
Tarabanis et al. [19]	3D based	✓	✓	✘	✘	✘	Resolution

Scott [27]	3D based	✓	✓	✓	✘	✘	Resolution
Maviranc et al. [16]	3D based	✓	✓	✓	✓	✓	Resolution
Alarcon [17]	3D based	✓	✓	✓	Self Only	✓	Resolution, Angle
Zhang el al. [21]	3D based	✓	✓	✓	✓	✓	Resolution, Angle
Proposed model	Image based	✓	✓	✓	✓	✓	Resolution, Angle, Occlusion

3.2 Image Formation

When it comes to forming a projected image of a three-dimensional scene represented using a global reference frame, a series of transformations needs to take place first [28] . Global coordinate frame, which sometimes is referred to as world coordinate frame, is represented in real three-dimensional coordinate space, denoted as $\mathbb{R}^3$. The image projected on the sensor's image plane, is represented in real two-dimensional coordinate space, denoted as $\mathbb{R}^2$. As shown in Figure 3.1, world coordinates are converted to camera coordinates, then from camera coordinates to image coordinates. Normally image space doesn't have the third dimension '*Z*' because an image is typically two-dimensional, but in our model, we keep it because it will become handy in image space occlusion and focus detection in section **3.3**.

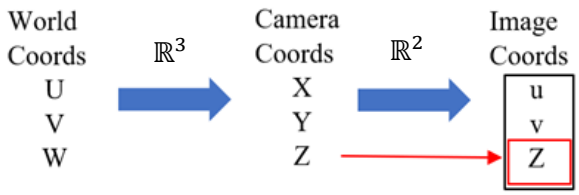


Figure 3.1: Series of transformation to convert from world coordinates to image coordinates, depth component '*Z*' is kept as image depth parameter

3.2.1 The Pinhole Camera Model

The pinhole camera model is a mathematical model that describes the behavior of an ideal pinhole camera. It describes the relation between a point in 3D space and its projection on the camera's image

plane in 2D space. In this model, the camera's aperture is described as a small point, (hence the pinhole naming) and the model does not account for lenses and their effects like lens distortion and focus.

As far as computer vision and computer vision applications are concerned, the pinhole model is often used as a good example of how a camera forms an image of a scene, that's because the effects that are not included, such as lens distortion, is so small in today's modern high quality cameras, that they can be safely neglected [28] .

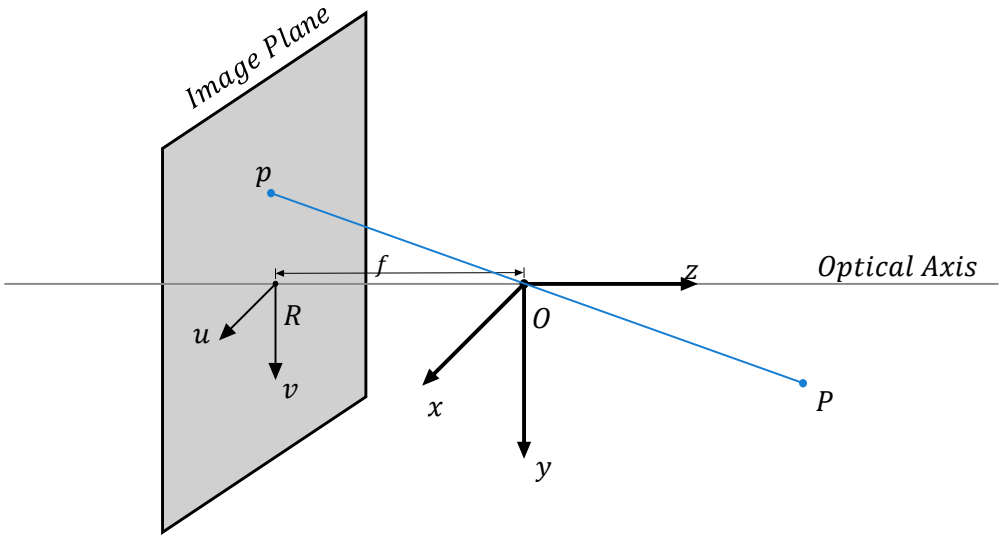


Figure 3.2: Pinhole Camera Model: The image plane sets at a distance 'f' from the optical center point 'O', point 'P' is projected on image plane at point 'p'

Referring to Figure 3.2, we can see that the camera's image plane is located at distance 'f' from the camera's optical center point 'O'. Point 'R' on the image plane is commonly referred to as the principal point in literature, which is the origin of the image coordinate system. Point 'P' will have its projection point on the image plane as point 'p'.

Projected point 'p' is given by:

$$p = [-f\frac{p_x}{p_z}, -f\frac{p_y}{p_z}]^T \tag{3.1}$$

3.2.2 Mapping World to Image Coordinates

Equation (3.1) relates a three-dimensional point in camera local coordinates to two-dimensional coordinates, but what about converting from world coordinates? A projected image of a point in world coordinates is given by:

$$p_h = CP_h \tag{3.2}$$

Where, p_h is projected point in homogenous coordinates, P_h is world point in homogenous coordinates and they are given by:

$$p_h = \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix}, P_h = \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} \tag{3.3}$$

And C is a 3×4 camera matrix, which is given by:

$$C = KT \tag{3.4}$$

$$K = \begin{bmatrix} \frac{f}{s_u} & 0 & R_u \\ 0 & \frac{f}{s_v} & R_v \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

K is a 3×3 intrinsic matrix, which a matrix that describe the camera intrinsic parameters such as focal length 'f', pixel dimensions 's_u' and 's_v'. 'R_u' and 'R_v' represent the position of the principal point on the image plane. T is a 4×4 homogenous transformation matrix that belongs to orthogonal group $SE(3)$ that represent the pose of point 'P' with respect to world coordinates.

3.3 Image Based Coverage Function

As we discussed in subsection 2.2.2, that our inspected task model is based on 3D CAD objects representation. This representation is composed of vertices and triangular faces, these triangular faces are the simplest representation of a given surface, or a plane. A triangular face orientation is defined by the positions of its three vertices in space. For each camera, a coverage function is to be evaluated for each triangular face of the inspected CAD task. The following definitions will define what constitute a covered triangle:

---

1 **Definition** (Covered Triangle):

A triangle is considered to be covered if and only if all its three vertices are covered.

---

2 **Definition** (Covered Vertex):

A vertex is considered to be covered if and only if all its coverage criteria are satisfied.

Before we can evaluate the coverage of a given triangle, we first need to convert it to image space, so we apply transformation to convert from world to image plane space, $T_w \rightarrow T_i$ where we use formula (3.2) to convert all of the triangle vertices. So, for a given camera 'C', for each projected triangle 'T_i' that it observes, the overall coverage is given by:

$$C_{overall}(T_i, C) = C_{FOV}(T_i, C)\ C_R(T_i, C)\ C_F(T_i, C)\ C_V(T_i, C) \tag{3.6}$$

The four different components of the overall function will be defined in the upcoming subsections.

3.3.1 Field of View

As discussed in subsection 2.2.2.1, the field of view of a given camera is typically represented as frustrum, with other models evaluate its coverage by checking if the task is geometrically positioned inside the frustrum. The following formulation attempts to simplify the geometry to image space.

For a given camera 'C', for each project triangle 'T_i' that it observes, the field of view coverage is given by:

$$C_{FOV}(T_i, C) = \begin{cases} 1 & if\ T_i \in I \\ 0 & \text{otherwise} \end{cases}, \text{ where '}I\text{' is the image plane} \tag{3.7}$$

3.3.2 Resolution

A novel resolution criterion is introduced in this thesis. The resolution is measured as the number of pixels that are covering a triangle of a given inspected object. It is measured as pixels per surface triangle.

For a given camera 'C', for each projected triangle 'T_i' that it observes, the resolution coverage is given by:

$$C_R(T_i, C) = \frac{Area(T_i)}{R_{min}} \tag{3.8}$$

Where $Area(T_i)$ is the area of the projected triangle and R_{min} is a task parameter, defined as minimum required resolution per triangle. This task parameter maybe used by the user to obtain sufficient resolution coverage for whatever task at hand.

Given a triangle T_i with vertices a, b and c which are defined in two-dimensional image space coordinates, the area of T_i is given by:

$$Area_{T_i}(a,b,c) = \left| \frac{a_x(b_y - c_y) + b_x(c_y - a_y) + c_x(a_y - b_y)}{2} \right| \tag{3.9}$$

3.3.3 Focus

Real camera lenses focus light at the image sensor at precise depth distances, which is the distance of the observed subject to the camera. Objects that are closer or further than subject distance suffer from a blur effect, this effect is called *Circle of Confusion* or *Blur Circle,* because it occurs in the form of a blurry circle in the image. The limits of acceptable focus in photography is typically called Depth of Field and it is defined by two distances for the near and far limits z_n and z_f.

These distances are given by:

$$z_n = \frac{Afz_s}{Af + c \min(s_u, s_v)(z_s - f)} \tag{3.10}$$

$$z_f = \frac{Afz_s}{Af - c \min(s_u, s_v)(z_s - f)} \tag{3.11}$$

Where 'A' is the diameter of the camera's aperture, 'f' is the focal length, 'z_s' is the subject in focus distance, 'c' is a task parameter defined as the maximum acceptable blur circle diameter in pixels and 's_u' , 's_v' are pixel dimensions.

For a given camera 'C', for each projected triangle 'T_i' that it observes, and 'T_i^z' is the distance of the triangle's centroid from the camera, the focus coverage is given by:

$$C_F(T_i, C) = \begin{cases} 1 & z_n \le T_i^z \le z_f \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

3.3.4 Visibility/Occlusion

Occlusion is often considered as the antonym of visibility, as far as computer vision and computer graphics applications are concerned, it is difficult to define one without the other. According to Zhang, occlusion handling represents a challenge in the field of visual sensor networks, especially for 3D CAD models, because they're composed of a large number of triangles and processing them usually have a huge time cost [21] .

A contribution of this thesis is the proposal of a reimagined graded, real valued visibility metric, that operates in the two-and-a-half dimension. Real value means more representation accuracy and less dimensions means less time cost.

Let 'C' be a camera and 'T_i' a projected triangle that it observes and 'j' is the number of all the other projected triangles that intersects with 'T_i' on the image plane, where 'T_1' is the closest triangle to the camera and 'T_j' is the furthest one to the camera, the occlusion coverage is given by:

$$O(T_i, C) = \begin{cases} \left(1 - \dfrac{Area(T_j \setminus \bigcup_{m=1}^{j-1} T_m)}{Area(T_j)}\right) & \text{if } O_{T_i} \text{wasn't calculated yet} \quad (3.13a) \\ 1 & if\ Area((\bigcup_{m=1}^{j-1} T_m) \cup T_j) = Area(\bigcup_{m=1}^{j-1} T_m) \quad (3.13b) \\ 1 & if\ T_i \text{ is self} - \text{occluding} \quad (3.13c) \\ \left(1 - \dfrac{Area(VA_{T_j} \setminus \bigcup_{m=1}^{j-1} T_m)}{Area(T_j)}\right) & otherwise \quad (3.13d) \end{cases}$$

The letter 'i' indexes the triangle in the CAD object, while the letter 'j' indexes intersected projected triangles. If a given projected triangle T_i whose occlusion coverage value is not processed yet, then formula (3.13a) is used, where the visibility of T_i to camera is the ratio between the areas of: the Boolean difference between the currently processed triangle T_j and the Boolean union of all the intersected triangles that are in front of it (i.e., the triangles that are closer to the camera than it) and the area of the currently processed triangle T_j. One minus visibility would give us occlusion, where '0' denotes that 0% of the triangle is occluded and '1' denotes that 100% of the triangle is occluded.

Formula (3.13b) is for detecting the completely occluded case. If area of the union of the currently processed triangle T_j and all the projected triangles in front of it is equal to the area of union of all the front triangles, then triangle T_j is completely occluded to the camera and is given occlusion value of 1.

Formula (3.13c) refers to a back-facing triangle, which is a definition that first must be established:

---

3 **Definition** (Self-occluding Triangle):

A triangle is considered to be self occluding if and only if the angle between triangle normal vector $\overrightarrow{N_T}$ *and optical axis vector* $\overrightarrow{O_c}$ *is less than or equal* 90 *degrees.*

A self-occluding triangle means a triangle whose normal direction (normal direction is vector that is perpendicular to the surface of the triangle) is facing more than 90 degrees away from the optical axis, self-occluding triangles are not visible to the camera and thus they're assigned occlusion value of 1.

