DEEPA NAIK

3D MESH SIMPLIFICATION TECHNIQUES FOR ENHANCED IMAGE BASED RENDERING

Master of Science thesis

Examiner: Prof. Atanas Gotchev
Examiner and topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on 7th of June 2017

ABSTRACT

DEEPA NAIK: 3D Mesh Simplification Techniques for Enhanced Image Based Rendering
Tampere University of Technology
Master of Science thesis, 69 pages
June 2017
Master's Degree Programme in Information Technology
Major: Signal Processing
Examiner: Prof. Atanas Gotchev
Keywords: Mesh Simplification, Depth Decimation

Three dimensional videos and virtual reality applications are gaining wide range of popularity in recent years. Virtual reality creates the feeling of 'being there' and provides more realistic experience than conventional 2D media. In order to feel the immersive experience, it is important to satisfy two important criteria namely, visual quality of the video and timely rendering. However, it is quite impractical to satisfy these goals, especially on low capability devices such as mobile phones. Careful analysis of the depth map and further processing may help in achieving these goals considerably.

Advanced developments in the graphics hardware tremendously reduced the time required to render the images to be displayed. However, along with this development, the demand for more realism tend to increase the complexity of the model of the virtual environment. Complex models require millions of primitives which subsequently means millions of polygons to represent it. Wise selection of rendering technique offer one of the ways to reduce the rendering speed. Mesh-based rendering is one of the techniques which enhances the speed of rendering as compared to its counterpart pixel based rendering. However, due to the demand for richer experience, the number of polygons required, always seem to exceed the number of polygons the graphics hardware can efficiently render. In practice, it is not feasible to store large number of polygons because of storage limitations in mobile phone hardware. Furthermore, number of polygons increase the rendering speed, which would necessitate more powerful devices.

Mesh simplification techniques offer solution to deal with complex models. These methods simplify unimportant and redundant part of the model which helps in reducing the rendering cost without negatively effecting the visual quality of the scene. Mesh simplification has been extensively studied, however, it is not applied to all the areas. For example, depth is one of the areas where general available simplification methods are not very well suitable as most of the methods do not consider depth discontinuities very well. Moreover, some of the state of the art methods are not capable of handling high resolution depth maps.

In this thesis, an attempt is made to address the problem of combining the depth maps with mesh simplification. Aim of the thesis is to reduce the computational cost of rendering by taking the homogeneous and planar areas of the depth map into account, while still maintaining suitable visual quality of the rendered image. Different depth decimation techniques are implemented and compared with the available state of the art methods. We demonstrate that the depth decimation technique which fits the plane to depth area and considers the depth discontinuities, outperforms the state of the art methods clearly.

PREFACE

The research work presented in this thesis was carried in Department of Signal Processing, Tampere University of Technology under the supervision of Prof. Atanas Gotchev. Aim of the research work was to explore the best depth decimation approach for enhanced image based rendering.

First of all, my sincere gratitude to my supervisor Atanas Gotchev for giving me an opportunity to work in his team. I thank him for his continuous support, guidance and feedback while carrying out this work. I would like to thank my co supervisor, Sergey Smirnov for his constant technical supervision, constructive comments and valuable feedback all through during this thesis work.

Special thanks to Olli Suominen who always helped during my tough times. Thanks to all my friends especially Lei Xu and Ramin for all the help and information provided to me during my thesis writing.

I must acknowledge my family, my dear husband Arun Ravindran, for his all time support, my two little princesses Ishnavi and Iisha for their cooperation during my studies.

Finally, I am grateful, to my parents, Ganapayya, Chandrbhagi and my brothers, Dhananjay, Deepak for their love and constant motivation.

Tampere, 7.06.2017

Deepa Naik

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

3D	3 Dimension
2D	2 Dimension
VR	Virtual Reality
HMD	Head Mounted Display
DIBR	Depth Image Based Rendering
CGR	Computer Graphics Rendering
IBR	Image Based Rendering
MVV	Multi View Video
ToF	Time of Flight
PSNR	Peak Signal to Noise Ratio
MSE	Mean Square Error
FEA	Finite Element Analysis
FEM	Finite Element Methods
PM	Progressive Mesh
LDI	Layered Depth Image

1. INTRODUCTION

"Everything you can imagine is real".

Pablo Picasso

This chapter starts with an introduction of the 3D technology, its wide range of applications and its recent developments. The following sections discuss problems related with 3D scene representation, 3D data acquisition and rendering. The final part of the chapter formulates the problem and motivates the thesis.

1.1 3D Visual Technology

In the mid 19^{th} century, Charles Wheatstone found out that human eye can be tricked with the impression of the depth by viewing two similar images put together side by side. The pair of images taken by two cameras displaced by slight horizontal shift mimic what each of our eyes would see in reality.

3D stereography was invented in 1839 [9]. The first 3D movies were available in 1900s. Three dimensional (3D) movies are as old as their fellow 2D movies. Fiction movies of older days have depicted three dimensional moving images.

Sense of the depth in 3D movies or images is perceived because of binocular nature of human visual system. 3D videos and images are produced by mimicking the nature of human eye. The images presented to the two eyes are shifted to address the retinal disparity being formed when seeing an object from different perspectives. In order to present the proper image to the proper eye, filters such as polarization or shutter or anaglif glasses are used.

Along with 3D movies virtual reality has gained wide range of popularity during these days. Virtual reality can be traced back to the early painting of panoramic

Figure 1.1 *Panoramic image depicting immersive experience*

images as shown in Figure 1.1. Painting in the figure depicts the battle field, which was mainly intended to fill the viewers with the larger field of view, making them feel as if they were in the battle field. In 1930s, a story by science fiction writer Stanley G. Weinbaum (Pygmalion's Spectacles) contained the idea of a pair of goggles that let the wearer experience a fictional world through holography, smell, taste and touch [10]. In a nutshell, what was explained by the scientist is nothing but the modern and emerging experience of virtual reality.

Head Mounted Display (HMD) contains a pair of glasses, which are used with virtual reality system bringing a total immersive experience to the user. The first modern appealing HMD was developed by Morton Heilig's in 1960s for the non-interactive film medium without any motion tracking [10]. In 1961, two Philco Corporation engineers (Comeau and Bryan) developed the first precursor to the HMD as we know it today -the Headsight. Since then, there has been continuous progress and improvement in the production and technology of HMD. Major advancement in the virtual reality happened in the first fifteen years of the 21st century. VR brings the sense of real life by simulating or recreating the real life environment or situation. Sensory experiences created by virtual reality includes hearing, touch, sensory and smell. The ultimate goal of VR allows the user to be physically engaged in the simulated environment which opens up the door for the most obvious usage of VR, which is gaming. One of the biggest beneficiary areas of VR is in the medical field where surgery can be simulated. Flight simulation for the Air Force is another interesting application where pilots are trained in simulated environment.

The recent success of 3D visualization techniques determines the research interest and development attempts to improve all components of the 3D technology, starting from capture, representation, processing, storage, transmission and rendering. The following section briefly represents the stages of 3D technology.

1.2 Projective Geometry

Projective geometry provides mathematical structure for 3D multi-view imaging. It helps in modelling 2D image formation from 3D view, generation of synthetic images and also reconstruction of 3D objects from number of images.

Eucledian geometry can be used to model points in 3D space. However, points in infinity cannot be modelled using this geometry. This special case is handled very well using Projective geometry. Projective geometry serves as mathematical model in mapping 3D points on to the 2D image plane.

The pinhole camera model describes how points in physical coordinate system is projected in the image plane of the camera. It describes the geometric relationship between real-world 3D points and 2D points on camera sensors.The pinhole model represents a camera as a system with an optical center and an image plane.

1.3 3D Scene Representation Formats

Three dimensional scene is rich and complex in nature. Human visual system (HVS) perceives and processes this complex scene by system of visual cues. In order to represent the 3D scene, only perceivable visual information is sufficient and visually unnecessary information can be excluded. In this way the original scene can be reproduced indistinguishable from the real one by processing less data. Figure 1.2 shows 3D scene processing chain [11].

Three dimensional scene processing chain consists of three stages. In the first stage, 3D data can be captured using either active or passive method. Details of scene acquisition is explained in section 1.5. Usually, captured scene is not in the format required for presenting it to the user. There has to be a mapping between the real scene to the viewing experience. In the second stage the scene represented in such a way that it can be stored and visualized discretely. This can be done for instance by 3D scene approximation using 3D primitives as points, line segments, polygons etc and stored in computer as discrete data. Finally, rendering pipeline uses the stored data and determines which pixel has to be shown to the user.

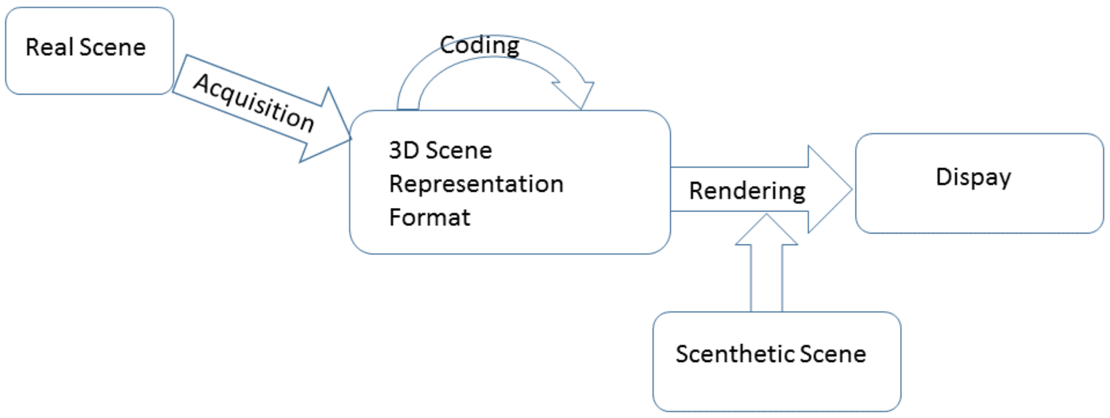


Figure 1.2 *3D scene processing chain*

1.4 Color-plus-depth Representation

3D video experience can be produced by several techniques. Stereoscopy [12] and Depth assisted video formats are two primary techniques for representing the 3D video.

In most of stereoscopy, two images with slightly different view point are presented to the right and left eye. The brain combines these images to give the perception of depth. Major processing of the raw information perceived through the eye takes place in brain. One of the important functions of the brain is to assess the relative distance of the objects from the viewer and also their corresponding depth dimension. The major advantages of the representation are its simplicity the provision of true stereo views ready for display. Disadvantages of this method are complications in synthesis of the virtual views and its limited scalability. Figure 1.3 shows the stereo pair left and right images which can be presented to the eyes to get the perception of depth.

3D scene is efficiently represented using single view plus depth than two channel stereo. In view plus depth representation, geometrical information contained in the disparity between two views is directly encoded. The difference in the location of an object between left and right eye is known as binocular disparity and the depth information can be directly extracted from disparity. Depth is represented as gray scale image where each pixel encodes the distance between camera optical center and the corresponding point in 3D space. Darkest value of the depth image represents the farthest point and brightest value represents the nearest point. Using depth

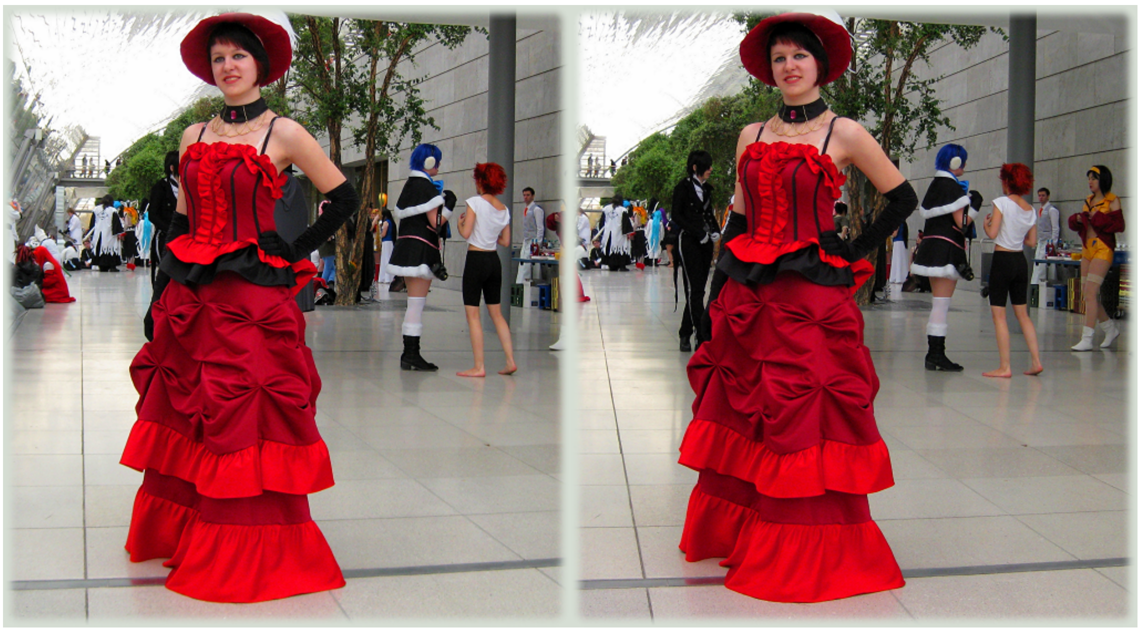


Figure 1.3 *Stereo pair right and left image [1].*

map, virtual views are generated through the technique known, as depth image based rendering (DIBR) [13]. This is one of the flexible methods which allows the generation of more than two views demanded by auto-stereoscopic display. The advantage of this method is the high quality of virtual view synthesis. However, there are several downsides of this method. One of the main drawbacks of this method is its inability to handle occlusions. This is the problem of DIBR where new synthesized image is missing the details behind the foreground object thus creating holes. A special technique known as occlusion filling is needed to fill the holes of the new synthesized image.