Formula (3.13d) is used when triangle T_i occlusion value was processed before; it is the same formula as (3.13a) the only difference is we use the expression VA_{T_j} which refers to the visible portion of the triangle from all previous processed iterations.

Now to get visibility from occlusion, all we have to do is subtract occlusion value from a value of one. For a given camera 'C', for each projected triangle 'T_i' that it observes, the visibility coverage is given by:

$$C_V(T_i, C) = 1 - O(T_i, C), \text{ range } [0,1] \tag{3.14}$$

CHAPTER 4

Simulation of the Proposed Model

4.1 Overview

In this chapter, a simulation of the coverage model that was proposed in Chapter 3 will be conducted. Simulation of the model was carried out using the *Khepra Simulation Environment*. A three-dimensional simulation environment for visualization and planning of multi-camera networks. *Khepra* is a standalone, user-friendly tool with a complete GUI that was developed via MATLAB specifically for this project.

Simulation of the proposed model was compared with another model that represent the state of the art, where a group of different 3D CAD models with different complexity were used as inspection tasks. Accuracy and time cost were compared and the results were reported. The results were found to indicate accuracy and time cost efficiency of the proposed model.

4.2 Graded Occlusion Evaluation Algorithm

In this subsection, we'll present the algorithm to calculate the graded occlusion criterion using formulas that were proposed in subsection 3.3.4. This algorithm is used by the *Khepra Simulation Environment* in order to evaluate overall coverage. The algorithm takes three inputs, where T_{uvw} is the set of projected triangles of the inspected task defined in 2.5D image space coordinates. $\overrightarrow{N_T}$ is a set of normal vectors for each triangle in three-dimensional space, $\overrightarrow{O_c}$ is a vector of the camera's optical axis, also defined in three-dimensional space. The output is O_T which is an array contain the partial occlusion value for each triangle in the inspected task.

Algorithm 1: *Graded Occlusion Evaluation*

Input: T_{uvw}, $\overrightarrow{N_T}$, $\overrightarrow{O_c}$

Output: O_T

Initialization: Initialize triangle occlusion matrix $O_T = NaN$, and triangle visible area matrix $VA_T = NaN$.

for $j = 1 \rightarrow N$ **do**

  if $\angle\left(\overrightarrow{N_T}, \overrightarrow{O_c}\right) \leq 90°$

    $O_{T_j} = 1$

    continue

  else

    Find all triangles that are overlapping with T_j

    Sort M overlapped triangles by their distance to camera according to their depth value

    $O_{T_1} = 0$

    for $i = 2 \rightarrow M$ **do**

      if $O_{T_i} = NaN$

        $O_{T_i} = \left(1 - \frac{Area(T_j \setminus \cup_{m=1}^{j-1} T_m)}{Area(T_j)}\right), VA_{T_i} = (T_j \setminus \cup_{m=1}^{j-1} T_m)$

        continue

      else

        if $Area((\cup_{m=1}^{j-1} T_m) \cup T_j) = Area(\cup_{m=1}^{j-1} T_m)$

          $O_{T_j} = 1$

        else

          $O_{T_i} = \left(1 - \frac{Area(VA_{T_j} \setminus \cup_{m=1}^{j-1} T_m)}{Area(T_j)}\right), VA_{T_i} = (T_j \setminus \cup_{m=1}^{j-1} T_m)$

        end if

      end if

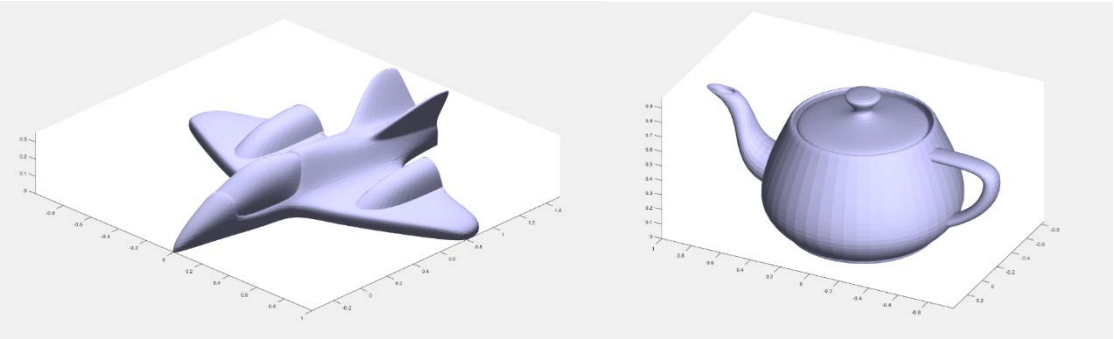
    end for

  end if

end for

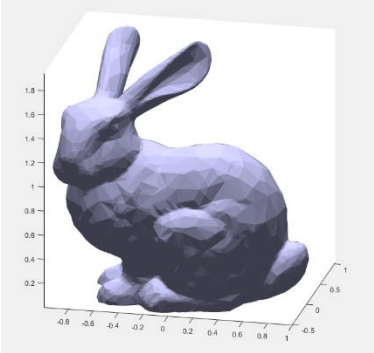
4.3 Simulation of the Coverage Model in Khepra

The *Khepra Simulation Environment* was used to inspect several different 3D CAD models that represent real world tasks. Coverage is visualized and is compared to the model proposed by Zhang el al. [21] , which is a model that represents the state of art.



(a) plane

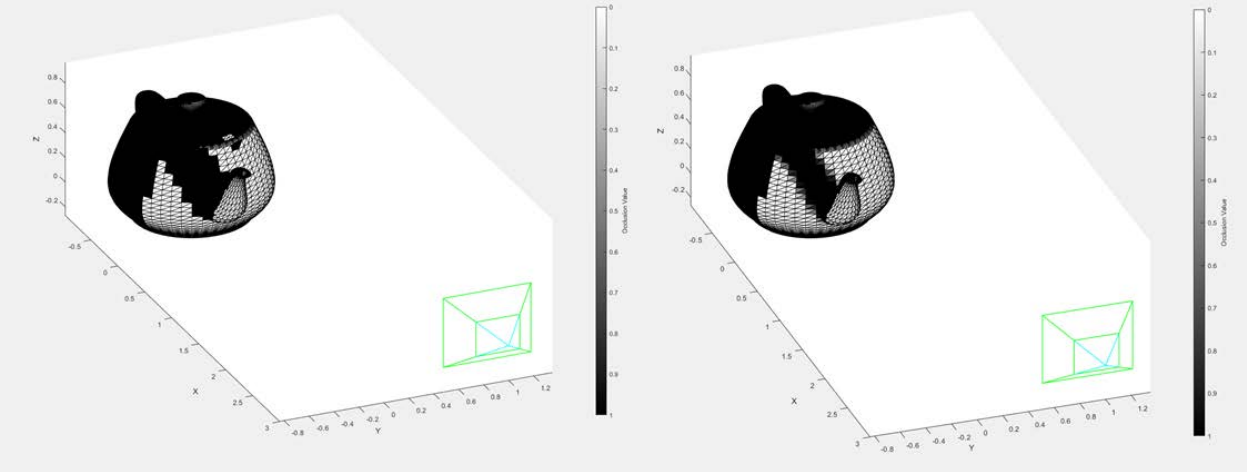
(b) teapot



(c) bunny

Figure 4.1: 3D CAD Models of different shapes

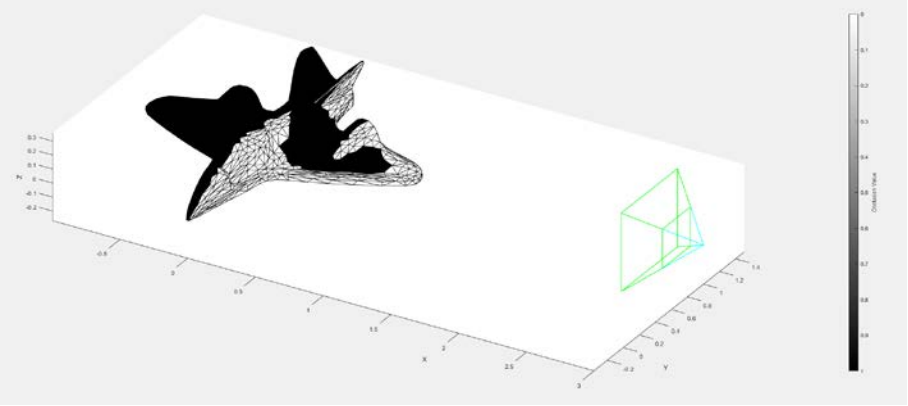
These models were selected because they are sufficiently complex, in which they will represent a good task to visualize and test the performance of the proposed model. In Figure 4.2, a comparison between Zhang's bivalent occlusion criterion (a) and the proposed partial occlusion criterion (b) from (3.12) is made. A camera, which is represented by the green frustrum in the scene, is positioned to observe the teapot, the pose of the camera is exactly the same in the two scenarios.



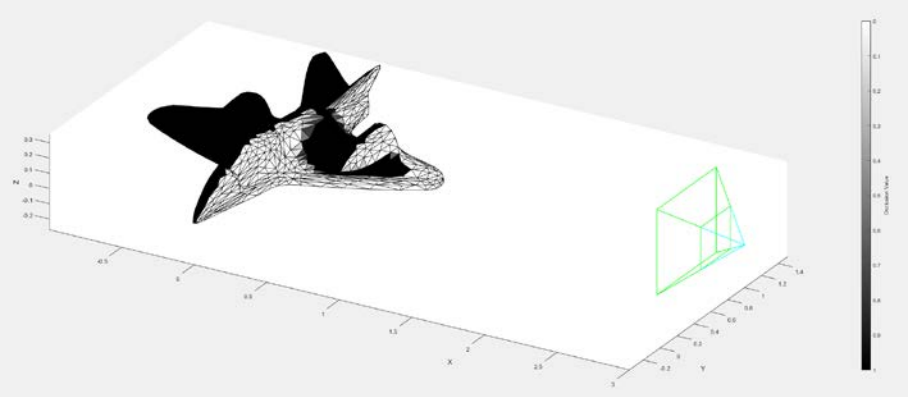
(a) Zhang's Bivalent Occlusion (b) Proposed Graded Occlusion

Figure 4.2: Brighter triangles means higher coverage values while darker means lower coverage

By observing the proposed model at (b), we can see that the triangles that are partially occluded by the teapot's spout are assigned a partial occlusion (grey) value according how much of their surface areas are visible to the camera, while in (a), Zhang's model was unable to capture these details.



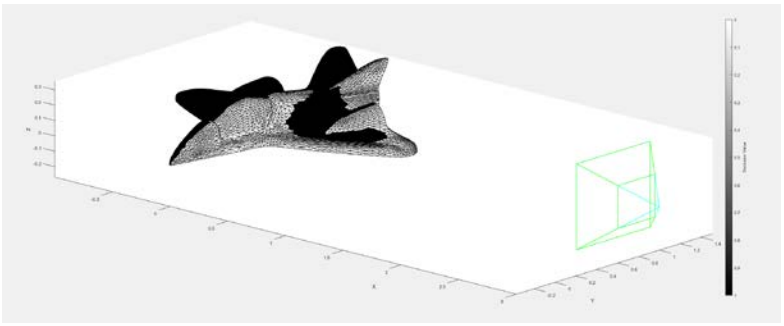
(a) Zhang's Bivalent Occlusion



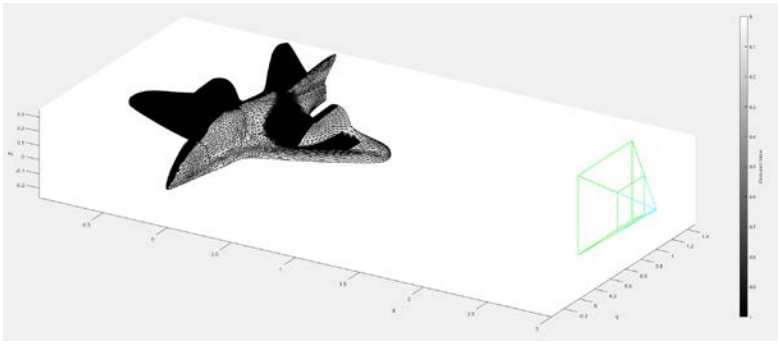
(b) Proposed Graded Occlusion

Figure 4.3: Brighter triangles means higher coverage values while darker means lower coverage

Moving on the plane mode in Figure 4.3, the same difference can be observed in the area that is behind the right engine. We can see that in (a), that area on the aircraft body is completely black, which means that it was assigned 100% occlusion value, even though the majority of the surfaces of these triangles are visible to the camera. In (b) we can see that these triangles were assigned a partial occlusion value.