1.5 Depth Map And Its Generation

Depth information plays a very important role while generating new view from the existing 2D image. Depth map is a gray scale image where each pixel represents distance between visible point in the 3D world and the camera. It represents the geometry from certain perspective. Structure-wise depth map is smooth image representing the gradual change in the depth within objects. It is also characterized by sharp discontinuities near object boundaries and it does not contain texture [14]. Figure 1.4 shows the texture image and corresponding disparity image on the right.

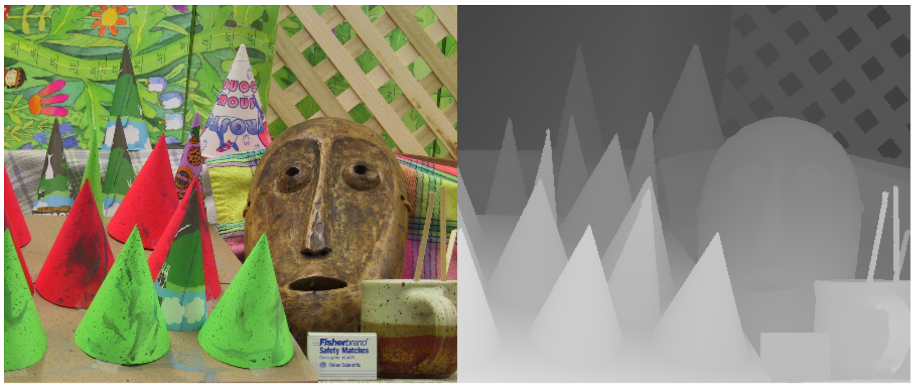


Figure 1.4 *Texture image and disparity image [2].*

The quality of depth map has an important role in the generation of the rendered image. Wrongly estimated depth maps and holes in depth maps will cause resulted depth map image to be distorted. Distorted depth maps will result in generation of wrong objects boundaries or contours in the rendered image causing degraded visual experience.

Precise estimation of the depth is possible during the capture stage using floating point representation. However, floating point representation is not very well suitable for compression and transmission. For transmission and compression purpose floating value is converted to integer value between 0 and 255. Therefore by proper operations like shifting, scaling and transformation, resolution and depth ranges have to be maintained so that they can be properly invertible. Quantization of the depth is performed in logarithmic scale, this helps to preserve the geometry details for the closer objects. In parallax based human stereo vision, degradation in the details of longer distance object is tolerable compare to the shorter distance. This is an important property and can be achieved by transmitting linearly quantized inverse depth map. Figure 1.5 depicts the scenario where due to inverse mapping nearby objects are brighter and farther objects are darker.

Depth map of natural scenes can be estimated by extracting the geometry of the scene. Several methods exist for scene geometry extraction, two of such methods are discussed here. These are categorized as active and passive methods.

Depth can be estimated either by active or passive approaches. Active approaches use special depth sensors based on Time-of-Flight (ToF) principles [15] or laser scanners [16] or structural light [17] to form depth maps.

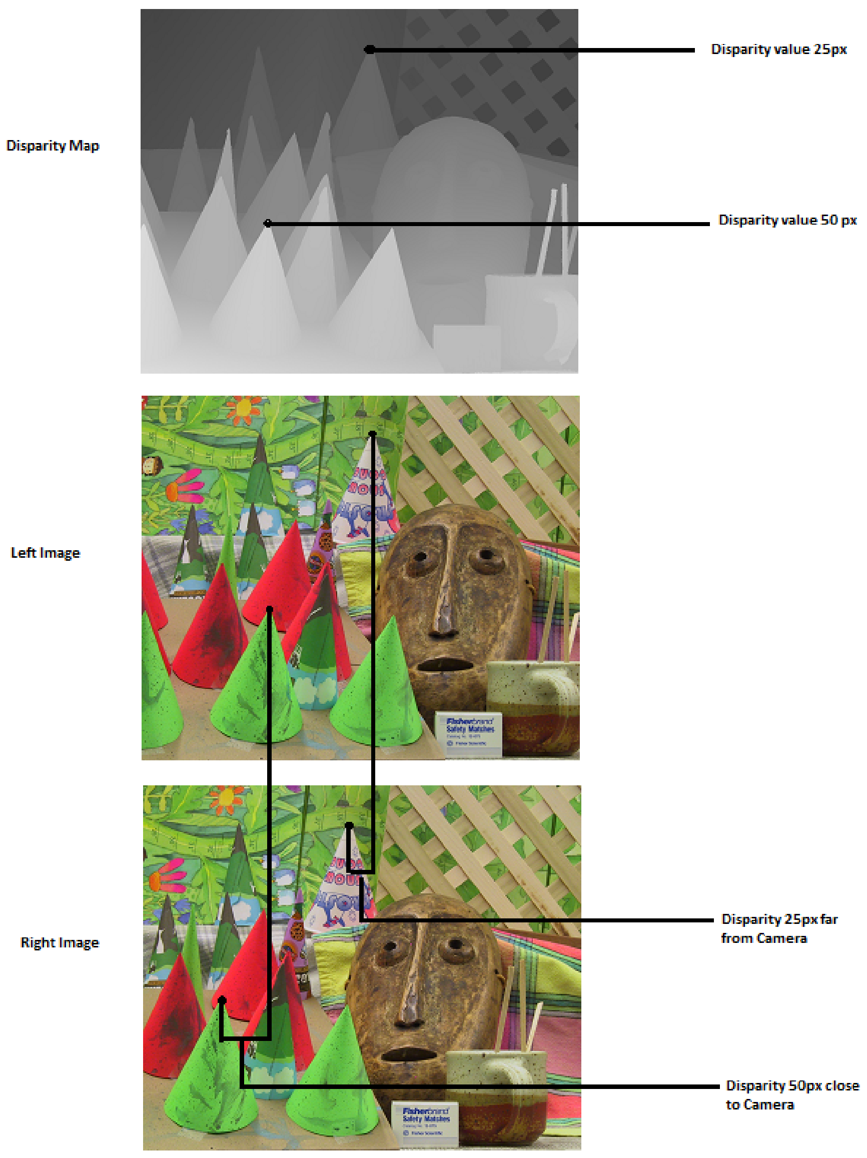


Figure 1.5 *Illustration of depth image where nearby objects are brighter and farther objects are darker.*

In ToF, depth is calculated based on round trip time taken by the emitted light by the source and its detected reflection by the sensor. The principle here is simple, however, the accuracy of the measured depth is subjected to several factors. Accuracy of measured distance depends on the precision of the time-delay measurement. A highly precise clock is required to measure the distance accurately. Because of these reasons this method is not suitable for high accuracy applications. ToF based on phase shift is another category used for acquiring the distance. Here the depth information is computed by detecting the phase difference of an amplitude modulated beam between the transmitted and received signal. Unlike the previous method this camera is able to acquire distance of many points at a time [15].

Structured light technique is another approach, which uses sensor for measuring depth.This process involves projecting known pattern of light onto the scene. Depending on the way the pattern deforms the surface, the system calculates the depth and surface information of the object.

Passive methods do not require special sensors for depth map estimation. Stereo matching which belongs to passive method, uses two or more images of the scene taken from different view points in order to estimate parallax between them. This is also known as binocular disparity. Figure 1.6 shows typical stereo matching pipeline [18] .

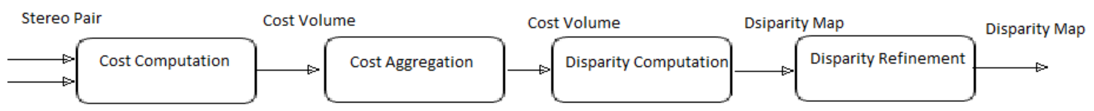


Figure 1.6 *Stereo matching pipeline*

Depth estimation from stereo images has several steps. First stage of this process is image rectification. If the two views from the stereo pair are parallel, search for stereo correspondence can be done in one direction. However practically it is not possible to align the cameras perfectly parallel, so rectification has to be performed before the images are given for depth estimation. Rectification process takes care of imperfect alignment of the camera in the captured image [19].

Image rectification is followed by disparity estimation. Disparity estimation can be sub divided as Cost volume and Cost aggregation processes. Cost volume involves finding dissimilarity between corresponding pixel in two views. Pixels with minimum difference are chosen as best match and their corresponding disparity is selected. Cost aggregation is per-slice filtering of the cost volume to make it more consistent within the disparity dimension. Different aggregations approaches followed are block, Gaussian etc [19]. Disparity computation is the process of selecting optimal disparity value for each pixel. Disparity can be computed from aggregate of the cost volume using winner takes all approach where disparity with the lowest aggregated cost is chosen. After the disparity map has been obtained, a final stage of post-processing (enhancement) can be applied to refine it.

During stereo matching process, several errors might occur in various stages. Disparity refinement process mitigates the error caused by stereo matching process. Most techniques rely on per-pixel matching for finding correspondences. Area with good textures work well for per pixel matching, however, texture less zones suffer with this approach. Model fitting is one of the approaches to reduce the error in such case. First, texture-less zones are extracted using segmentation and then plane is fitted to get the depth values.

Occlusion is another systematic error produced in stereo matching pipeline. Left and Right views observe the scene from two different perspectives therefore some pixels in one view might be hidden in the other. Occlusion is handled in the post-processing stage by a kind of guessing. The simplest approach is to apply median or weighted-median prediction. While it is able to recover significant part of the occluded zones it fails in more complicated areas. Plane fitting within a color segment is another standard remedy.

Some other set of errors could be due to sensors. This is mitigated by median or some other type of filter. Geometrical or scenery difficulties are another set of errors, standard approach to mitigate such errors is to include more observations or improve the lightening conditions.

1.6 Problem Definition

For realtime rendering mesh-based representations are more suitable as they make use of underlying graphics hardware efficiently. We mainly need meshes for rendering. Image-based rendering can be broadly classified as rendering without geometry, rendering with implicit geometry and rendering with explicit geometry. Mainly, rendering with explicit geometry is considered in this thesis. Rendering with explicit geometry use geometric description such as polygons for rendering. Depth Image-Based Rendering comes under this category. DIBR uses depth map and associated texture to synthesize the new view. DIBR includes several other algorithms along with image warping and mesh-based rendering. Image warping renders each pixel on to the screen, this is not an ideal choice to use with meshes, so the natural choice is to use mesh-based rendering.

Mobile platforms are substantially less capable in terms of graphics card and processors compared to powerful desktop computers. In this thesis our focus is mainly on mobile platform. Mesh-based representations help to increase the performance to a greater extent. However, in order to produce smoother realistic 3D visualization, the amount of meshes we need is much more higher than the fastest mobile device can render them in smoother way. For example, for an image of size 1110x1390, there are total of 1,542,900 pixels to be rendered in case of DIBR and total of 3,080,802 triangles to be rendered in mesh-based rasterization. So, rendering time is directly proportional to the amount of polygons. So, we need to simplify the meshes. The first challenge is to find ways to reduce the complexity of the model so that the mobile device can efficiently render them while still producing suitable visual experience as compared to the full model.

Mesh simplification has been studied extensively, however, its application to the depth map is not very well explored. The second challenge is not only to apply the simplification process to the depth map but also consider important decimation criteria required for the depth map such as, visual appearance, depth discontinuity, object boundaries while simplifying the complex model.

1.7 Motivation

Motivation of the thesis comes from two aspects. The first aspect is related with the structure of the depth map. Structure-wise depth map is smooth image representing the gradual change in the depth within objects [14]. There are many constant regions which can be potentially simplified by removing the redundant vertices and faces. This will help in bringing down the overall complex model to the simpler model which can be efficiently rendered.

The second aspect is related with the current configuration of the mobile devices. Rendering high resolution (up to 8k) images with high frame rate requires good graphics cards, powerful processors, and related programming interfaces. These are easily available on desktop platforms, but unavailable on mobile devices such as cellular phones. Battery capacities of mobile devices are another issue. Rendering high resolution images require substantial battery power. So, simplification of meshes help to render the complex images smoothly on low configuration devices such as mobile.

1.8 Thesis Outline

Chapter 1 gives a brief introduction about history of 3D technology, its growing popularity, and the need for understanding the basics of 3D representation and rendering. Chapter also introduces problem and motivates the thesis.

Chapter 2 discusses basics of meshes, different ways meshes can be represented and different mesh simplification techniques.

Chapter 3 further elaborates on the projective geometry, pinhole camera basics, different matrices which are required for rendering the image when view and depth are given.

Chapter 4 presents different rendering techniques, their advantages and disadvantages.

Chapter 5 presents the approaches used in this thesis for mesh decimation, different data sets used for the experiments and finally the experiments that are carried out.

Chapter 6 summarizes and analyses the experimental results part of the thesis.

Chapter 7 contains the conclusion arising from the thesis work.

2. 3D MESH REPRESENTATION

"All things are bound together, all things connect".

Chief Seattle

This chapter presents an introduction of geometrical models based on meshes followed by a discussion about mesh processing, which is also the main focus of this thesis work.

2.1 Introduction

Polygon models are used in many applications including interactive computer graphics, product design, manufacturing, gaming, simulation, cultural heritage, archaeology, medicine, bio informatics, etc [20]. Their historical success traces back to the Finite Element Analysis (FEA) problem. Originally FEA has been developed for solving problems in solid mechanic. FEA is applied for structure analysis such as cantilever, bridges and in solid mechanics such as gear and automotive platforms and in electrical analysis such as actuators, electrical signal propagation etc [21]. Practical application of Finite Element Method (FEM) comes from FEA, which is applied as computational tool to reveal the properties and state of the system. FEM solves differential equations using piece-wise polynomials. It is a numerical method and subdivides a larger complex problem into simpler and smaller pieces called elements. FEM uses techniques such as mesh generation for dividing complex problem into smaller elements. The shapes of these elements are in the form of triangles, rectangles, tetrahedral, etc. Figure 2.1 shows different types of mesh elements.

Due to the evolution of computer animation during the first decades of 2000, mesh generation gained incredible popularity compared to FEMs from which it has evolved. Nowadays areas such as computer graphics, entertainment and gaming rely on various types of meshes. Objects are modelled as meshes before rendering them in

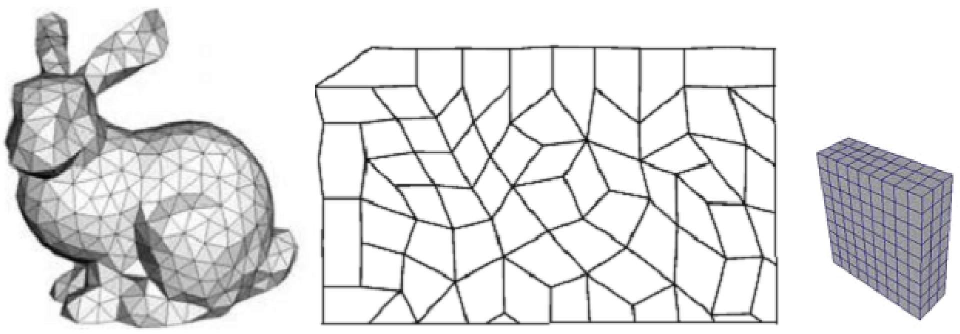


Figure 2.1 *Different types of mesh elements.*

these computer graphics applications. Meshes are also widely used in geography and cartography where terrain data can be represented compactly [22]. Meshes are also used in various other applications such as aerial land surveying and image processing [23]. Due to the existence of wide range of applications, mesh generation and its processing indeed gained strong research interest.

2.2 Mesh Generation

A mesh is a discretization of a geometric domain into small simple shapes such as triangles or quadrilaterals in two dimensions and tetrahedral or hexahedral in three dimensions [22].

Based on dimensionality and choice of elements, meshes can be categorized as, triangular, tetrahedral, quadrilateral and hexahedral meshes. Elements of the meshes include vertices, edges, faces, polygons and surfaces as illustrated in Figure 2.2. Vertex is a point where two or more lines meet, edges are connections between two vertices, face is closed set of edges where three edges make a triangle and four edges make quad face.

Meshes are also broadly classified as structured and unstructured meshes [23]. In Structured meshes all the nodes have same topology. Since connectivity of neighbor node is defined implicitly, structured node provides memory efficiency. However, for complex geometry it may not be easy to generate the structured meshes. Structured meshes have the characteristics of regular connectivity. Generation of structured meshes can be roughly classified into elementary approaches (hand-generated) and complex approaches (algebraic) [22].

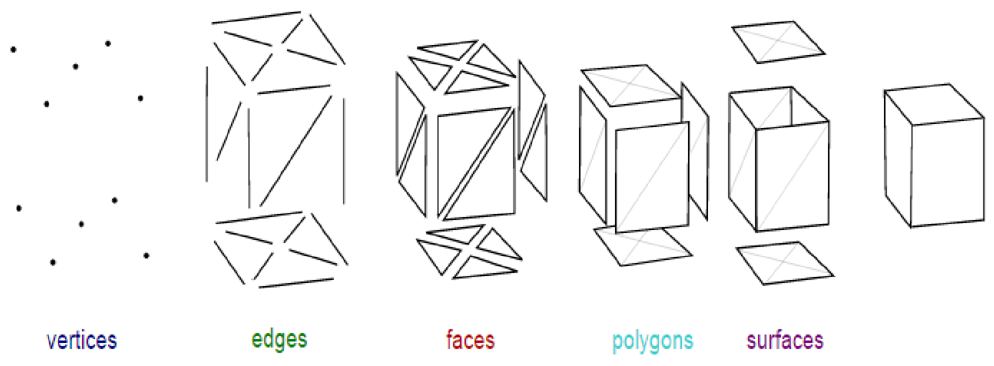
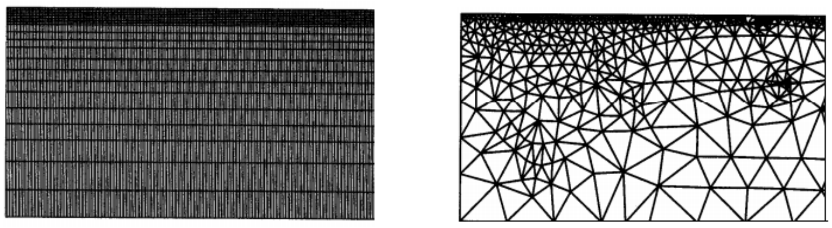


***Figure* 2.2** *Vertices,Edges and Faces of a cube.*



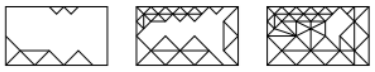
(a) Structured Mesh [23] (b) Unstructured Mesh [23]

***Figure* 2.3** *Structured and Unstructured mesh*

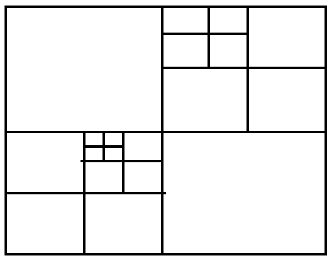
Unstructured meshes are defined as the meshes whose vertices have arbitrarily varying neighborhood [23]. On the contrary to structured mesh, unstructured mesh can be easily used to fit the complex domain. Furthermore, unstructured mesh needs fewer elements as compared to the structured mesh for the same domain as they grade in size rapidly. Unstructured meshes can be generated using several approaches, namely, advancing front, quad tree decomposition and Delaunay triangulation. Structured and Unstructured meshes are shown in Figure 2.3

2.2.1 Advancing Front

In this method, elements are constructed one by one from the boundary of the model advancing inward [23]. This method first discretizes the boundary to form the edges. These edges act as the initial front and a new triangle is formed using these edge as base by joining two ends of the current base. Edges of the newly formed triangle act as the current front to form the next triangle. This process continues until the entire domain is covered. Efficiency of the Advancing front method works well at the



(a) Advancing Front [23]



(b) Quadtree

Figure 2.4 *Advance Front and Quadtree*

boundaries compared to interiors. This technique is exemplified in Figure 2.4(a).

2.2.2 Quadtree

Quadtree encloses entire domain as one square. Initially, the square is divided into four equal sized blocks. Each block is tested for certain criteria, if the block meets the criteria, the block is not further divided otherwise it is divided into four equal sized blocks. The process repeats until a criterion is met or the blocks cannot be further divided. As a result of this, resulting blocks may have different sizes. Typical Quadtree decomposition is shown in Figure 2.4(b). In [24], a binary space partition method has been proposed. This method adaptively divides the depth map into meshes. The divided mesh has the form of binary triangular tree. However, this method mainly aims at the compression of the depth map. In [25], the work considers the image warping using depth map. This method is based on adaptive triangulation of depth buffer and its simplification. In line with the goal of the this thesis, this method also aims at reducing the rendering cost by taking the advantage of the scene geometry.

2.2.3 Delaunay Triangulation

Delaunay creates triangular mesh using set of points, that tend to avoid skinny triangles. The main application of this triangulation is in image morphing. A triangle is said to be delaunay if the circum circle of any triangle in the triangulation does not contain any other points where circumcircle is the smallest circle enclosing the triangle. However, it is not guaranteed that the triangulation always exists. Delaunay triangulation maximizes the minimum angle of the triangles. Because of this, the created triangles have the best shape. Sample delaunay triangles created from the set of points are shown in Figure 2.5. This method offers many functionalities

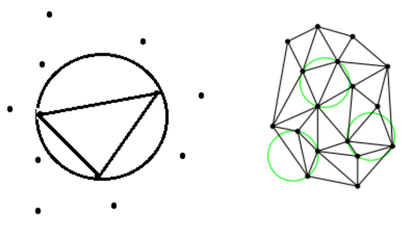


Figure 2.5 *Delaunay Triangulation [3]*

for developing triangulation-based applications such as search for triangles, modify triangulation to insert/remove points, triangulate to remove the triangles outside the domain, etc [26].

2.3 Mesh Representation

Meshes can be represented in several ways. The following section briefs some of the mesh representation methods.

- **Vertex-Vertex:**

  This is the simplest representation. It uses a set of vertices connected to other vertices to represent an object. This is not a very popular method as generation of faces require traversal of entire data set for rendering. However, it has the advantage of small memory foot-print. Figure 2.6 shows representation of four sided box as Vertex-vertex mesh. In this representation, all vertices store the indices of its neighboring vertex thus they implicitly store the faces and edges.

- **Face-Vertex:**

  This approach uses both vertices and faces to represent an object. It is one of the popular methods as faces and vertices are explicitly stored. Each face has three vertices stored and they use the shared vertices. Performance-wise this is faster compared to the Vertex-vertex. However, this method has larger memory foot print compared to the previous approach. Face Vertex table is shown in Figure 2.7.

Vertex List

v0	0,0,0	v1 v5 v4 v3 v9
v1	1,0,0	v2 v6 v5 v0 v9
v2	1,1,0	v3 v7 v6 v1 v9
v3	0,1,0	v2 v6 v7 v4 v9
v4	0,0,1	v5 v0 v3 v7 v8
v5	1,0,1	v6 v1 v0 v4 v8
v6	1,1,1	v7 v2 v1 v5 v8
v7	0,1,1	v4 v3 v2 v6 v8
v8	0.5,0.5,1	v4 v5 v6 v7
v9	0.5,0.5,0	v0 v1 v2 v3

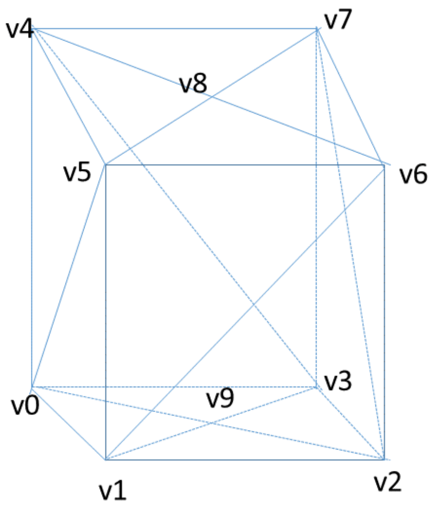
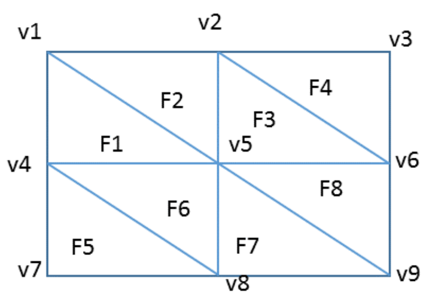


Figure 2.6 *Vertex Vertex mesh*

Vertex Table

v1	x1 y1 z1
v2	x2 y2 z2
v3	x3 y3 z3
v4	x4 y4 z4
v5	x5 y5 z5
v6	x6 y6 z6
v7	x7 y7 z7
v8	x8 y8 z8
v9	x9 y9 z9