(a) Zhang's Bivalent Occlusion



(b) Proposed Graded Occlusion

Figure 4.4: The same plane model but with a higher number of triangles

Figure 4.4 shows a resolution model version of the plane, the difference is still observable.

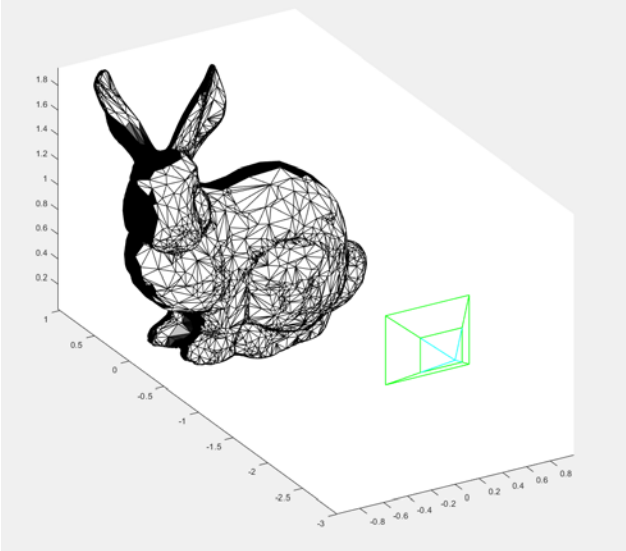


Figure 4.5: A bunny model with the graded visibility criterion

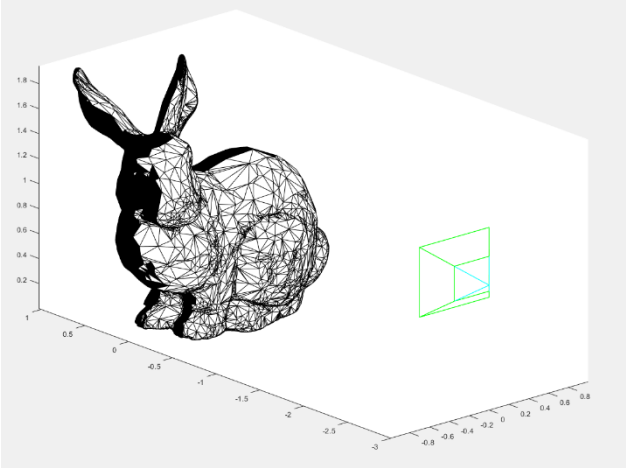


Figure 4.6: A bunny model with the bivalent visibility criterion

Figures 4.5 and 4.6 show occlusion comparison between the two models. Note in Figure 4.5 the grey triangles on the base of the ears and between front legs, which are missing in Figure 4.6.

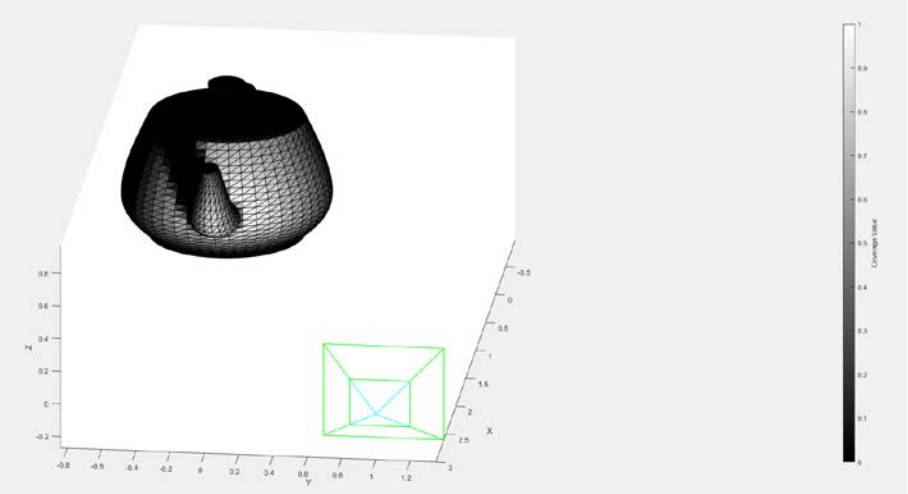


Figure 4.6: Overall coverage on the teapot model

Figure 4.5 shows overall coverage function (3.6), notice that the middle portion of the teapot has a higher coverage value due to the resolution criterion, where this region is covered by more pixels than the other darker areas.

4.3.1 Simulation Time Cost Report and Comparison

Execution time cost of the simulation was measured using stopwatch timer functions in MATLAB: $tic()$ and $toc()$. These functions are solely dedicated for measuring performance [30] . The simulations in this chapter were executed on a general-purpose personal computer. The specifications of this computer are shown below:

Table 4.1: Specifications of the computer used

System Type	64-bit Windows 10
Processor	Intel i7-4790 @ 3.60GHz
Processor Cache	8 MB Intel® Smart Cache
Main Memory	16GB DDR3
Hard Drive	250GB SSD SATA
Graphics	Nvidia GeForce GTX 970

The overall coverage functions of the proposed model and Zhang's model were executed on seven different 3D CAD models for ten consecutive times each. In Table 4.2, statistical time data were recorded such as mean time, best time, worst time and standard deviation. The tested 3D CAD models had different triangle count ranging from low to high.

Table 4.2: Time Cost Statistics of Ten Executions in Seconds

Model Total Triangles	Plane1 1970	Plane2 9856	Teapot1 1560	Teapot2 6400	Teapot3 9216	Bunny1 1602	Bunny2 5122
Mean							
Proposed Model	17.776	307.995	9.029	142.892	286.261	9.181	109.563
Zhang's Model	18.396	426.600	7.997	163.233	325.406	11.230	168.456
Best							
Proposed Model	17.286	298.928	8.372	136.940	272.249	8.929	106.710
Zhang's Model	16.935	382.31	7.725	158.462	317.406	10.835	159.174
Worst							
Proposed Model	19.190	322.471	9.609	146.289	294.230	9.819	111.953
Zhang's Model	21.409	464.805	8.469	168.545	340.909	11.972	174.744
Standard Deviation							
Proposed Model	0.5933	7.427	0.447	2.521	6.411	0.311	1.649
Zhang's Model	1.851	29.319	0.262	3.085	8.086	0.393	6.428

We can see clearly that the proposed model is faster in all models except for *Teapot1*, the model with the lowest triangle count. A trend can be seen that when the model has low triangle count, performance of both models seems to be close, but when triangle count increases, the proposed model becomes more efficient. The proposed model was 35% faster in evaluating overall coverage of the *Bunny2* model.

An important aspect that needs to be pointed out that the proposed model implementation is weighed down by MATLAB's Boolean operations internal library, a much faster time is expected to be achieved if these operations were implemented from scratch. Zhang's model implementation does not suffer from this pitfall because it does not depend on any slow MATLAB libraries.

CHAPTER 5

Experimental Validation

5.1 Overview

In this chapter, we'll conduct experimental validation of the coverage model that was presented in Chapter 3. Firstly, the relationship between the performance criteria that were discussed in the previous chapter and performance of the task in a real-world scenario is examined and then verified. A visualization software, 3ds Max was used to produce images for the experiment that are equivalent to real-life photos [31] .

5.2 Validation of Model Criteria

The ArUco Augmented Reality library, which is based on the OpenCV computer vision software library, can detect, identify and estimate the three-dimensional pose of a fiducial marker with sub-pixel accuracy [32] . The ArUco library is capable of such detection just by a single image of the fiducial marker depicted in Figure 5.1.

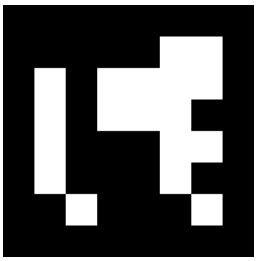


Figure 5.1: ArUco marker with ID=1, an example of a 6x6 fiducial marker

A fiducial marker such as the ArUco marker is a square marker composed of a black border region with a binary code matrix inside it. The binary code determines the marker identifier or number. The size of

the marker also determines the size of the binary matrix and the number of unique markers we can generate.

For example, a marker size of 4x4 is made up of 16 bits. A single marker provides enough information to estimate the camera pose by its four corners, while the inner binary code allows us to determine marker identification and rotation in space [32] .

In order for the ArUco library to successfully detect a marker, the size of the marker perimeter in the inspected image should be at least equal to or larger than a predefined marker perimeter size, which we'll denote with the following symbol, P_{min}, this parameter is configurable by the user. In our experiment, we set it to 128 pixels, so any marker with a perimeter size less than 128 pixels, will not be detected by the ArUco library. A marker with a perimeter of 128 pixels is composed of 1024 pixels. So, we set the resolution parameter R_{min}, which is the minimum required resolution per marker to 1024 pixels.

Also, the library requires that all four corners of the marker be visible for detection, so partial occlusion will present a problem in this task, so we set minimum acceptable visibility per marker $\mathbf{V}_{min}$ to 1.

Focus also affects the detection process, so severely out of focus views should be discarded. The minimum blur circle diameter for detection $\mathbf{C}_{min}$=5.755, so any marker that that has bigger blurriness circle will not be detected. From the above requirements, we can summarize the set of task parameters for the detection task using our model as follows:

Table 5.1: Requirements for marker detection task

Parameter	R_{min}	C_{min}	V_{min}
Value	1024	5.755	1

5.2.1 Simulated Validation Setup

3ds Max is a 3D computer graphics software that is used in making visualizations, 3D animations and video games. It can simulate real life physical cameras and lighting effects and produce photorealistic imagery [31] . 3ds Max was used in this experiment to simulate taking pictures of the ArUco marker task using a simulated camera. The virtual camera parameters were selected after (iCube NS4133BU) camera with 8mm (K0740) lens. Table 5.2 presents the specifications of the camera.

Table 5.2: Specifications for iCube NS4133BU camera

Sensor Resolution	1280 ×1024 pixels
Sensor Size	1/1.8”
Pixel Size	5.3µm × 5.3µm
Focal Length	8mm

In the simulation, a group of 30cm plates with markers placed on them were used. The camera itself is placed at height 1.44 meters from the ground. The simulation setup is shown in Figure 5.2. The plates position with respect to camera and their number can be manipulated with ease to produce any number of required images. One advantage of conducting this particular experiment as a simulation as opposed to a real camera setup, is that cameras and tasks can be placed in the environment with absolute accuracy, so we will not need to worry about camera placement error. A wooden table was placed in the scene to provide a sense of scale.

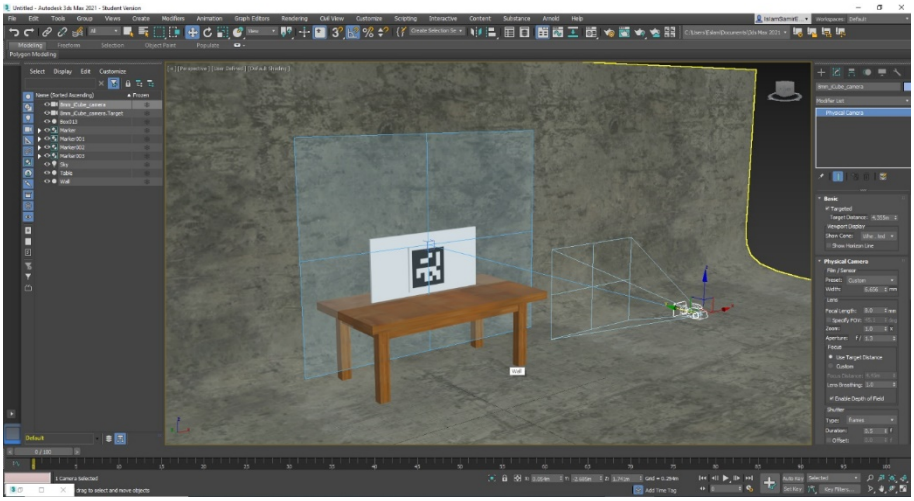


Figure 5.2: Simulation setup on 3ds Max, the camera is placed at a distance 4 meters from a 30cm ArUco marker

5.2.2 Field of View

A total of fifteen ArUco markers were placed at a distance 4 meters from the camera and perpendicular to the camera’s optical axis. Then, the markers were translated to positions that are fully inside, partially inside and fully outside the camera’s field of view. Each marker represents a testcase. The field of view coverage $\boldsymbol{C_F(M_i)}$, ($\mathbf{M_i}$ is for marker) is evaluated according to model formula using MATLAB, and the ArUco marker detection algorithm was executed on the image produced from 3ds Max, successful detections of markers are recorded.