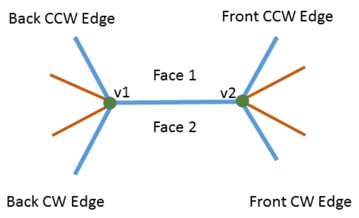
Face Table

F1	v1 v51 v4
F2	v1 v2 v5
F3	v2 v6 v5
F4	v2 v3 v6
F5	v4 v8 v7
F6	v4 v5 v8
F7	v5 v9 v8
F8	v5 v6 v9

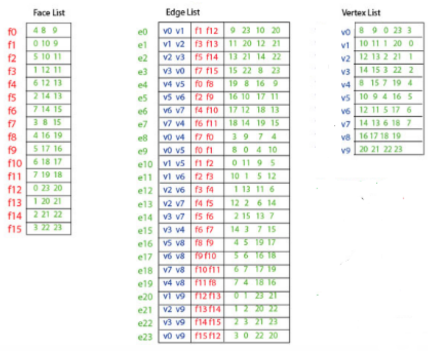
Figure 2.7 *Face vertex mesh*

- **Winged Edge Mesh:**

  Winged edge representation stores faces, vertices and the edges. The basic block of the winged edge mesh is edge. Two faces are separated by edges. An edge has four wings. The name winged edge is derived by imagining the two polygons as butterfly's wings [27]. Winged edge mesh stores the edges in an arbitrary order, and a given edge might either point in clockwise or anti clockwise direction. Graphics hardware needs to generate the face index while rendering winged edge meshes. A drawback of this representation is the need



(a) [23]



Face List

f0	4 8 9
f1	0 10 9
f2	5 10 11
f3	1 12 11
f4	6 12 13
f5	2 14 13
f6	7 14 15
f7	3 8 15
f8	4 16 19
f9	5 17 16
f10	6 18 17
f11	7 19 18
f12	0 23 20
f13	1 20 21
f14	2 21 22
f15	3 22 23

Edge List

e0	v0 v1	f1 f12	9 23 10 20
e1	v1 v2	f3 f13	11 20 12 21
e2	v2 v3	f5 f14	13 21 14 22
e3	v3 v0	f7 f15	15 22 8 23
e4	v4 v5	f0 f8	19 8 16 9
e5	v5 v6	f2 f9	16 10 17 11
e6	v6 v7	f4 f10	17 12 18 13
e7	v7 v4	f6 f11	18 14 19 15
e8	v0 v4	f7 f0	3 9 7 4
e9	v0 v5	f0 f1	8 0 4 10
e10	v1 v5	f1 f2	0 11 9 5
e11	v1 v6	f2 f3	10 1 5 12
e12	v2 v6	f3 f4	1 13 11 6
e13	v2 v7	f4 f5	12 2 6 14
e14	v3 v7	f5 f6	2 15 13 7
e15	v3 v4	f6 f7	14 3 7 15
e16	v5 v8	f8 f9	4 5 19 17
e17	v6 v8	f9 f10	5 6 16 18
e18	v7 v8	f10 f11	6 7 17 19
e19	v4 v8	f11 f8	7 4 18 16
e20	v1 v9	f12 f13	0 1 23 21
e21	v2 v9	f13 f14	1 2 20 22
e22	v3 v9	f14 f15	2 3 21 23
e23	v0 v9	f15 f12	3 0 22 20

Vertex List

v0	8 9 0 23 3
v1	10 11 1 20 0
v2	12 13 2 21 1
v3	14 15 3 22 2
v4	8 15 7 19 4
v5	10 9 4 16 5
v6	12 11 5 17 6
v7	14 13 6 18 7
v8	16 17 18 19
v9	20 21 22 23

(b)

Figure 2.8 *(a) Winged Edge Mesh (b) Winged Edge Mesh Table*

for large storage space. However, this representation is advantageous in the domain which change the mesh geometry dynamically. Winged edge mesh is shown in Figure 2.8.

2.4 Mesh Complexity

Due to the advancement of computer graphics in the recent years, there has always been a demand for visualizing/rendering large complex data. Polygons, such as tetrahedral and triangles are considered as one of the most common used primitives for computer graphics applications. Geometric representation of complex objects might require up to several millions primitives (e.g polygons). Rendering such huge amount of data needs either fast processing graphics hardware or complexity reduction methods. Unfortunately, the complexity of the model measured by number of polygons grows much faster than the current graphics hardware can efficiently render them. Which means the number of polygons we need to create smooth and realistic experience always seem to exceed number of polygons our graphics hardware can afford [4]. However, this is not practical as large number of polygons require big memory and the rendering speed is directly proportional to the number of polygons.

Quality of rendering improves as the amount of triangles or the primitives required to represent the data, however it comes with significant cost in terms of time. Higher the number of polygons, better the rendering quality. However, complex geometry requires far less than the existing mesh to represent the similar quality of the visualization. Due to this there is significant need to study the geometry of the model and find the way to reduce the complexity of the model for various purposes. Reducing the complex geometry of the object to simpler and more efficient representation while still maintaining the visual fidelity is called mesh simplification. Figure 2.9 illustrates the mesh simplification.

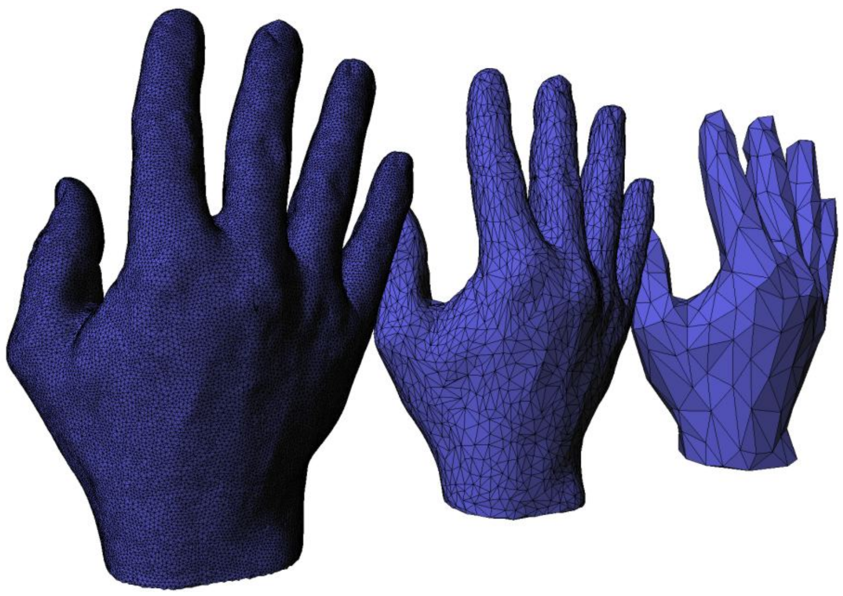


Figure 2.9 *Illustration of Mesh Simplification*

2.5 Mesh Processing

Mesh processing includes simplification and decimation. Simplification techniques use a set of algorithms to transform a given polygonal mesh into another mesh with fewer faces, vertices and edges thus reducing the complexity of a given mesh [5].

There are several different applications that require mesh simplification for various reasons. Some applications may have complex model with redundant data. Aim of such application is to remove the redundant data to either reduce the model size or to improve the run time performance.

Some other reasons for simplification could be not only to reduce the model size but also to maintain the geometric accuracy, visual fidelity and pre-process time [4].

The type of the model is another reason for simplification. For example, complex mechanical models, large assembly of small objects and models with lot of texture require simplification for further analysis and processing.

The mesh topology is an important part of mesh simplification. Mesh simplification algorithms can be classified into two categories, namely, topology preserving and topology modifying algorithms. Topology preserving algorithms maintain manifold connectivity while the modifying ones do not. If the neighborhood of every feature consists of a connected ring of polygons forming a single surface, then it is called manifold connectivity [4].

Topology preserving applications are the best suit for various applications. Figure 2.10 shows the non manifold mesh.

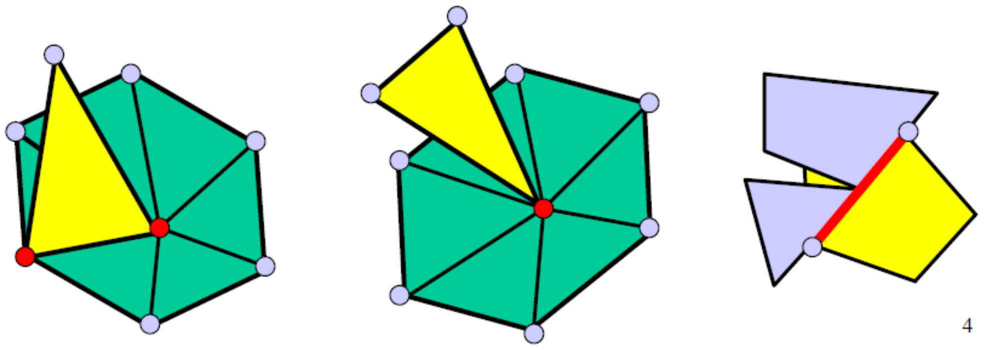


Figure 2.10 *Non manifold mesh [4]*

Broadly, mesh simplification algorithms can be classified as Incremental decimation and Vertex clustering. Incremental decimation includes several other algorithms such as Edge collapse, Vertex collapse, Half edge collapse, etc. Vertex clustering includes Progressive mesh and Image driven simplification.

2.5.1 Incremental Decimation

Incremental decimation simplifies the mesh iteratively by simple topological operations like vertex removal or edge collapse instead of changing the organization of the mesh. In order to simplify the meshes, this algorithm can take user constraints into account for decimation. However, due to high order asymptotic complexity they are demanding for the large input mesh. There are several operators used for decimation which can either preserve or modify the topology. Examples of the commonly used operators are Vertex Collapse, Edge collapse, Half edge collapse.

Vertex Collapse: The basic principle of vertex collapse/removal is shown in Figure

2.11. Vertex collapse algorithm for triangular mesh is explained in this section which

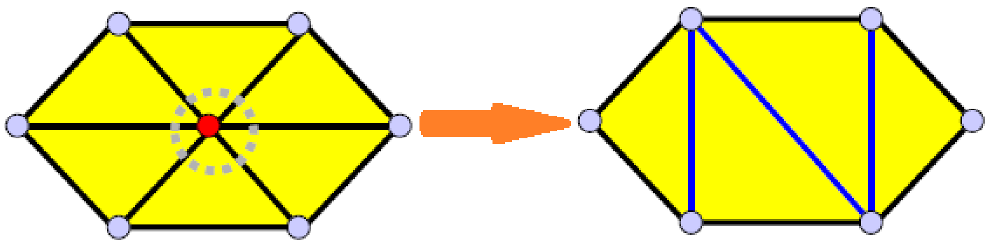


Figure 2.11 *Principle of vertex removal [5]*

was first presented by Schroeder et.al [28]. This is one of the fast and topology preserving algorithms which aims at reducing the total number of triangles in a triangular mesh [28]. The main principle of this algorithm is to iteratively remove some vertices and all triangles associated with the removed vertices, meeting certain criteria. The algorithm consists of three steps. The first step identifies the potential vertex candidates for deletion. There are possible five classifications to which a vertex can belong to. They are simple, complex, boundary, interior edge or corner vertex[28]. The geometry of this classification is shown in Figure 2.12[28].

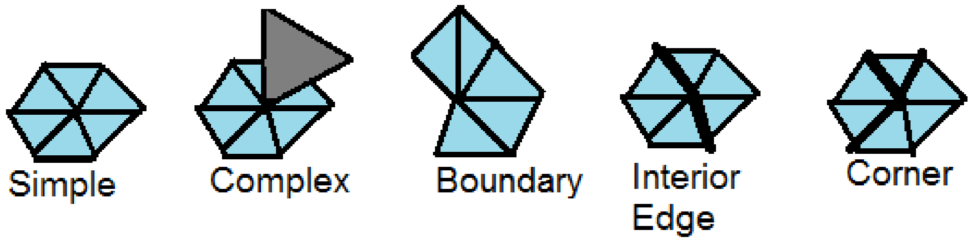


Figure 2.12 *Possible Vertex Classification*

Vertices belonging to complex category are not candidates for removal. The second step evaluates whether the triangles forming the loop can be deleted and replaced by another triangulation, which is exclusive of the original vertex. The third step is similar to post processing where the holes created due to the triangle removal in previous two steps are triangulated again. This process continues until there are no more vertices for removal. Vertices and Faces produced by this algorithm are subset of the original data set.

Edge Collapse: This method selects an edge for removal and merges adjacent vertices to a single new vertex as shown in Figure 2.13. Two triangles that share

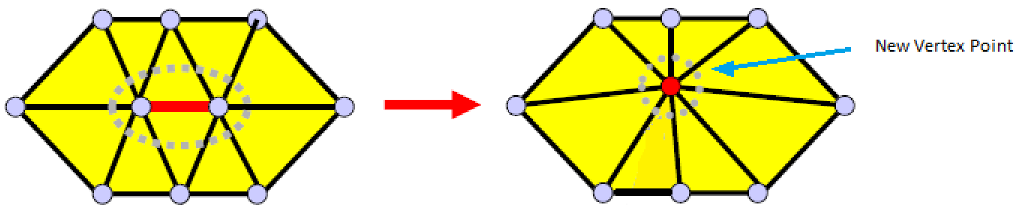


Figure 2.13 *Edge Collapse and Vertex Merge*

the removed edge are also collapsed thus reducing vertex count by one and triangle count by two.