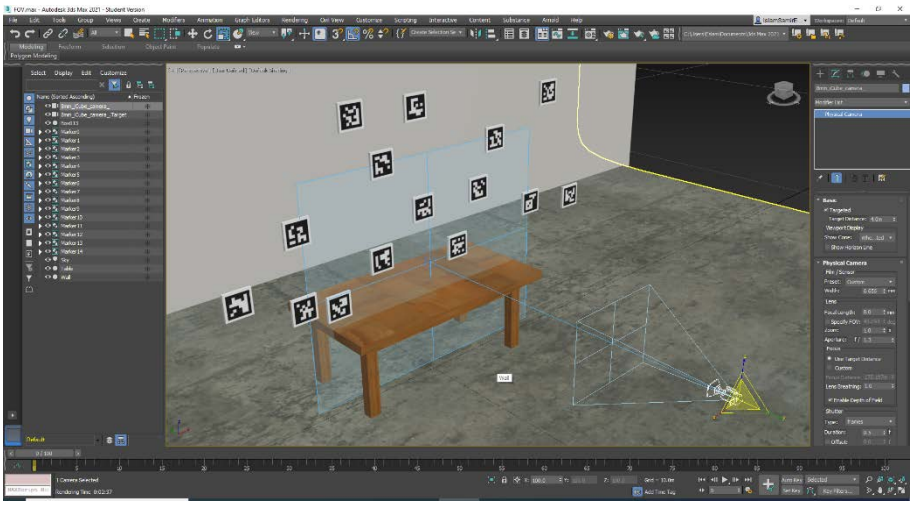


Figure 5.3: FOV Simulation setup on 3ds Max, 15 markers are placed at different positions as test cases.

The field of view simulation setup is shown in Figure 5.3. The blue grid plane represents the virtual camera's field of view.

The field of view simulation setup is shown in Figure 5.3. The blue grid plane represents the virtual camera's field of view. Table 5.3 compares the calculated FOV coverage by the formula vs the actual detection of the marker task by ArUco library. The output image that was produced by 3ds Max is shown at Figure 5.4. That image was processed by the ArUco library and detected markers were highlighted with a green border and a blue label with each marker's identifier, as shown in Figure 5.5.

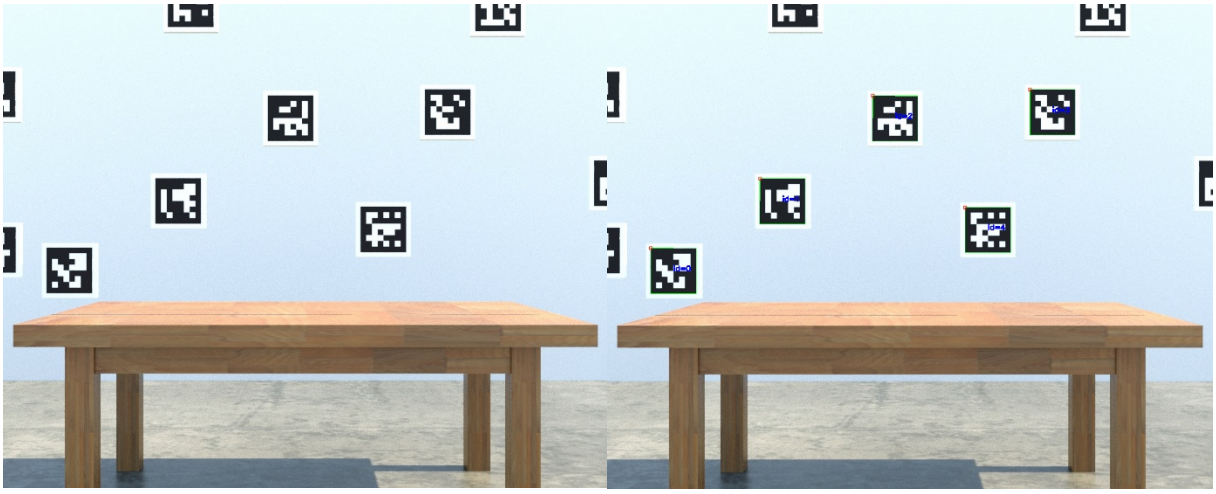


Figure 5.4: (Left) Test image is produced by 3ds Max, (Right) Image shows markers with IDs of 0,1,2,3 and 4 successfully detected by ArUco library

The results of the formula were evaluated by MATLAB and compared to the detection results from ArUco. By looking at Table 5.3, we can see that the model successfully predicted to the performance of the marker detection task, a coverage value of 1 is detected and the opposite is for 0.

Table 5.3: Field of view validation report

Marker ID #	Location	Calculated $C_{FOV}(M_i)$	Detection Result	Test Verdict
0	Fully Inside	1	Detected	Pass
1	Fully Inside	1	Detected	Pass
2	Fully Inside	1	Detected	Pass
3	Fully Inside	1	Detected	Pass
4	Fully Inside	1	Detected	Pass
5	Partially Inside	0	Not Detected	Pass
6	Partially Inside	0	Not Detected	Pass
7	Partially Inside	0	Not Detected	Pass
8	Partially Inside	0	Not Detected	Pass
9	Partially Inside	0	Not Detected	Pass
10	Fully Outside	0	Not Detected	Pass
11	Fully Outside	0	Not Detected	Pass
12	Fully Outside	0	Not Detected	Pass
13	Fully Outside	0	Not Detected	Pass
14	Fully Outside	0	Not Detected	Pass

5.2.3 Resolution

To validate the resolution criteria, fifteen markers were used. The markers were placed at different distances from the virtual camera, the 1st marker with ID=0 was placed at a distance 4 meters from the camera and perpendicular to the camera's optical axis. The remaining markers are then placed 1 meter further from the one preceding it, the 15th marker with ID=14 would be placed at a distance 18 meters from the camera. Figure 5.5 depicts the resolution simulation setup.

As stated before, the ArUco library is configured to only detect markers whose perimeters are equal to or larger than 128 pixels. Figure 5.6 (right) depicts the input image that was produced by 3ds Max, the depth of effect was disabled in the virtual camera parameters to prevent the far markers from being out of focus. The same effect can be achieved in real-life experiment by decreasing the aperture size of the lens used.

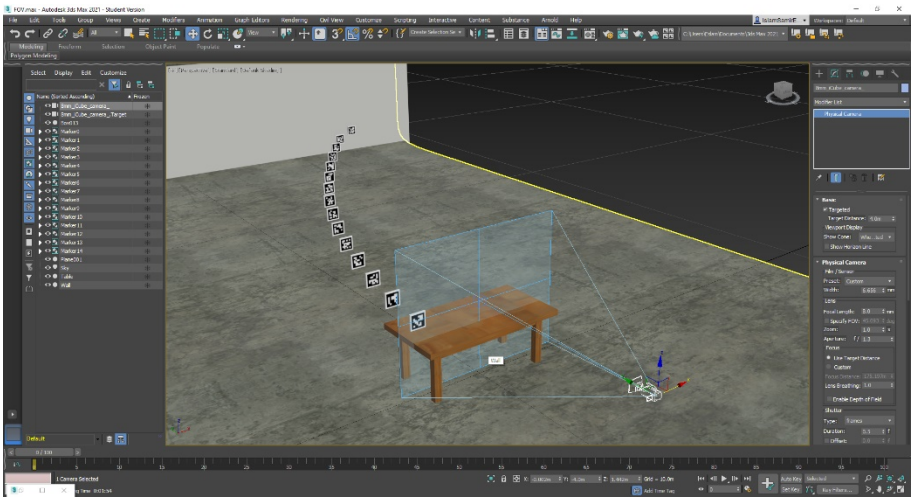


Figure 5.5: Resolution simulation setup on 3ds Max, 15 markers are placed at different distances from the camera, each is further by 1 meter.

Figure 5.6: (Right) image produced by 3ds Max, (Left) first nine markers were only detected

Figure 5.6 (left) shows that ArUco was successful in detecting the first 10 markers, which is the same outcome that the model predicted. Table 5.4 shows the results of the calculated resolution coverage $\boldsymbol{C_R(M_i)}$ vs the actual task performance.

Table 5.4: Resolution validation report

Marker ID #	Distance from Camera (m)	Calculated Pixels/Marker	Calculated $C_R(M_i)$	Detection Result	Test Verdict
0	4	13389	13.07	Detected	Pass
1	5	8281	8.08	Detected	Pass
2	6	5776	5.64	Detected	Pass
3	7	4225	4.12	Detected	Pass
4	8	3136	3.06	Detected	Pass
5	9	2500	2.44	Detected	Pass
6	10	2116	2.06	Detected	Pass
7	11	1764	1.72	Detected	Pass
8	12	1482	1.44	Detected	Pass
9	13	1296	1.26	Detected	Pass
10	14	961	0.94	Not Detected	Pass
11	15	900	0.89	Not Detected	Pass
12	16	841	0.82	Not Detected	Pass
13	17	784	0.76	Not Detected	Pass
14	18	676	0.66	Not Detected	Pass

5.2.4 Focus

Moving on to the focus criterion, the aperture size of the virtual camera was adjusted to get an image with high depth of field. The aperture size we used in this test is f/0.25, expressed as f-number. Fifteen markers are positioned one meters from each other, while the first marker is placed at a distance of two meters from the virtual camera.

In this test, we'll configure ArUco library to detect markers as small as 100 pixels, to isolate detection to focus blurriness conditions only, so we set P_{min}= 38 pixels. We calculate the focus coverage $C_F(\mathrm{M}_i)$ and compare the performance of the ArUco library vs model prediction in Table 5.4. We can see the model results reflect the experiment result, we only have one false negative at marker #0, where the ArUco library managed to detect it despite being blurred.

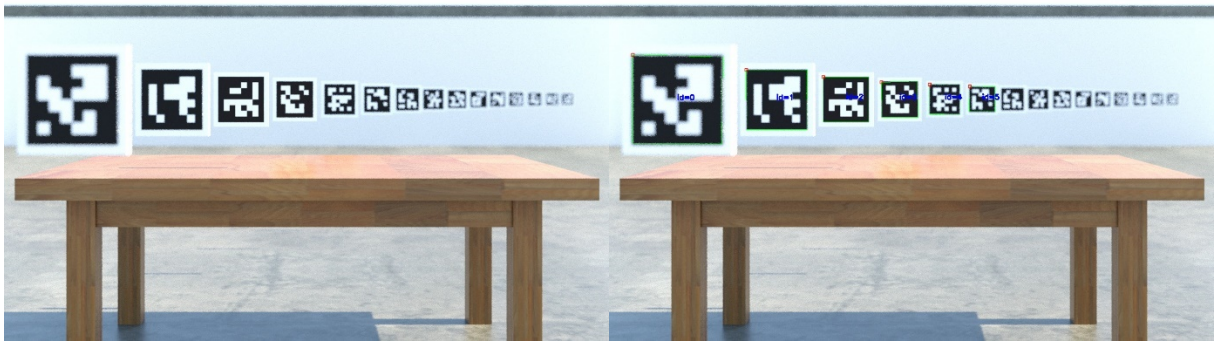


Figure 5.8: (Right) image produced by 3ds Max with a large aperture virtual camera, (Left) first six markers only were detected

Table 5.4: Focus validation report

Marker ID #	Distance from Camera (m)	Calculated $C_F(M_i)$	Detection Result	Test Verdict
0	2	0	Detected	Fail
1	3	1	Detected	Pass
2	4	1	Detected	Pass
3	5	1	Detected	Pass
4	6	1	Detected	Pass
5	7	1	Detected	Pass
6	8	0	Not Detected	Pass
7	9	0	Not Detected	Pass
8	10	0	Not Detected	Pass
9	11	0	Not Detected	Pass
10	12	0	Not Detected	Pass
11	13	0	Not Detected	Pass
12	14	0	Not Detected	Pass
13	17	0	Not Detected	Pass
14	18	0	Not Detected	Pass

5.2.5 Visibility

In this test, fifteen markers were positioned to be fully visible, partially visible or fully occluded by each other, The setup shown in Figure 5.9 will be used to validate the partial occlusion criterion and algorithm that were proposed in Chapter 3.

Each marker has a binary matrix that enable us to read its identifier, Marker #0 was positioned so that it occludes 25% of the surface area of marker #1, marker # 2 occludes 50% of marker #3. Marker #4 occludes 75% of marker #5 and so on. The actual occlusion of each marker is stated in Table 5.5.

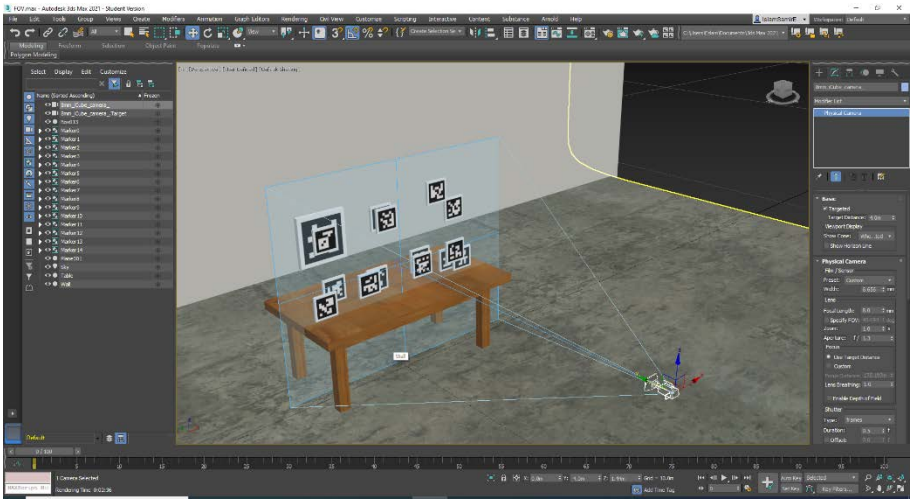


Figure 5.9: Visibility simulation setup on 3ds Max

Figure 5.10: (Right) image produced by 3ds Max (Left) only fully visible markers were correctly detected

Figure 5.10 (right) shows the resulting image from 3ds Max, while Figure 5.10 (left) shows that ArUco was able only detect fully visible markers. Visibility and occlusion coverage values $\mathbf{C_V(M_i)}$ and $\mathbf{C_O(M_i)}$ and were computed in MATLAB according to model and reported in Table 5.5. Actual visibility denotes

the actual area that is visible from each marker. Detection result shows which marker was successfully detected by ArUco library. The validation report shows that only markers with 100% visibility were detected, which reflects expected reality.

Table 5.5: Visibility validation report

Marker ID #	Actual Visibility	Calculated $C_O(M_i)$	Calculated $C_V(M_i)$	Detection Result	Test Verdict
0	100%	0	1	Detected	Pass
1	75%	0.25	0.75	Not Detected	Pass
2	100%	0	1	Detected	Pass
3	50%	0.5	0.5	Not Detected	Pass
4	100%	0	1	Detected	Pass
5	25%	0.75	0.25	Not Detected	Pass
6	100%	0	1	Detected	Pass
7	75%	0.25	0.75	Not Detected	Pass
8	25%	0.75	0.25	Not Detected	Pass
9	100%	0	1	Detected	Pass
10	75%	0.25	0.75	Not Detected	Pass
11	100%	0	1	Detected	Pass
12	0%	1	0	Not Detected	Pass
13	100%	0	1	Detected	Pass
14	100%	0	1	Detected	Pass

Part II
Application on Coverage Modeling

CHAPTER 6

Segmentation based Camera Deployment

6.1 Overview

In this chapter, the theory behind the coverage model that was proposed in Part II will be put to practice by attempting to solve a camera deployment problem. A shape segmentation-based camera deployment method is proposed.

The main contribution behind this method is that it examines the surface properties of the inspected objects and determines a group of camera poses to maximize coverage while keeping the number of cameras used to a minimum. A simulation of the proposed method was carried out using the *Khepra* tool where it was able to report good performance.

6.2 Shape Segmentation

Before we progress through the chapter, we need to first to establish the notion of shape segmentation. According to Chen et al. "Automatic segmentation of 3D surface meshes into functional parts is a fundamental problem in computer graphics". They consider that segmentation of a given 3D shape is the process of decomposing it into smaller segments or parts that are useful enough to achieve a given task [33] .

Shape segmentation have applications in the fields of computer vision and computer graphics. These applications include mesh reconstruction [34] , mesh simplification [35] , texture mapping [36] and skeleton pose estimation [37] .

6.3 Camera Deployment: Shape Segmentation Approach

In the proposed shape segmentation approach, the inspected task is divided into different segments according to the task surface properties, where triangles with similar normal vector direction are

merged into the same segment. The average normal direction of each section is used to get the optimal camera pose to cover that particular section.

The notion behind such approach is avoiding exhaustive search approaches utilized in some of the methods in literature, such as the method proposed by Alarcon [17] , in which a candidate camera pose is generated for each triangle in the task. The latter approach gives us a solution space that's equal to the number of triangles N_t. So, in this case, coverage will have to be evaluated N_t times, in contrast to the proposed method, where coverage only needs to be evaluated once for each segment.

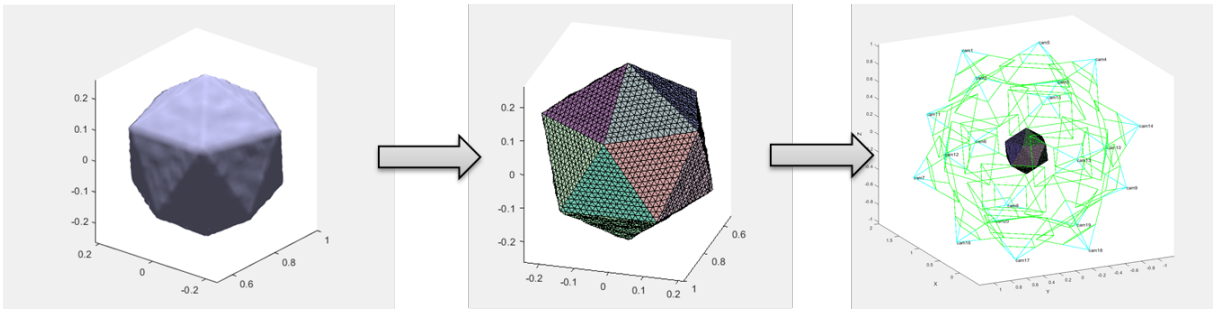


Figure 6.1: Segmentation of an Icosahedron is used to get optimal camera poses for each segment

Figure 6.1 shows the basic process for camera deployment. The shape is first segmented using the recursive mesh segmentation algorithm, then each color-coded segment is assigned a camera pose.

6.3.1 Recursive Mesh Segmentation

In the recursive mesh segmentation method, the condition for merging two triangles T_1 and T_2 is given by:

$$\theta\left(\overrightarrow{N_{T_1}}, \overrightarrow{N_{T_2}}\right) \leq \alpha \tag{6.1}$$

Where $\overrightarrow{N_{T_1}}$ and $\overrightarrow{N_{T_2}}$ are the normal vectors of the two triangles respectively and α is the merging tolerance angle. This means that for the two triangles to be merged into one segment, the angle θ between their normal vectors should be less than or equal α.

Next formula is the condition for merging a given triangle T' with an existing segment S:

$$\max_{i=1}^{n_s} \theta(\overrightarrow{N_{T_i}}, \overrightarrow{N_{T'}}) \leq \alpha \tag{6.2}$$

Where $\theta(\overrightarrow{N_{T_i}}, \overrightarrow{N_{T'}})$ is the angle between the normal vector of a given segment triangle T_i and a new triangle T', α is the merging tolerance angle and n_s is the segment triangle count. This formula is the

same as the previous one, but the difference is that it evaluates the maximum angle difference between triangle T_i and all triangles in segment S.

We test formulas (6.1) and (6.2) to segment an *icosahedron*, which is a polyhedron with 20 faces. Figure 6.2 shows the segmentation result on the icosahedron with merging tolerance angle selected 'α' as 15°. Various segments are distinguished from each other by different colors, we can see that the icosahedron individual faces were segmented perfectly.

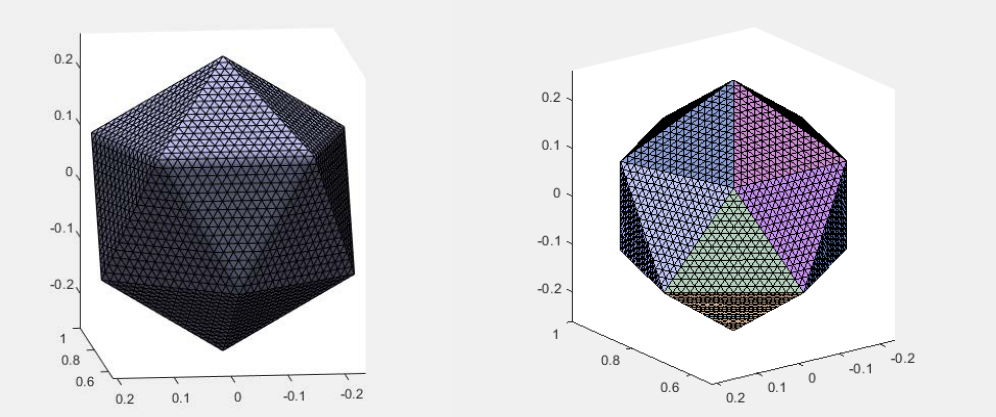


Figure 6.2: Icosahedron individual faces were segmented with α=15° into 20 regions perfectly.

However, a problem presents itself when it comes to noisy surfaces, as formula (6.2) was sensitive to the small variations in the noisy surface that was larger than tolerance angle 'α', so each face was segmented into various smaller segments, as depicted in Figure 6.3.

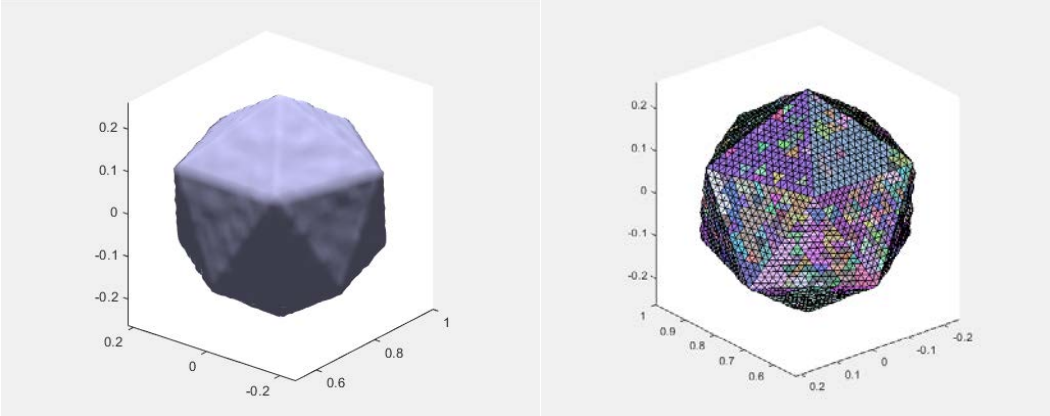


Figure 6.3: Noisy icosahedron segmentation with α=15°, note the segmentation errors due to noise.

Gaussian noise was applied on the icosahedron using the 3ds Max software.

The solution to the local surface noise problem can be obtained by remerging similar segments into one, the formula for merging two segments S_1 and S_2 is given by:

$$\theta\left(\frac{1}{n_{S_1}}\sum_{i=1}^{n_{S_1}}\overrightarrow{N_i^{S_1}}\ ,\frac{1}{n_{S_2}}\sum_{i=1}^{n_{S_2}}\overrightarrow{N_i^{S_2}}\right)\ \leq\ \beta \tag{6.3}$$

Where expressions $\frac{1}{n_{S_1}}\sum_{i=1}^{n_{S_1}}\overrightarrow{N_i^{S_1}}$ and $\frac{1}{n_{S_2}}\sum_{i=1}^{n_{S_2}}\overrightarrow{N_i^{S_2}}$ are the average normal direction vectors of all the triangles in segments S_1 and S_2 respectively. 'β' is segment merging tolerance angle.

The result of this update can be seen in Figure 6.4, where the noisy icosahedron was segmented perfectly with both 'α' and 'β' set to 15°.