Several metrics can be used for pair contraction such as quadric error metric, pseudo global metric etc. Pair contraction using quadric error metric (surface simplification) was first proposed by Garland et.al [29]. This algorithm however does not preserve the topology but maintains the overall shape of an object. Maintaining the overall shape is an important criterion in rendering applications.

2.5.2 Vertex Clustering

Unlike the triangle mesh decimation algorithm mentioned in the previous section, vertex clustering is a topology modifying algorithm. The vertex clustering algorithm has three major steps. The first step is to subdivide the given mesh into a regular grid. The second step is to find the important representative vertex. Grid with more than one vertex is mapped to this representative vertex as shown in Figure 2.14. There are number of ways to find the representative vertex. One of the easiest ways is to find the mean of the vertices belonging to the same cell and select that mean as the representative vertex. If $P_1, P_2, P_3 \ldots P_k$ are the vertices in a cell, then the representative vertex can be computed as $P = \frac{(P_1+P_2+\ldots+P_k)}{K}$. Median or other robust estimates can also be used.

The third step is the mesh generation step. If the set of vertices $p_0, p_1, p_2, \ldots p_m$ have the representative vertex P and the vertices $q_0, q_1, q_2, \ldots q_n$ have representative vertex Q, then P and Q are connected in the decimated mesh if one pair of the vertices (p_i, q_j) was connected in the original mesh.

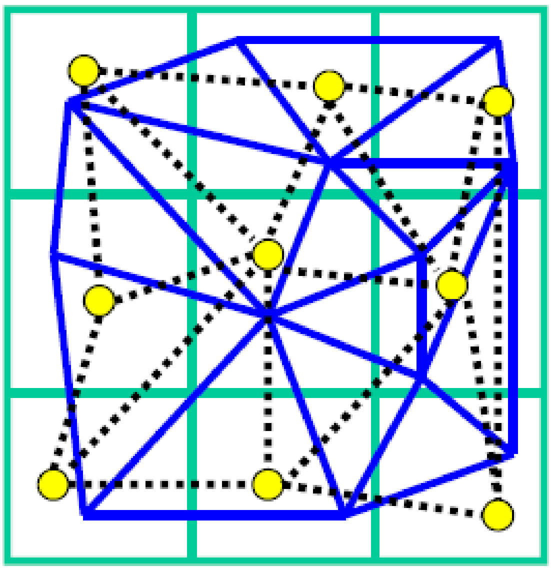


Figure 2.14 *Non manifold mesh*

The quality of the simplification depends on the resolution of the grid. Finer the quality of the grid, better the quality. For over-sampled meshes, where geometry is relatively simple, the resulting simplified mesh is an approximation with excellent complexity demands. Figure 2.15 shows the quality of the simplified mesh when representative vertex is selected with different technique.

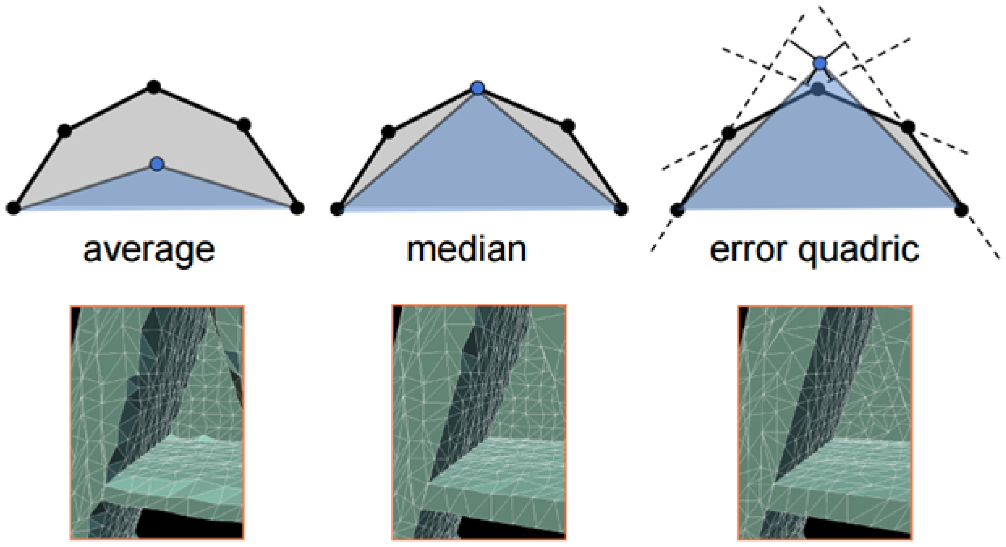
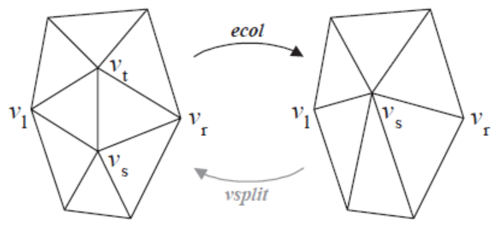


Figure 2.15 *Figure depicting quality of simplified mesh*



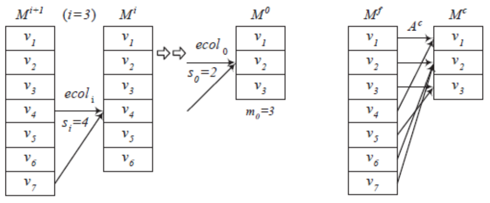


Figure 2.16 (a) Edge collapse and inverse splitting transformation (b) Consecutive edge collapse in the vertex buffer.

2.5.3 Progressive Mesh

Unlike the conventional representation, progressive mesh representation is optimized for storing and transmitting triangular meshes. In Progressive Mesh (PM) form, an arbitrary mesh $\bar{M}$ is stored as a coarser mesh M^0 together with a sequence of n detail records that indicate how to incrementally refine M^0 exactly back into the original mesh $\bar{M} = M^n$. PM uses the similar edge collapse method with a specific uses different energy function for collapsing the edge [30]. A series of edge collapse *ecol* operation transforms the base mesh $\bar{M}$ to a much simpler mesh M^0.

$(\bar{M} = M^n) \xrightarrow{ecol_{n-1}} \ldots \xrightarrow{ecol_1} M^1 \xrightarrow{ecol_0} M^0$ [30]. A key aspect of this representation method lies in the fact that the edge collapse method is invertible, which means it can construct the original mesh back by a an inverse transformation called vertex split to a base mesh $\bar{M}$. A vertex split transformation *vsplit* adds a new vertex and two faces as shown in Figure 2.16 to reconstruct the original mesh from the coarse mesh.

This simplification method preserves geometry along with the overall appearance. Like any other novel approach, the quality of the simplified mesh depends on the edge selected for collapse. There are handful of mesh simplification techniques based on PM. PM selects the edge collapse method based on the need of the application and importance is given to the trade off between speed and accuracy. The edge for collapse can be selected by crude way of random selection or by a much sophisticated quadric error matrix.

State of the art method [30] based on edge collapse uses composite energy function E(M) and it is defined as composition between several components. $E(M) = E_{\text{dist}}(M) + E_{\text{spring}}(M) + E_{\text{scalar}}(M) + E_{\text{disc}}(M)$. The energy function $E(M)$ measures the accuracy of the modified mesh M with respect to the base mesh $\bar{M}$.

The distance energy E_{dist} is equal to the sum of the squared distances from the points $X = \{x_1, \cdot..., x_n\}$ to the mesh

$$E_{\text{dist}}(K, V) = \sum_{i=1}^{n} d^2(x_i.\phi_v(K)). \tag{2.1}$$

E_{spring} is added to guarantee existence of the minimum

$$E_{\text{spring}}(K, V) = \sum_{\{j,k\} \in K} K \parallel v_j - v_k \parallel^2. \tag{2.2}$$

$E_{\text{scalar}}(M)$ measures the accuracy of the scalar attributes, and $E_{\text{disc}}(M)$ measures the geometric accuracy of its discontinuity curve.

The energy function $E(M)$ is minimized using outer and inner nested loops. In the outer loop, connectivity of the mesh K is optimized by applying edge transformation in three different ways, named as edge split, edge collapse and edge swap. In the inner loop, for each candidate, the algorithm computes $E_{k^1} = min_v\,(E_{dist}(V) + E_{spring}(V))$ by optimizing over vertex position v [30]. The energy E_k at the connectivity K of vertex v and the energy at E_{k^1} at new vertex v^1 are computed. If $\Delta E = E_k - E^1$ is negative, a transformation is applied. All the edges that are legal for collapse are placed in a priority queue, where priority depends on the calculated ΔE. In each iteration, the one which has lowest ΔE in the front of the priority queue is transformed and priorities of edges in the neighborhood of the current transformation are recomputed.

This process continues until no further simplification is possible. Along with maintaining the overall geometry of the mesh, this method also preserves the vertex attribute such as color and texture.

2.5.4 Image Driven Simplification

Unlike other simplification models which use geometry to simplify the model, in image driven simplification methods, an image is used to decide the part of the model to be simplified. An edge collapse operator is used for simplification. The cost of the edge is determined by comparing the original model with the simplified model. Image difference between the original model and simplified model shows that the simplified model is very close to the original model.

Most of the mesh simplification methods use Euclidean distance as an optimization metric to collapse the edge. In image driven simplification, the metric used for simplification is based on visual similarity measured by image differences. This is quite reasonable as not all methods need geometrical correctness with respect to the original model. However, in this method visual fidelity takes over geometrical correctness.

All edge collapse algorithms have two principles in common. One is to prioritize the edge to collapse and second is to find the new place of vertex after edge collapse. For the vertex replacement, a heuristic approach based on geometry defined in [31] is used and for the edge collapse the cost function given by the image matrix is used.

Suppose I^0 is the collection of the images of the original model, I^k is the model after collapse of k edges and I^{k+1} after the collapse of an additional edge e. Then an expression for the cost $f(e,k)$ of collapsing edge e in iteration k is given by [32]

$$f(e,k) = \sum_{h=1}^{l} \sum_{i=1}^{m} \sum_{j=1}^{n} \left[\left(I_{hij}^{0} - I_{hij}^{k+1} \right)^2 - \left(I_{hij}^{0} - I_{hij}^{k} \right)^2 \right]. \tag{2.3}$$

In most of the geometry based algorithms the edge cost remains the same for the iteration k except for the small set of edges that had collapsed in the previous iteration. However, this method requires edge cost to be evaluated for each iteration of k for the entire set of all remaining edges.

3. PROJECTIVE GEOMETRY

"There is geometry in humming of the strings there is music in the space of the sphere".

Pablo Picasso

This chapter opens up a brief discussion about need for projective geometry which plays an important role in 3D scene acquisition and synthesis. This is followed by an introduction to the pinhole camera basics which maps the relationship between 3D world and 2D image plane. Different projection matrices required to synthesize new images corresponding to different view points are explained further. The chapter concludes with short explanation of different types of projections.

3.1 Need For Projective Geometry

In order to understand the need of projective geometry, it is necessary to understand the Euclidean geometry. Euclidean geometry mainly describes the angles and shapes of the object [33]. When Euclidean transformation is applied, properties of angle and length are preserved and also parallelism is maintained. Using Euclidean geometry, an exception is needed to explain the basic concepts of the geometry, for example, intersection of lines. Eucledian geometry however maintains the important properties, this is not all that suitable for image processing, because when 3D object is mapped to 2D plane, parallel lines may intersect as shown in Figure 3.1 and also length and angles are not preserved. However, by adding an ideal point at infinity where parallel lines meet, Euclidean space is transformed into a new type of geometric object called projective space. So projective space is an extension of Euclidean space where two lines always meet in a point at infinity. It provides a mathematical formula to describe the geometry of cameras and the associated transformations. Projective geometry enables the design of computational approaches that manipulates 2D projections of 3D objects [6].

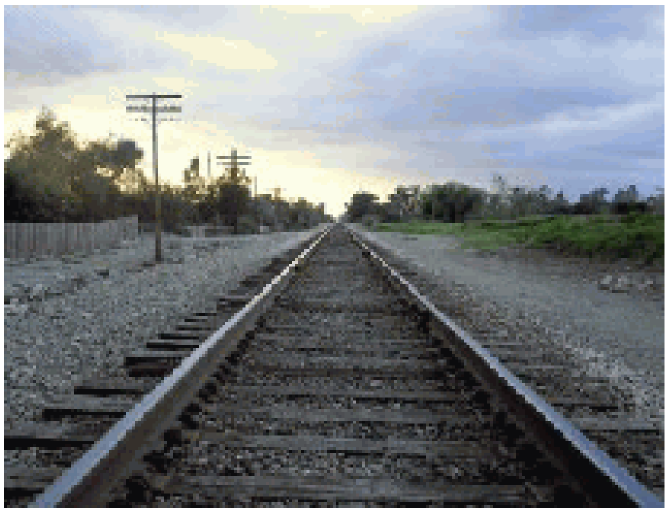


Figure 3.1 *Illustration of parallel lines intersecting at infinity.*

Projective geometry relies mainly on Homogeneous coordinate system. Homogeneous coordinate system helps to capture the concept of infinity. Using infinity, concept of geometry and other computations can be simplified greatly. A point in 2D (x, y) in Cartesian coordinate system can be converted to Homogeneous coordinate by adding another variable W. So a point (x, y) in Cartesian coordinate becomes (x, y, W) in Homogeneous coordinate system. To convert back a point from Homogeneous to Cartesian, points x and y have to be divided by W.

$$(x, y, W)_{Homogeneous} \Longleftrightarrow \left(\frac{x}{W}, \frac{y}{W}\right)_{Cartesian}. \tag{3.1}$$

Similarly, Cartesian coordinate can be expressed in Homogeneous coordinate system as (x, y, z, W)

$$(x, y, z, W)_{Homogeneous} \Longleftrightarrow \left(\frac{x}{W}, \frac{y}{W}, \frac{z}{W}\right)_{Cartesian}. \tag{3.2}$$

3.1.1 Pinhole Camera Model

Pinhole camera model is derived from the physical construction of the early cameras. It describes how points in 3D space are projected on the image plane of the camera. [34]. It is a simple optical imaging device in the shape of a closed box. It contains a small hole called an aperture through which light propagates and creates an image of the outside space on the opposite side of the box. When a 3D world object is projected on the 2D plane, the image is inverted and the size of the image is reduced compared to the original image size.

Principle of pinhole camera is very simple. In pinhole camera of all the light that is reflected in different direction, only a small bundle of light traveling in the same direction enters the aperture of the pinhole model. This process creates flipped image of an object. Size of aperture plays an important role; Smaller the size of an aperture, sharper the image is. Figure 3.2 shows pinhole basics .

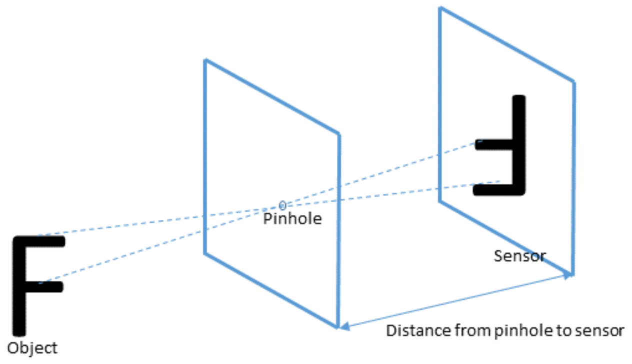


Figure *3.2 Illustration of basics of pinhole camera model [6]*

3.2 Image Formation In Pinhole Camera

The geometric model of the camera projection is shown in Figure 3.3. Light rays pass through optical center O and intersect at image plane $I(u, v)$. Optical axis is the line perpendicular to the image plane and passes through the optical center. Principal point is intersection of the optical axis and the image plane I. The distance from the principal point to the optical center is called focal length. A point $P(x, y, z)$ in the 3D world is mapped to point in 2D image plane $I(u, v)$, by the intersection of a line going through both pinhole and the image plane. Principal point and optical axis define the capturing direction of the camera.

The relation between point $P(x, y, z)$ in 3D space and the point projected on 2D plane can be formulated by the rule of similar triangles as

$$(u, v) = \left(f\frac{x}{z}, f\frac{y}{z}\right). \tag{3.3}$$

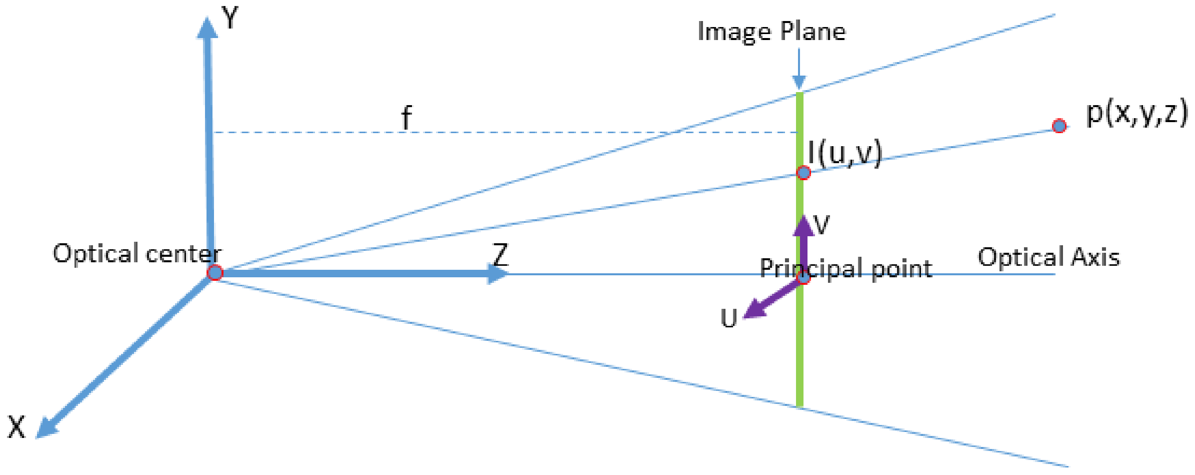


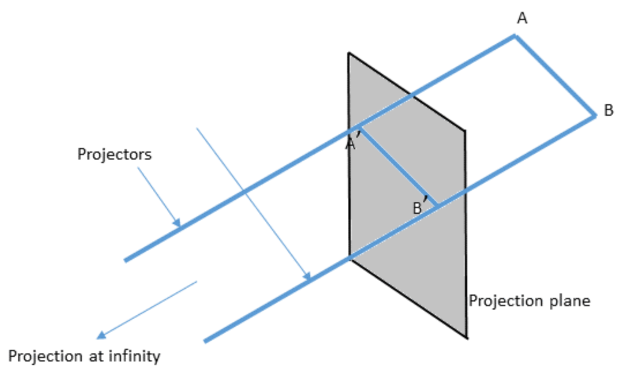
***Figure* 3.3** *Geometric model of the camera projection*

However, this is an ideal case, in practical scenario errors in camera calibration, flaws in digital sensors, unintentional distortion introduced by lenses may cause resulting image having non square pixels. All these problems are tackled by introducing calibration parameters. Two separate focal lengths f_x and f_y in x and y directions, compensate for image having non square pixels. Furthermore, depending on the convention used, center point of the image sensor could be different from its internal origin. Principal point (c_x, c_y) defines the location of the principal point relative to the film's origin corrects the error caused by difference in the origin. These correction parameters are called intrinsic parameters and with the inclusion of the intrinsic parameters the formula becomes

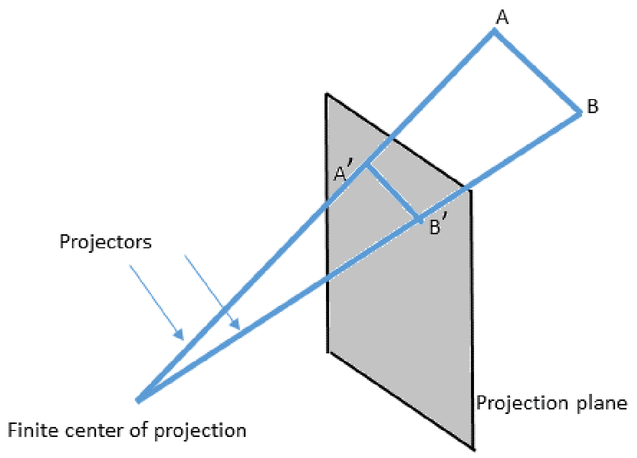
$$(u, v) = \left(f_x \frac{x}{z} + c_x, f_y \frac{y}{z} + c_y\right). \tag{3.4}$$

3.2.1 Orthographic And Perspective Projection

To view 3D objects on 2D display, projection from 3D to 2D is required. Projection of 3D points onto image plane by set of parallel line results in orthographic projection as shown in Figure 3.4(a). Orthographic projection retains the relative proportions of the object but is poor in giving realistic appearance. A simple orthographic



(a)



(b)

Figure 3.4 *(a) Orthographic (b) Perspective Projection.*

projection with plane $z = 0$ can be represented in matrix form [35] as

$$\begin{bmatrix} x_p \\ y_p \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x_v \\ y_v \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_v \\ y_v \\ z_v \\ 1 \end{bmatrix}. \qquad (3.5)$$

If the center of projection is at finite distance from the image plane, it results in perspective projection shown in Figure 3.4(b). In this case the image is inversely proportional to the distance. From the similar triangle rule as in Figure 3.5, it is clear that

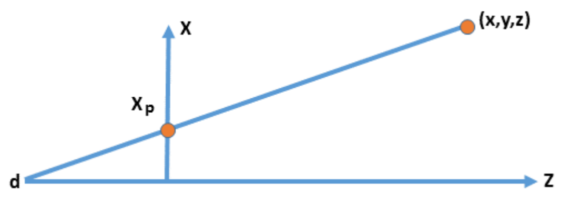


Figure 3.5 *Figure dipicting similar triangle.*

$\frac{x_p}{d} = \frac{x}{z+d}$; $\frac{y_p}{d} = \frac{y}{z+d}$

$x_p = \frac{d.x}{z+d}$; $y_p = \frac{d.y}{z+d}$; $z_p = 0$

Hence, a perspective projection matrix is given as $M_{\text{per}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{bmatrix}$

$$M_{\text{per}}.P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ \frac{z+d}{d} \end{bmatrix}.$$

3.3 Homogeneous Geometrical Representation

Homogeneous coordinates help to easily model the geometric operations of the pinhole camera using matrix representation. This representation enables shifting the principal point by allowing translation operation to be performed as matrix operation. A point (x, y, z) in 3D space becomes (x, y, z, k) in homogeneous coordinate system, so a pixel coordinate (u, v) becomes (u', v', k) with following relation $(u', v', k) = (\frac{u'}{k}, \frac{v'}{k})$ where k is usually selected as 1 for the sake of simplicity. Expressing Equation 3.4 in homogeneous representation gives

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{3.6}$$

The camera intrinsic matrix allows mapping between camera coordinates and pixel coordinates in the image plane.

3.4 Camera Extrinsic

Camera extrinsic parameters describe the position of the camera in the world and the direction it is pointing to [36]. It defines the camera location and orientation in relation to the world frame. The extrinsic matrix comprises of two components, rotation matrix R which is a 3×3 in the left block and translation vector t which is 3×1 to the right

$$\left[\begin{array}{c} R|t \end{array}\right] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right]. \tag{3.7}$$

For the easier processing of the matrix, an extra row $(0, 0, 0, 1)$ is added to the bottom of the matrix which makes the extrinsic matrix a square matrix. This allows consequtive transformations to be formulated as incremental multiplications. For instance, rotation followed by translation can be represented by multiplication of rotational and translation matrices.

$$\begin{aligned} \left[\begin{array}{c} R|t \\ \hline 0|1 \end{array}\right] &= \left[\begin{array}{c} I|t \\ \hline 0|1 \end{array}\right] \times \left[\begin{array}{c} R|0 \\ \hline 0|1 \end{array}\right] \\ &= \left[\begin{array}{ccc|c} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \times \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & 0 \\ r_{2,1} & r_{2,2} & r_{2,3} & 0 \\ r_{3,1} & r_{3,2} & r_{3,3} & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]. \end{aligned} \tag{3.8}$$

By the above transformations, points in world coordinates are transformed to camera centric coordinates. The vector t represent the position of the world origin in camera coordinate, the matrix R represents the direction of world axis in camera coordinate.

3.5 Camera Matrix

Finally, the camera matrix P combines the extrinsic and intrinsic matrices thus describing how a point in the 3D world is projected to the camera imaging sensor.

$$P = M \left[\begin{array}{cc} R_{3\times3} & T_{3\times1} \end{array}\right]. \tag{3.9}$$

where $[R_{3\times3}\, T_{3\times1}]$ is concatenation of the rotation and translation matrices, and M

is the intrinsic matrix. Pixel position (u, v) of world point $(x, vectors, z)$ in the homogeneous coordinate system (u', v', k) is given by

$$\begin{bmatrix} u' \\ v' \\ k \end{bmatrix} = P \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{3.10}$$

4. RENDERING

"I make use of painting to render thoughts visible".

Rene′ Magritte, Belgian surrealist painter

This chapter overviews the 3D graphics with rendering pipeline, followed by discussion of various image rendering techniques with their advantages and disadvantages. The last section of the chapter describes briefly the occlusion handling problem.