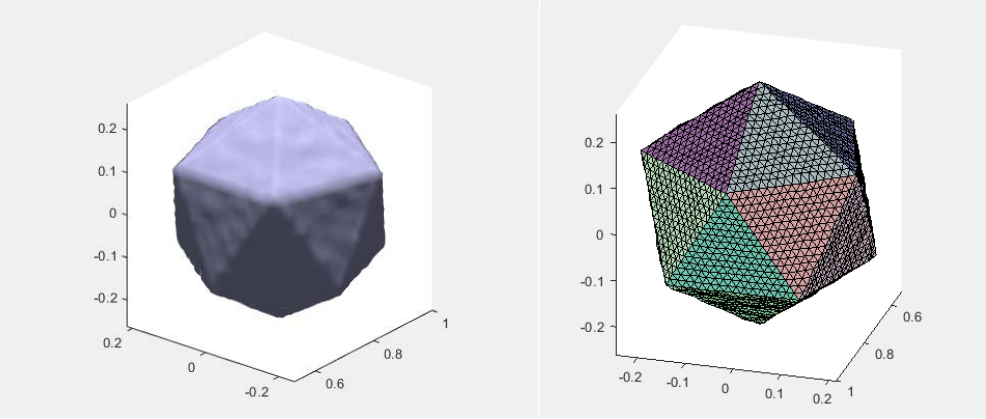


Figure 6.4: Noisy icosahedron was segmented perfectly into 20 segments with α=15°, β=15°

6.3.2 Camera Pose Calculation

To get an optimal camera pose for a given triangle, the camera should be facing the triangle surface in a perpendicular fashion from a distance 'z_n', which is the near limit of the camera's depth of field. We extend on this concept, in the sense that we try to find a camera pose that faces the average perpendicular direction of each segment.

To find the pose, we make use of a "*Look At*" or "*Observer to Target*" transformation matrix, which is a 4×4 transformation matrix that belongs to Lie group SE(3). "*Observer to Target*" transformation matrix is heavily used in computer graphics applications and videogames to have virtual cameras looking at or following a specific subject. The *Look At* function only needs the Cartesian coordinates of two points in space representing the location of the observer (camera) and the target (task) to build a transformation matrix that describes the pose of the observer looking at the target.

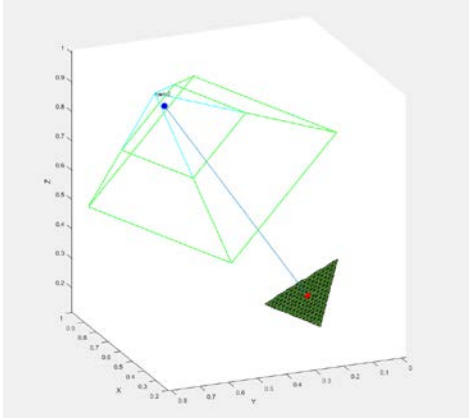


Figure 6.5: Blue dot is "*Observer*", Red dot is "*Target*", the blue line is the average normal direction of all triangles in segment

Figure 6.5 shows two points representing the position of the observer and target, and the green frustrum shows the pose built by the *Look at* function, which is looking at a cutoff segment from the noisy icosahedron. For a given segment S, target point position T_S is given by:

$$T_S = \frac{1}{n_s}\sum_{i=1}^{n_s} v_i \tag{6.4}$$

The above formula describes the average position point of all the vertices in segment S.

To get observer point position, we first need to find the average normal vector $\overrightarrow{V_{avg}}$ of all the triangles in the segment S, which is given by:

$$\overrightarrow{V_{avg}} = \frac{1}{n_S}\sum_{i=1}^{n_S} \overrightarrow{N_i} \tag{6.5}$$

The observer point position O_S is given by:

$$O_S = \overrightarrow{V_{avg}} + \frac{Z_{min}\overrightarrow{V_{avg}}}{|\overrightarrow{V_{avg}}|} \tag{6.6}$$

$$Z_{min} = \frac{W cot(\frac{HFOV}{2})}{2} \tag{6.7}$$

where Z_{min} is a distance scaling factor, 'W' is the maximum width of a given segment, $HFOV$ is the horizontal field of view of angle of the camera. The factor $Z_{min}$was added to keep observer point at a

sufficient height for the field of view of the camera to cover the whole segment. cot () function is cotangent function.

Given "*Observer*" O_S and "*Target*" T_S location points, the Look At pose is given by:

$$P_S(O_S, T_S) = \begin{bmatrix} X & Y & Z & O_S \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.8}$$

where X, Y and Z are $SO(3)$ rotation matrices:

$$Z = \frac{T_S - O_S}{||T_S - O_S||} \tag{6.9}$$

$$X = \frac{Up \times Z}{||Up \times Z||} \tag{6.10}$$

$$Y = \frac{Z \times X}{||Z \times X||} \tag{6.11}$$

Where "Up" is a reference vector that point for the up direction, the positive *Z-axis* is usually selected at the direction for up, so the vector is given value: $Up = [0,0,1]$.

6.4 Simulation Using Khepra

The *Khepra Simulation Environment* was used to conduct segmentation and camera deployment for two 3D CAD models that represent real world tasks. Segmentation and camera networks are visualized and the effects of changing segmentation tolerance angles 'α' and 'β' on the number of cameras deployed and the overall network coverage are shown. Time cost vs coverage performance results are also reported.

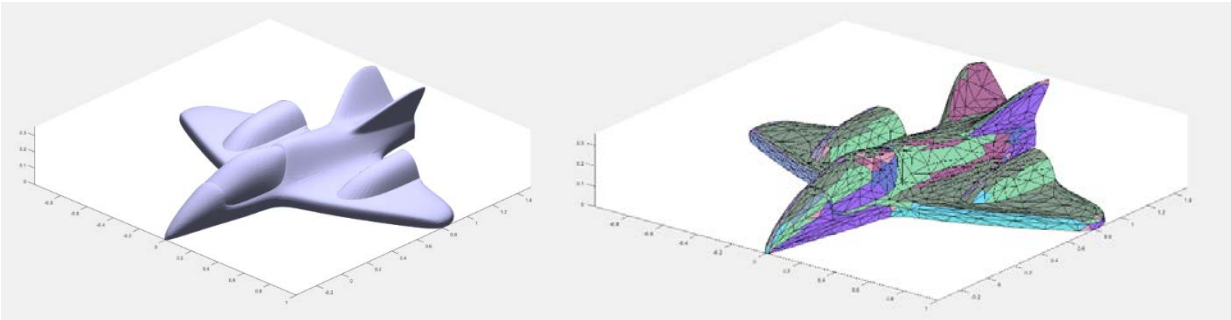


Figure 6.6: Segmentation on 1970 triangle plane, divided into 30 segments, $\alpha = 15,\ \beta = 25$

In Figure 6.6, we can see that the plane was segmented, where each region is color-coded, choosing tolerance angles $\alpha = 15,\ \beta = 25$ resulted in 30 segments.

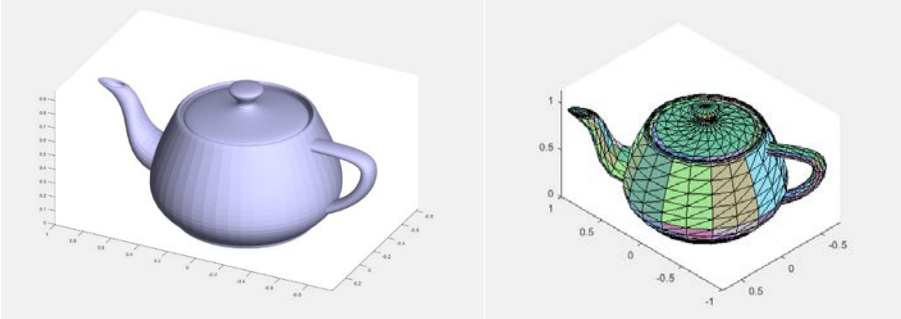


Figure 6.7: Segmentation on 1560 triangle teapot, divided into 37 segments, $\alpha = 15,\ \beta = 25$

In Figure 6.7, the teapot was divided into 37 segments, with tolerance angles set as $\alpha = 15,\ \beta = 25$

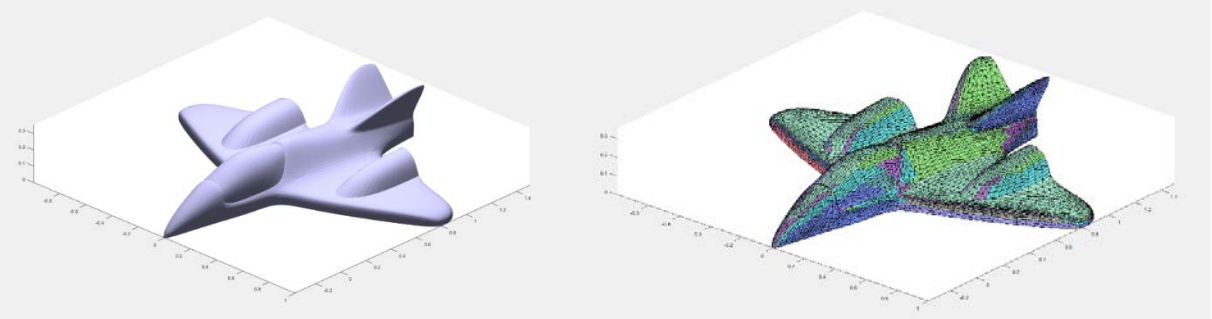


Figure 6.8: Segmentation on 9856 triangle plane, divided into 32 segments, $\alpha = 15,\ \beta = 25$

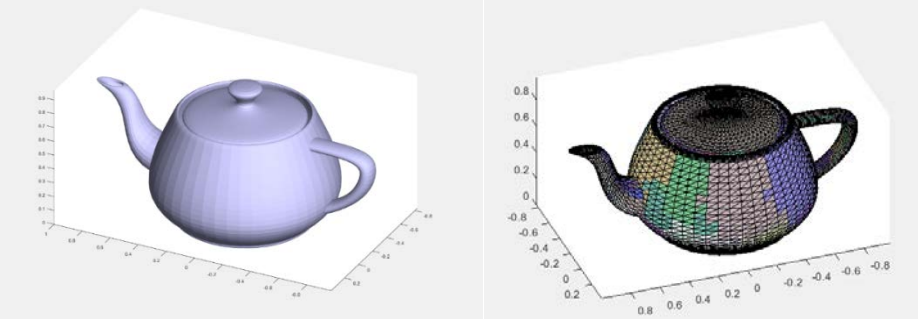


Figure 6.9: Segmentation on 9216 triangle teapot, divided into 42 segments, $\alpha = 15,\ \beta = 25$

Segmentation on a higher resolution models are tested next, as in Figures 6.8 and 6.9 we can see that using the same tolerance angles resulted in similar segment count.

Figure 6.10 depicted the camera network the 1970 triangle plane model, we can see that a camera view was generated for each segment. Figures 6.11 and 6.12 show the camera network changes with respect to changing tolerance angles 'α' and 'β'.

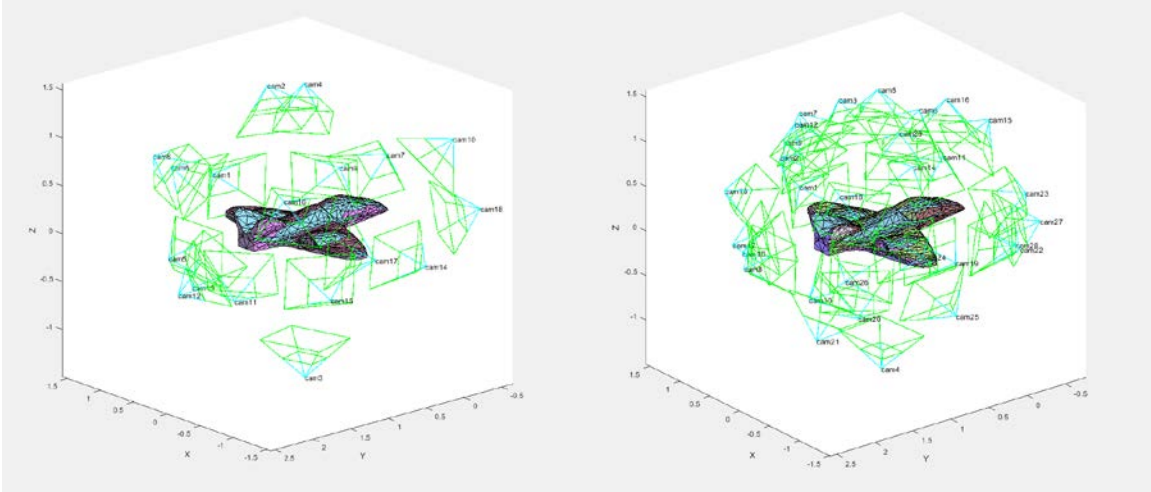


Figure 6.10: Segmentation on 1970 triangle plane, $\alpha = 15,\ \beta = 25$ results in 30 camera poses (left), while $\alpha = 25,\ \beta = 35$ results in 18 camera poses.