4.1 Introduction

Rendering is the last stage in the graphics pipeline that gives final appearance of the object to be shown. To render an image, several inputs are required, which include, camera position, focal length, points, line, polygons of an object, light sources, textures etc. The rendering produces 2D image in the form of color pixels, which can be displayed on a screen. Main functionality of rendering involves vertex transformation, combination of the vertices into lines and polygons, rasterization and applying the texturization. Conventional 3D graphics use scene geometry for rendering which means 3D geometry of the scene is known in hand and graphics pipeline follows modelling, animation and rendering. Synthetic contents are used in this type of rendering and they are added to the image until it looks real enough.

However, there is another type of rendering called image-based rendering which synthesizes the new view from a set of input images. Image-based rendering gained more popularity in recent years and can be used as an alternative option to traditional geometry based rendering.

Typical 3D graphics rendering pipeline is shown in Figure 4.1[37]. Objects are modelled using polygons, lines or points in their own coordinate system. The first stage of the rendering pipeline is the model transformation, which transforms the

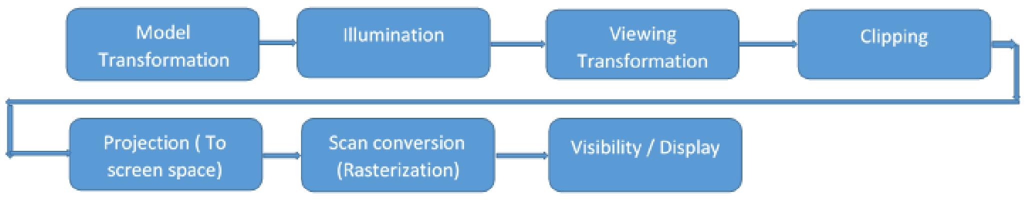


Figure 4.1 *Graphics rendering pipeline.*

object from local coordinate system to world coordinate system. Model transformation is followed by illumination where the model is illuminated according to material properties, light sources and surface property. The third stage of the rendering pipe line is viewing transformation where the model is mapped from world space to eye-/camera space. In camera coordinate system the camera is considered at the origin; this transformation helps in simplifying the projection of 3D to 2D. Clipping process removes the portion of the objects (primitives) outside the camera view. Clipping process is followed by projection, where the objects within the viewing volume are projected onto the image plane (screen space). Rasterization takes place just before the display converts the objects or primitives to pixels which can be displayed. This process may interpolate attribute values such as texture, color etc of a polygon. Final stage of the graphic pipeline is displaying the rasterized pixels.

Depending on the amount of geometric description IBR uses for rendering, rendering techniques are classified in three different categories namely, rendering without geometry, rendering with implicit geometry and rendering with explicit geometry [38] as shown in Figure 4.2.

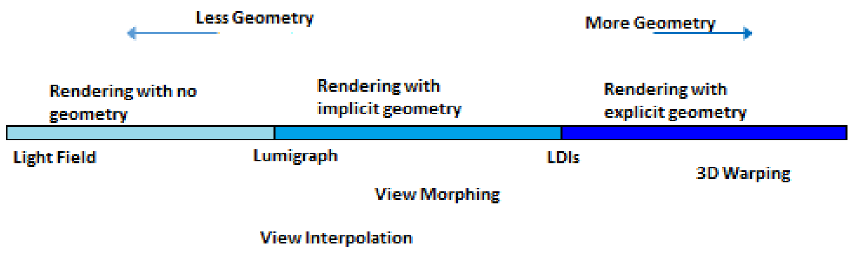


Figure 4.2 *Representation of ray in the light field.*

4.2 Image-Based Rendering

Image-Based Rendering (IBR) has gained wide range of popularity in these years because of its ability to create photo realistic images. IBR uses images as main rendering primitives as opposed to geometry, to synthesize the novel view. IBR helps in solving two important problems in computer graphics: helps to represent the complex scene with simple model and accelerates the rendering speed. Rendering speed is accelerated by replacing the traditional geometric model with image-based representation and thus by detaching rendering time from scene complexity [39].

This technique relies on several input images taken from different view points for rendering the scene. Light field [7] and Lumigraph [40] belong to this category of rendering which either don't use or use very minimal geometric information. In order to address a ray of light, 5 parameters x, y, z, θ, ϕ are required in light field as shown in Figure 4.3 . Light field rendering synthesizes new ray by interpolating set

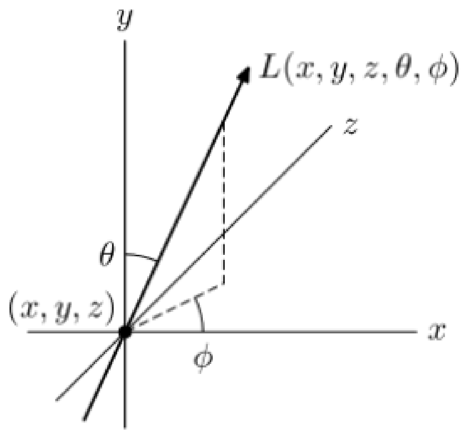


Figure 4.3 *Light field rendering representation [7].*

of neighbouring rays. The idea behind this technique is to represent the light filed as function of position and direction. For practical purpose light field is represented by four parameters, a plane and angle x, y, θ, ϕ or by two plane (u, v) and (s, t) [7]. Light field slab representation is shown in Figure 4.4 . Input images are represented as 2D slices of 4D function. New view is synthesized by extracting and re-sampling the slices. Generation of the new image is formed by different pieces of the original images, no geometric information is used to extract the image values. The image generation mainly depends on re-sampling. To render a new ray r, its intersection with the focal plane is found first. Then, the closest ray r that passes through the

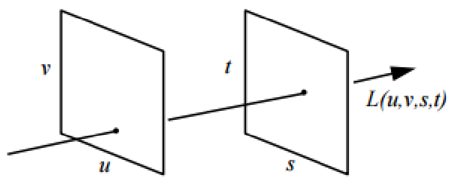


Figure *4.4 Light field slab representation [7].*

camera and intersection point is collected. Finally, light field rendering interpolates r from these new rays by using simple linear interpolation to blend the rays [41]. Aliasing is major problem in light field rendering. Scenes with complex geometry should be sampled very densely in order to avoid aliasing artifacts in the rendering images. Lumigraph is similar to light field but it also allows to include the captured geometry to enhance the performance [42].

4.3 Rendering With Implicit Geometry

These techniques rely on positional correspondences across a small set of images to render new views [43]. Since the geometry is not directly available, the term implicit is used. New view is generated by manipulation of the positional correspondences. Common techniques which fall under these categories are View Interpolation [44], View Morphing [45] and Joint View Interpolation [46].

- **View Interpolation:** This method takes advantage of consistency of sequence of images taken from the close view point, which means there is negligible difference in the objects of the image shot from the close view point. View interpolation [44] uses camera position, orientation and range data of the image to compute the pixel correspondence between images. These correspondences are stored as morph maps. Depending on user control, in between images are created interpolating stored morph map. However, this method has the drawback that the intermediate new views may not be always geometrically correct.

- **View Morphing:** View morphing [45] is the special case of view interpolation which preserves the shape of the morphed image. This technique produces the new view of the original scene maintaining smooth and realistic transition. If the input images produced by the camera are parallel, then the intermediate view produced is the linear combination of the input images thus ensuring the geometrical correctness of the intermediate view. However, if the images are not parallel, a special pre-warping technique is employed to rectify the input images thus ensuring the parallel scan lines. Post warping stage is applied to unrectify the intermediate images.

- **Joint View Interpolation:** This method handles the weakness of previous morphing technique, which lacks the depth information while morphing. The algorithm has two steps. The first step is seed selection, where the algorithm matches the point of interest with high texture in order to function as seed point for bootstraping a region to grow. In the second step, the algorithm starts to grow from the most reliable match and propagates in the neighborhood from most textured pixel to the less textured pixel [46].

4.4 Rendering With Explicit Geometry

Rendering with explicit geometry uses the description of the source. Descriptions are in form of scene geometry, texture map, surface reflection, etc.

- **Depth Image-Based Rendering:** DIBR is the process of creating virtual views from color image and the associated depth map. Virtual view synthesis can be explanied by two step process.
  In the first step, image pixel points are projected back to the 3D world using their associated depth data. In the second step, these 3D space points are projected onto the image plane of a desired virtual viewing position. This projecting from 3D to 2D and back is called 3D image warping. Consider a point in 3D space projected onto 2D image planes $I1$ and $I2$ at points $m1$ and $m2$ as shown in Figure 4.5. Projection of these two points resulting in two views can be written using projective equation as [8]

$$m_1 = \frac{1}{Z_1} K_1 . R_1 \Big[\ I| - C_1\ \Big] M \tag{4.1}$$

$$m_2 = \frac{1}{Z_1} K_2 . R_2 \Big[\ I| - C_2\ \Big] M \tag{4.2}$$

where K_1,R_1 C_1 and K_2 R_2 and C_2 are intrinsic matrix, rotation matrix and

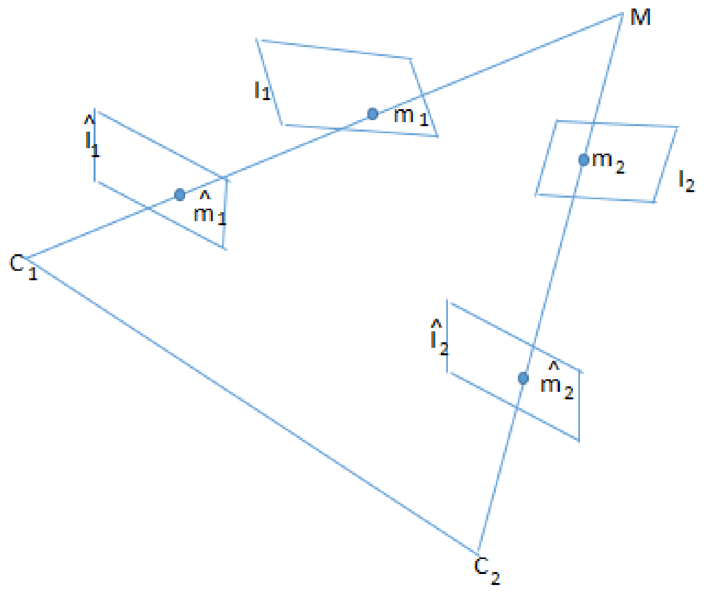


Figure 4.5 *Depth image-based rendering principle [8]*

translation matrix of camera 1 and 2 respectively, m_1 and m_2 are homogeneous coordinates of the point projected on to the reference view and virtual view, Z is the depth of the 3D point M. Expressing the coordinate system of camera 1 as the world coordinate system, the above projective equation is simplified as

$$m_1 = \frac{1}{Z_1} K_1 . \Big[\ I|0\ \Big] M \tag{4.3}$$

$$m_2 = \frac{1}{Z_1} K_2 . R \Big[\ I| - C\ \Big] M \tag{4.4}$$

R and C are the rotation and translation matrix of second camera. Substituting equation 4.3 to 4.4, the classical 3D image warping can be obtained as

$$Z m_2 = Z K_2 R {K_1}^{-1} + K_2 C \tag{4.5}$$

The above equations require the projection matrix of the virtual camera relative to the reference camera and the virtual camera's intrinsic parameters for image synthesis. However, there are certain limitations in producing virtual view for a desired camera position.

- Some of the visible 3D points in the original view will become hidden (invisible) from the virtual camera position, but still can appear in rendered view because of incorrect sampling.

  - Data projected onto the image plane is irregularly sampled at possibly non-integer coordinates thus requiring interpolation.
  - Some pixels at the virtual camera position will find no correspondences at the reference view (dis-occluded pixels).
  - Errors in depth maps are directly proportional to the distance between reference and virtual camera. Smaller distance contributes to less error while the bigger distance contributes visually noticeable error in the depth map.

- **Rendering using triangular mesh:** An issue with Depth Image-Based Rendering (DIBR) is that the input pixel of the reference view is not mapped to the integer pixel position in the virtual view. One approach to overcome this problem is to interpolate the neighboring pixel positions. Another complication is multiple pixels from the reference view can project to the same integer position in the virtual view. A foreground pixel in the reference view can occlude background pixel in the rendered view, which results in overlapping pixel. Finally, synthesized views may have some invisible area from the original view point. The above mentioned issues are addressed by rendering using triangular mesh.

  In mesh-based rendering, given mesh is provided as a set of triangles (i.e. faces) where each vertex is defined in some 3D space. This is the space where optical center of a given camera is the origin of a coordinate system and z-axis faced forward from the camera. Along with the 3D coordinates, each vertex contain also corresponding pixel coordinate from the original color image. This pixel coordinate directly corresponds to so-called *UV-attributes*. These attributes are often used to obtain corresponding texture coordinate for a particular pixel (also called *fragment*) in a rendered image.

  Virtual view is synthesized by re-projecting every vertex of a triangle to a desired camera sensor. This may result in irregularly-sampled points. Filling of the resulting triangle with the correct color or texture values in the domain of the rendered image known as a *Rasterization*.