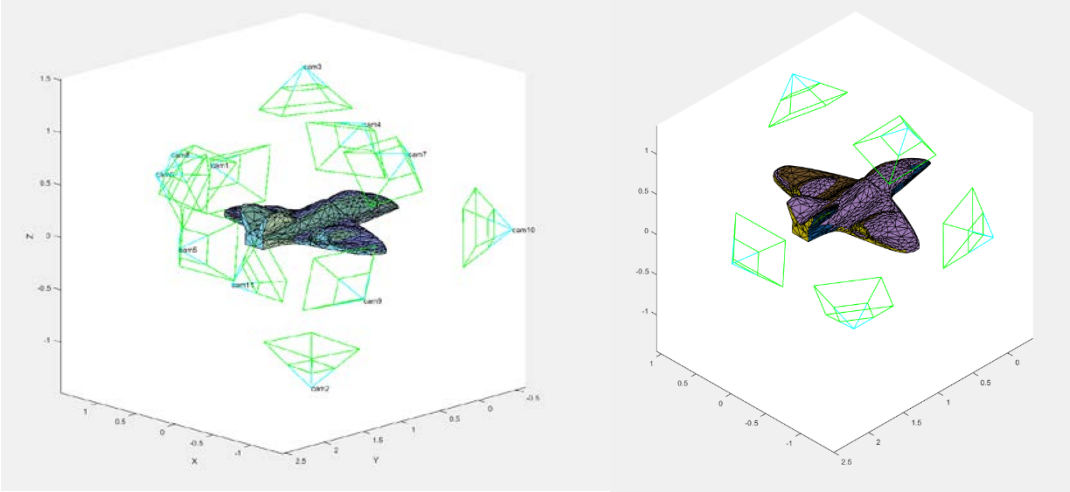


Figure 6.11: Segmentation on 1970 triangle plane, $\alpha = 35,\ \beta = 45$ results in 11 camera poses (left), while $\alpha = 55,\ \beta = 65$ results in 5 camera poses.

6.4.1 Coverage Performance and Simulation Time Cost Report

Network Coverage performance is gauged according to percentage of non-covered triangles with respect to the total number of triangles in the model. The lower the percentage of non-covered triangles, the better is the coverage performance.

In Table 6.1, coverage performance of four generated camera networks are reported along with time cost statistics of ten consecutive executions. It's clear to see that coverage performance is decreased as the number cameras decreases. Higher camera count gave almost perfect performance (0.25% non-covered) in the case of 30 segments, but at the cost of time. While lower segments give lower

performance but better time cost. The case with 6 segments was able to report a good performance (5.63% non-covered) with minimal time cost.

Table 6.1: Time Cost Statistics of Ten Executions in Seconds

Model Total Triangles	Plane1 1970	Plane1 1970	Plane1 1970	Plane1 1970
α	15	25	35	30
β	25	35	45	60
Number of cameras/segments	30	18	11	6
Percentage of non-covered triangles	0.253%	0.71%	2.99%	5.63%
Time Cost (in seconds)				
Mean	549.769	274.418	178.054	100.123
Best	518.567	271.111	170	91.745
Worst	570.156	279.799	191.544	113.270
Standard Deviation	18.155	3.1734	7.664	6.293

Conclusion

CHAPTER 7

Conclusions

7.1 Summary of Contributions

This thesis presents three main contributions. The first contribution is development of an image space-based coverage model which was presented in Subsection 3.3, also a graded occlusion evaluation algorithm was proposed in Subsection 4.2.

In Chapter 4, the proposed coverage model was simulated using the *Khepra Simulation Environment,* in which it reported superior coverage representation accuracy and time cost versus the state of the art. In Chapter 5, the model was validated against a real-world vision task. Individual model criteria were tested against testcases composed of images that were produced using 3ds Max visualization software.

The second contribution is development of a mesh segmentation-based camera deployment method. The method was tested on two 3D CAD models and excellent coverage performance was reported, almost reaching 100% covered triangles in the case of 30 segments.

The third contribution is the development of *Khepra Simulation Environment.* Khepra is a stand-alone, user friendly coverage planning and simulation tool with a complete GUI. It was developed on MATLAB R2018a and is capable of importing STL format 3D CAD files.

7.2 Conclusions

By examining the results reported in this thesis, we can observe that the Image Space Coverage Model represents an adequately accurate and efficient model for modeling vision systems. Reformulation of the geometry of traditional coverage criteria into image space have simplified its complexity and reduced coverage evaluation time cost. Experiments and comparisons with previous models substantiate the claims made above.

The shape segmentation-based camera deployment method, which uses a novel approach of examining the task surface properties, was tested on various 3D CAD shapes and was shown to achieve positive coverage performance and time cost.

7.3 Directions for Future Work

When it comes to optimization of visual camera networks, minimizing occlusion can still represent a major challenge. While the model presented in Chapter 3 provides a new measure to assess graded occlusion and the camera deployment method proposed in Chapter 7 was able to achieve good coverage performance, it wouldn't be able to nullify occlusion completely, especially in 3D models that have cavities. A possible direction for future work is researching a method that can process a given 3D model, and identify occlusion prone areas such as cavities and holes, and then try to mitigate occlusion by finding optimal poses to cover those areas.

Studying topology of 3D CAD file might be a better approach for camera deployment instead of conducting an exhaustive search over a vast solution space via optimization algorithms such as particle swarm algorithms or generic algorithms to find optimal solutions. Additional experiments can be conducted to compare coverage performance of the proposed deployment method against other methods that use exhaustive searching.

Another subject that might be worthy of further investigation is exploring the idea of incorporating a measure of scene lighting quality into the coverage model itself. Sufficient scene lighting is vital prerequisite for a vision system to be able to achieve task coverage. While in this thesis, we didn't account lighting in the model because we assumed good lighting conditions. Another direction is applying the proposed coverage model to another set of tasks such as industrial inspection or failure detection in 3D printing and examine the performance of the proposed method against competition.

Appendix A: Khepra Simulation Environment

A.1 Introduction

Khepra[2] is a three-dimensional simulation environment for planning and visualization of multi-camera networks. It is a standalone user-friendly tool with complete GUI. It is a precise implementation of the Image Space Coverage Model that was described in Chapter 3. It also supports the coverage model that was proposed by Zhang et al. [21] . The latter model was incorporated in Khepra for comparison purposes.

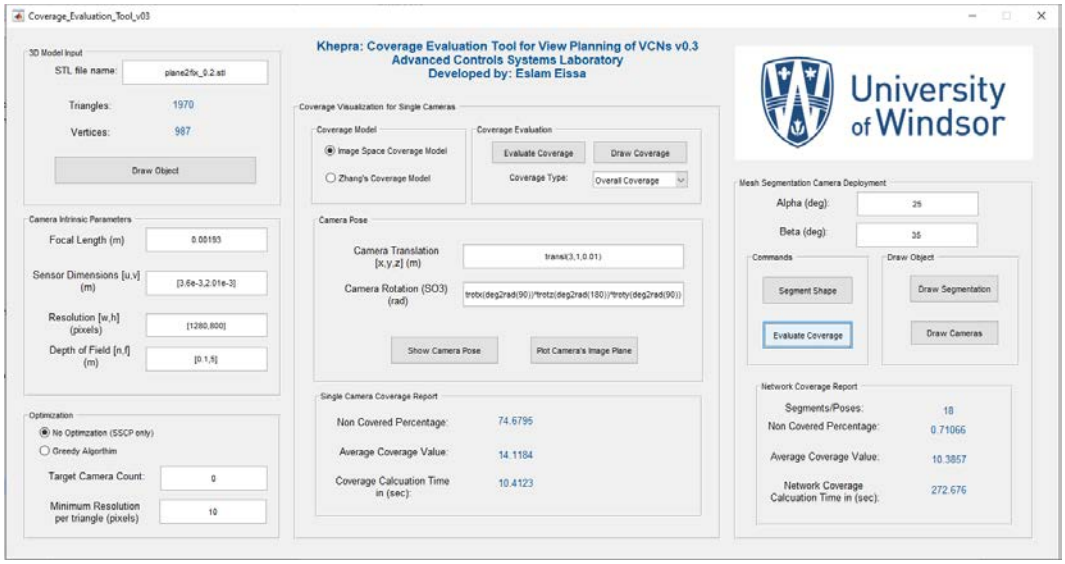


Figure A.1: Khepra's User Interface

The idea behind the tool is to provide an easy way to manage complex camera and model function concepts. Khepra was developed using MATLAB 2018a. MATLAB was selected due to its ability to produce cross platform and standalone applications. Cross platform means that the tool can be released for various platforms such as Windows®, Linux® and Mac with minimal effort [38] . Standalone means that it comes in the form of an executable file and the user doesn't need to have MATLAB installed on his system in order to run it. All of simulation and experimental work in this thesis made use of Khepra.

[2] Named after a god in ancient Egyptian mythology, who represents the light of the morning sun.

A.2 Khepra User Interface

Khepra's interface is divided into five main panels, each serves a dedicated purpose.

A.2.1 3D Model Input Panel

The first panel deals with importing and displaying the 3D task files.

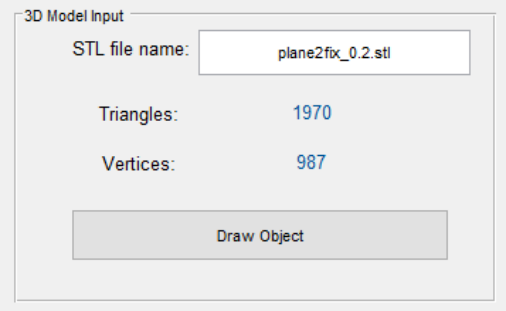


Figure A.2 3D Model Input Panel

The user just needs to type in the name of inspected STL CAD file and Click on *"Draw Object"* to view it. The panel also displays the number of triangles and vertices of the model.

A.2.2 Camera Intrinsic Parameters Panel

The second panel allows the user to specify the camera's focal length, sensor dimensions, resolution and depth of field limits. The data should be input in meters.

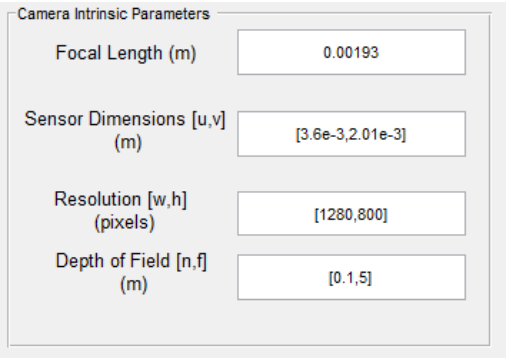


Figure A.3 Camera Intrinsic Parameters

A.2.3 Optimization Panel

The third panel allows the user to specify whether to perform greedy optimization on the output of the shape segmentation camera deployment algorithm or not. The user can specify the needed camera count for the greedy algorithm and also set the resolution task parameter R_{min}.

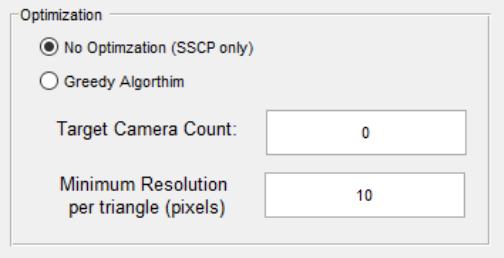


Figure A.4 Optimization Panel

A.2.4 Coverage Visualization for Single Cameras Panel

The fourth panel is dedicated for coverage visualization and evaluation for single cameras. The coverage model can be select from the *"Coverage Model"* subpanel on the upper left corner.

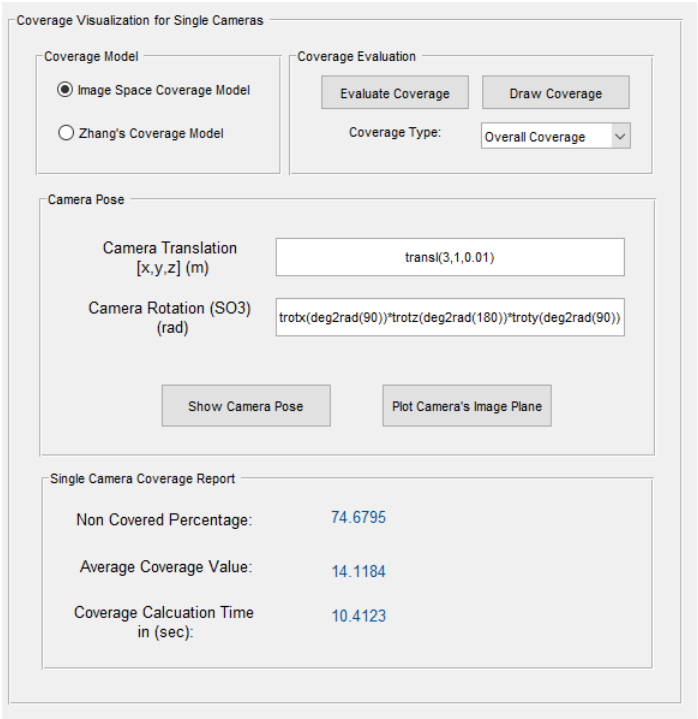


Figure A.5 Coverage Visualization for Single Cameras Panel

The *"Coverage Evaluation"* subpanel allows the user to select which coverage function to evaluation, whether its full coverage or individual coverage functions such occlusion or resolution, etc.

The "*Evaluate Coverage*" button will calculate coverage according to the selected function and "*Draw Coverage*" will draw coverage of each triangle in greyscale colors, where white means 100% coverage and black means 0% coverage.

The middle subpanel, which is dubbed "*Camera Pose*", is used to specify the camera pose. The user can input a 4×4 transformation matrix in MATLAB's syntax to specify the camera's translational and rotational components. The bottom panel is used to report coverage information such as performance and time cost.

A.2.5 Mesh Segmentation Camera Deployment Panel

The last panel is dedicated to mesh segmentation camera deployment evaluation. "*Alpha*" and "*Beta*" are the merging tolerance angles. The "*Commands*" subpanel has the commands for segmenting shapes and evaluating coverage. The "*Draw Object*" subpanel has the button for visualizing individual segments on the CAD model and also a button to draw the camera network.

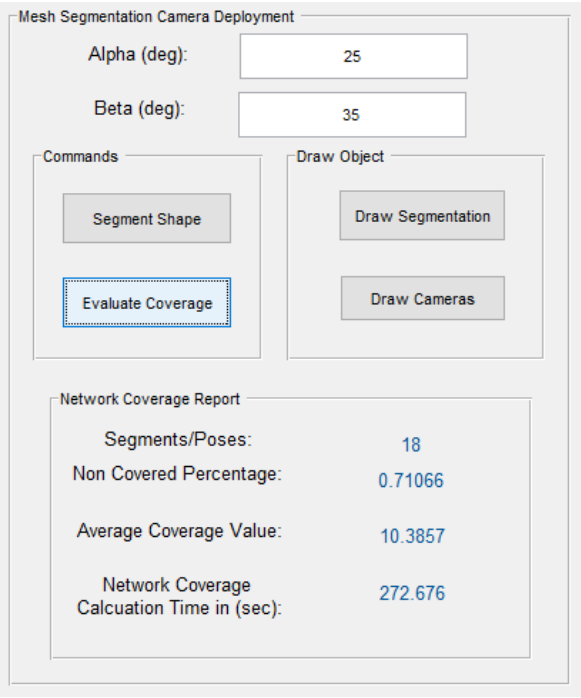


Figure A.6 Mesh Segmentation Camera Deployment Panel

Bibliography

[1] Farzadpour, F. (2018). *A new Measure for Optimization of Field Sensor Network with Application to LiDAR* (Doctoral dissertation, University of Windsor). Electronic Theses and Dissertations. Retrieved May 30, 2020, from https://scholar.uwindsor.ca/etd/7418.

[2] Advanced Sensor Coverage (n.d.), *Tesla.com*. Retrieved from https://www.tesla.com/en_CA/autopilot?redirect=no

[3] Gonzalez, L., Montes, G., Puig, E., Johnson, S., Mengersen, K., & Gaston, K. (2016). Unmanned Aerial Vehicles (UAVs) and Artificial Intelligence Revolutionizing Wildlife Monitoring and Conservation. *Sensors, 16*(1), 97. doi:10.3390/s16010097

[4] Casbeer, D., Li, S., Beard, R., Mehra, R., & Mclain, T. (2005). Forest fire monitoring with multiple small UAVs. *Proceedings of the 2005, American Control Conference, 2005.* doi:10.1109/acc.2005.1470520

[5] Moons, T., Van Gool, L., & Vergauwen, M. (2010). 3D Reconstruction from Multiple Images Part 1: Principles. *Foundations and Trends® in Computer Graphics and Vision, 4*(4), 287-404. doi:10.1561/9781601982858

[6] Wu, C., Aghajan, H., & Kleihorst, R. (2008). Real-Time Human Posture Reconstruction in Wireless Smart Camera Networks. *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008).* doi:10.1109/ipsn.2008.20

[7] Angella, F., Reithler, L., & Gallesio, F. (2007). *IEEE Conference on Advanced Video and Signal Based Surveillance* (pp. 388-392). London: IEEE. doi: 10.1109/AVSS.2007.4425342.

[8] Fu, Y., Zhou, J., & Deng, L. (2014). Surveillance of a 2D Plane Area with 3D Deployed Cameras. *Sensors, 14*(2), 1988-2011. doi:10.3390/s140201988

[9] Mavrinac, A., & Chen, X. (2012). Modeling Coverage in Camera Networks: A Survey. *International Journal of Computer Vision, 101*(1), 205-226. doi:10.1007/s11263-012-0587-7

[10] Ma, H., & Liu, Y. (2007). Some problems of directional sensor networks. *International Journal of Sensor Networks, 2*(1/2), 44. doi:10.1504/ijsnet.2007.012981

[11] Ai, J., & Abouzeid, A. A. (2006). Coverage by directional sensors in randomly deployed wireless sensor networks. *Journal of Combinatorial Optimization, 11*(1), 21-41. doi:10.1007/s10878-006-5975-x

[12] Jiang, Y., Yang, J., Chen, W., & Wang, W. (2010). A Coverage Enhancement Method of Directional Sensor Network Based on Genetic Algorithm for Occlusion-Free Surveillance. *2010 International Conference on Computational Aspects of Social Networks.* doi:10.1109/cason.2010.76

[13] Hörster, E., & Lienhart, R. (2009). Optimal Placement of Multiple Visual Sensors. *Multi-Camera Networks,* 117-138. doi:10.1016/b978-0-12-374633-7.00007-0

[14] Erdem, U. M., & Sclaroff, S. (2003). Automated placement of cameras in a floorplan to satisfy task-specific constraints. Tech. Report, Boston University.

[15] Malik, R., &Bajcsy, P. (2008). Automated placement of multiple stereo cameras. In Proceedings of 8th ECCV workshop on omnidirectional vision, camera networks and non-classical cameras.

[16] Mavrinac, A., Herrera, J. L., & Chen, X. (2010). A fuzzy model for coverage evaluation of cameras and multi-camera networks. *Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras - ICDSC '10*. doi:10.1145/1865987.1866003

[17] Alarcon, J. (2014). *A Measure of Vision Distance for Optimization of Camera Networks* (Doctoral dissertation, University of Windsor). Electronic Theses and Dissertations. Retrieved May 30, 2020, from https://scholar.uwindsor.ca/etd/5199/.

[18] Cowan, C. K., & Kovesi, P.D. (1988). Automatic sensor placement from vision task requirements. IEEE Transactions on Pattern Analysis and Machine Intelligence, 10(3), 407–416.

[19] Tarabanis, K. A., Allen, P. K., & Tsai, R. Y. (1995). A survey of sensor planning in computer vision. IEEE Transactions on Robotics and Automation, 11(1), 86–104.

[20] Piciarelli, C., Micheloni, C., & Foresti, G. L. (2010). Occlusion-aware multiple camera reconfiguration. *Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras - ICDSC '10*. doi:10.1145/1865987.1866002

[21] Zhang, X., Zhang, B., Chen, X., & Fang, Y. (2019). Coverage optimization of visual sensor networks for observing 3-D objects: Survey and comparison. *International Journal of Intelligent Robotics and Applications, 3*(4), 342-361. doi:10.1007/s41315-019-00102-6

[22] Zhang, X., Chen, X., Alarcon-Herrera, J. L., & Fang, Y. (2015). 3-D Model-Based Multi-Camera Deployment: A Recursive Convex Optimization Approach. *IEEE/ASME Transactions on Mechatronics, 20*(6), 3157-3169. doi:10.1109/tmech.2015.2411593

[23] Park, J., Bhat, P. C., & Kak, A. C. (2006). A look-up table-based approach for solving the camera selection problem in large camera networks. In Proceedings of international workshop on distributed smart cameras.

[24] Wang, C., Qi, F., & Shi, G. (2009). Nodes Placement for Optimizing Coverage of Visual Sensor Networks. *Advances in Multimedia Information Processing - PCM 2009 Lecture Notes in Computer Science,* 1144-1149. doi:10.1007/978-3-642-10467-1_114

[25] Shen, C., Zhang, C., & Fels, S. (2007). A multi-camera surveillance system that estimates quality-of-view measurement. In Proceedings of IEEE international conference on image processing (pp. 193–196).

[26] Mavrianc, A. (2012). Modeling and Optimizing the Coverage of Multi-Camera Systems (Doctoral dissertation, University of Windsor). Electronic Theses and Dissertations. Retrieved May 30, 2020, from https://scholar.uwindsor.ca/etd/5372/

[27] Scott, W. R. (2007). Model-based view planning. *Machine Vision and Applications, 20*(1), 47-69. doi:10.1007/s00138-007-0110-2

[28] Ma, Y. (2011). *An invitation to 3-D vision: From images to geometric models*. New York: Springer.

[29] Chen, S., & Li, Y. (2004). Automatic Sensor Placement for Model-Based Robot Vision. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 34*(1), 393-408. doi:10.1109/tsmcb.2003.817031

[30] Performance and Memory. (n.d.). Retrieved June 02, 2020, from https://www.mathworks.com/help/matlab/performance-and-memory.html?s_tid=CRUX_lftnav

[31] 3ds Max. (n.d.). Retrieved May 31, 2020, from https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2017/ENU/3DSMax/files/GUID-74ECAC41-574C-491F-B98A-E6D7812A78B0-htm.html

[32] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., & Medina-Carnicer, R. (2016). Generation of fiducial marker dictionaries using Mixed Integer Linear Programming. *Pattern Recognition, 51*, 481-491. doi: 10.1016/j.patcog.2015.09.023

[33] Chen, X., Golovinskiy, A., & Funkhouser, T. (2009). A benchmark for 3D mesh segmentation. *ACM SIGGRAPH 2009 Papers on - SIGGRAPH '09*. doi:10.1145/1576246.1531379

[34] Attene, M., Falcidieno, B., & Spagnuolo, M. (2006). Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer, 22*(3), 181-193. doi:10.1007/s00371-006-0375-x

[35] Cohen-Steiner, D., Alliez, P., & Desbrun, M. (2004). Variational shape approximation. *ACM SIGGRAPH 2004 Papers on - SIGGRAPH '04*. doi:10.1145/1186562.1015817

[36] Lévy, B., Petitjean, S., Ray, N., & Maillot, J. (2002). Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG), 21*(3), 362-371. doi:10.1145/566654.566590

[37] Katz, S., Leifman, G., & Tal, A. (2005). Mesh segmentation using feature point and core extraction. *The Visual Computer, 21*(8-10), 649-658. doi:10.1007/s00371-005-0344-9

[38] Application Compiler. (n.d.). Retrieved May 31, 2020, from https://www.mathworks.com/help/compiler/create-and-install-a-standalone-application-from-matlab-code.html

[39] Chen, S.Y., Li, Y.F.: Automatic sensor placement for model-based robot vision. IEEE Trans. Syst. Man Cybern. B Cybern. 34(1), 393–408 (2004a)

[40] Chvátal, V. (1975), "A combinatorial theorem in plane geometry", *Journal of Combinatorial Theory*, Series B, 18: 39–41

[41] D. Kim, I. Dong and S. Lee, "Triangular Mesh Segmentation Based on Surface Normal" *(2002). ACCV2002: The 5th Asian Conference on Computer Vision, 23--25 January 2002, Melbourne, Australia.*

[42] Evangelos Kalogerakis, Aaron Hertzmann, Karan Singh, "Learning 3D Mesh Segmentation and Labeling", *ACM Transactions on Graphics, Vol. 29, No. 3, July 2010 (also in SIGGRAPH 2010, Los Angeles, USA)*

[43] Corke, P. (2017). *Robotics, Vision & Control*. Springer. ISBN 978-3-642-20143-1

Vita Auctoris

Eslam Eissa was born in 1992 in Riyadh. He graduated from Badr High School in 2011. From there he went on to Modern Sciences and Arts University in Cairo, where he obtained a B.Sc. in Electrical Communications and Electronics Engineering in 2016. After working for two years in the automotive industry as a research and development engineer at Valeo, he moved to Canada to pursue graduate studies. He is currently a candidate for the Master's degree in Electrical Engineering at the University of Windsor and is expected to graduate in Summer 2020.