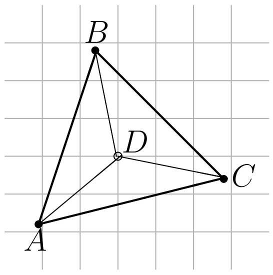


Figure 4.6 *Illustration of rasterization principle.*

On the Figure 4.6 example of rasterization for one projected triangle is illustrated. Here, the thin regular lines correspond to integer pixel coordinates, and their intersections are geometric centres of regularly-sampled pixels.

After the projection of coordinates of vertices A, B and C on a virtual image grid, one can recover bounding box around triangle, defined in the pixel coordinates. From the Figure 4.6, three lines formed by pairs of points can be defined as AB, BC and CA , and their equation parameters can be estimated as

$$y = a_l x + b_l, \tag{4.6}$$

where parameters a_l and b_l are different for every line.

Now for every pixel (x_i, y_i), within the bounding block one can estimate signed difference between a pixel and each line as

$$d_l(x_i, y_i) = a_l x_i + b_l - y_i. \tag{4.7}$$

Here, the positive difference would mean that a pixel is located from right-hand side of the line, and the negative difference would mean otherwise.

For all the image pixels within the triangle, all the $d_l(x_i, y_i)$ values will have the same sign. Exact value of the sign (positive or negative) depends on how the sequence of vertices was defined in the original mesh, clock-wise or counter-clock-wise. Thus, one can estimate the pixels belonging to a particular rasterized triangle.

Using the weighted interpolation, one can estimate attribute value at the coordinate of the rasterized pixel D, taking attributes of given vertices as a source. Nevertheless, this simplistic weighting approach does not take possible projective distortion into the consideration. If the projected 2D triangle corresponds to a significantly slanted triangle in the 3D space, simple 2D weights will no longer be adequate. In order to obtain correctly interpolated attribute values, one should use distances $|DA|$, $|DB|$ and $|DC|$ defined in the 3D space, thus accounting for real size of an underlying triangle.

4.5 Difficulties With Rendering

View plus depth is one of the well known 3D representations. When a new new view is synthesized from this representation, holes are created in the new image due to disocclusion. Disoclusion occurs because some of the points that were occluded in the original image become visible in the new synthesized view. So, the image warping has to be followed by filling the disoculded holes. The aim of the occlusion filling procedure is to fill the holes produced by image warping and to make the image look natural [47]. There are several ways to fill the dis occluded holes. One of the simplest occlusion filling techniques is by assuming that the occluded pixels belong to the background. Then, the simplest appropriate solution is to fill dis-occluded regions by copying neighboring background pixels. However, such simple approach results in artifacts. In order to avoid such artifacts, several sophisticated algorithms can be used. In general, hole filling can be categorized as realtime and non realtime. Some of the realtime hole filling methods are constant color filling, linear color interpolation, background extrapolation etc. Non realtime methods use much more complex algorithms than the realtime methods. They mainly use the whole image for hole filling than just the borders [48]. Some of the non realtime methods are copy and paste [49] and texture synthesis [50]. However, there is no single algorithm which suits all cases. The quality of the hole filled image mainly depends on content and particular algorithm used.

Another difficulty with DIBR is due to sampling. The projected data is sampled irregularly. So it needs to be sampled onto regular grids. Holes created due to this is filled with nearest neighbour interpolation.

5. APPROACHES

"A theory can be proved by experiment; but no path leads from experiment to the birth of a theory".

Manfred Eigen

5.1 Introduction

Visual quality of the rendered image depends on the model. Complex models produce high rendered quality compared to simpler models. However, in order to render complex models the number of polygons or pixels required are prohibitively high. Even for the high speed machines, it takes considerable amount of time to render such high amount of polygons/pixels. One of the options in this case is to study the complex model to find the ways to reduce the complexity, without much compromising the visual quality.

When view and depth are given, DIBR is used as one of the methods for rendering. Image warping algorithm of DIBR renders each pixel to the screen. It is not possible to exploit the advantage of graphics hardware in image warping. So the natural choice in this case is to use the mesh-based rendering. Mesh-based rendering fit very well in graphics hardware. Using the mesh-based rendering, it is possible to take the advantage of geometry of the depth map and simplify the model. In this thesis an attempt is made to study the quality of the rendered image by reducing the scene complexity by various techniques using the mesh-based rendering.

5.2 Approaches

Mesh simplification is performed in several ways. Restricted Quadtree [51] [52] is one of the Quadtree construction approaches for mesh simplification, However, it mainly concentrates on terrain data set. The goals of the terrain visualization system are dynamic scene management, quick adaptive triangulation, multiresolution rendering and high performance geometry rendering. These approaches consider texture image for Quadtree construction and do not exploit the nature of depth map or discontinuities of the depth for Quadtree construction which is not suitable for our purpose. In [24], binary space partition method is proposed. However, this method mainly aims at compression of the depth map. On the other hand, [25] uses restricted Quadtree approach to perform depth map simplification which also aims at reducing the rendering cost by taking the advantage of the scene geometry which is in line with the goal of our thesis.

Several other approaches such as simple decimation of the edge and merge coplanar faces are tried. No one of these methods are fast enough and they take considerably long time to decimate the faces from the full mesh to the desired mesh. Approaches selected in this thesis are quite close to the realtime simplification and give better rendered view on the reduced mesh.

In general, decimation techniques we have used belong to one of the three categories. In the first category, full depth map is directly simplified by removing its pixels. Simplified depth map is triangulated later. In the second approach, first the the full mesh is constructed from the whole depth map. Full mesh is later simplified using user given threshold. In the third approach, full depth map is used to find the homogeneous area within the depth map. Mesh is constructed from the homogeneous area. Larger homogeneous area leads to bigger triangles resulting in higher simplification.

In this section, we use two of the existing methods along with three of our approaches to comparatively study the quality of the rendered image after decimation. The first approach uses naive decimation of the pixels and is followed by two other methods which use existing mesh decimation from the full constructed mesh. The final approach uses two different Quadtree based approaches for construction of the mesh depending on certain criteria. Finally, DIBR is used to compare the quality of the rendered image in all decimated approaches.

5.2.1 Direct Decimation Of Depth Map

This is one of the simplest approaches for simplification of rendering complexity. Keeping the same resolution of a input color image one can decimate underlying depth map which directly results in the lower rendering complexity. Decimation of depth map can be done in many different ways, one may be just by picking every other pixel from the full depth image, or taking the median of each block, etc. In[53], downsampling of the depth map is performed by choosing the median value of the selected window.

$$img_{(down)}(x, y) = median(W_{(m\times m)}). \tag{5.1}$$

where $W_{(m\times m)}$ represents the $m \times m$ window, and m is the scaling factor for down sampling the image.
In this thesis we consider, a simple decimation approach where, every 3^{rd}, 4^{th}, 5^{th} and 6^{th} pixels are dropped. Triangular mesh is constructed from the decimated depth map. Special care is taken to drop the discontinues depth edges. In order to remove the depth discontinuity, calculate the absolute difference between the disparity in horizontal and vertical directions. While triangulating the depth map, first check if the sum of the absolute difference of disparity between the neighbouring pixel is zero. If zero then there is no depth discontinuity and triangle can be constructed, otherwise no. Size of the rendered image is of the same resolution as the full textured image. Direct decimation beyond certain reduction, lead to amplified artifact near depth discontinuity due to undersampling.

Since direct decimation may lead to amplified artifacts, the depth map can be passed through a filter for smoothing. Passing the depth map through the average filter attenuates the high frequency components leading to over smooth edges. This causes distortion in rendered view especially at the object boundaries. Moreover, there may be some impulsive noise in the depth map due to the presence of discontinuous edges. So, median filter, which is not prone to impulsive noise and preserves edges is used with suitable radius. Smoothed depth map after filtering is then utilized for rendering. This technique gives slightly better results as compared to the direct decimation without filtering.

5.2.2 Adaptive Remeshing

This approach belongs to second type of simplifications, where the full mesh is reduced to simplified mesh. This is illustrated in Figure 5.1. A full mesh is con-

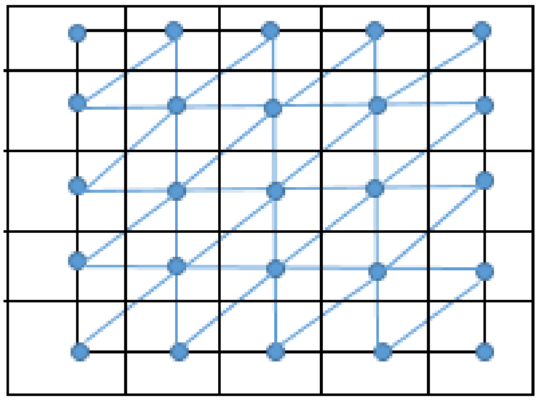


Figure 5.1 *Construction of full triangular mesh from depth image*

structed from the grid pixels of the depth. Simplification function takes this full mesh and depending on the reduction factor, reduces the faces and vertices while attempting to preserve the overall shape of the original mesh. The algorithm used in this approach is based on [30], where faces and vertices are reduced by iteratively collapsing the shortest edge to the mid point. As shown in Figure 2.16, edge collapse operation combines adjacent vertice v_t and v_s into one vertex v_s. Two adjacent faces $\{v_t, v_s, v_r\}$, $\{v_s, v_t, v_l\}$ and vertex v_t are removed in the process.

In this thesis, we used a built in function from MATLAB^{TM}. The algorithm used in this function has few simple steps. The input to the algorithm is the fully constructed mesh. In the first step, the algorithm finds the edge for collapse. The edge with shortest length is selected for collapse. Next step is to select the new vertex point for the collapsed edge. In this algorithm, mid point of the selected edge is selected as the new vertex. In the third step, the algorithm combines the collapsed edges of neighbouring faces to single edge. In the fourth step, remove the faces connected with the eliminated edge. In the fifth step, update the length of the remaining edges. Finally, replace the vertices that were using the end point to start point. Repeat the process until no more edges to collapse or condition met.

Since this approach uses much more sophisticated edge collapse mechanism than the simple decimation and also the decimation is performed on the fully constructed mesh, the object boundaries are better preserved. Hence, the quality of the rendered view is very well maintained even for much higher reduction factor.

5.2.3 Surface Simplification Using Quadric Error Metric

This approach works on full polygon meshes. It uses quadric error metric for contraction of edge. The algorithm [29] reduces the complexity of the mesh by contractions of the vertex pairs while maintaining the overall surface. The algorithm is also capable of joining the unconnected regions of the mesh.

This algorithm uses quadric error distance measure and the edge collapse operator. A vertex of the mesh is associated with an error metric and quadric error. For each edge to be collapsed, a new vertex with minimum error is calculated and used for collapsing the edge. There are five different steps in this algorithm.

- First step is the selection of relevant pair of vertices. A pair of vertex (m_1, m_2) is a valid for contraction if (m_1, m_2) is an edge or $||m_1 - m_2|| < p$, where p is user defined parameter. Higher value of p allows unconnected pair to connect together allowing non manifold mesh.
- Second step is to calculate the initial 4×4 symmetric matrix Q_v for all the vertices. Error quadric for vertex v can be found by associating a set of planes with each vertex. The error of the vertex with respect to the set of planes is then defined as sum of the squared distance to its plane [29].

$$\Delta(v) = \Delta([v_x\, v_y\, v_z 1]^T) = Q_v = \sum_{p \in planes(v)} (p^T v)^2, \tag{5.2}$$

  where p is $[a\, b\, c\, d]^T$ which represents the equation of the plane $ax+by+cz+d = 0$. Equation 5.2 can be written as

$$\begin{aligned} \Delta(v) &= (v^T.p)(p^T.v) \\ &= v^T(pp^T)v \\ &= v^T \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} v \\ &= v^T M_p v. \end{aligned} \tag{5.3}$$

For a given mesh, for each vertex, the error matrix of vertex v is the sum of all $M_p(v)$ which is given by

$$Q_v = \sum_{p \in planes(v)} M_p(v), \tag{5.4}$$

where p is a plane that contains triangle with vertex v.

- The third step is to find the target for contraction of each valid pair (m_1, m_2). In a given iteration to select the contraction, a cost has to be defined for a contraction. In order to perform the contraction $(m_1, m_2) \rightarrow \bar{m}$, a new error matrix at $\bar{Q}$, which approximates the error at $\bar{m}$ has to be calculated. This is computed with the additive rule $\bar{Q} = Q_1 + Q_2$. It is not only sufficient to approximate the error matrix at $\bar{m}$ but also required to select the new position for $\bar{m}$. One of the easiest ways is to move m_1 to m_2 , move m_2 to m_1 or move m_1 and m_2 to $\frac{(m_1+m_2)}{2}$ based on which of the methods yields the minimum value $\Delta(\bar{v})$. Because Δ is a quadratic function, linear equation can be used to find its minimum. Thus $\bar{m}$ can be obtained by solving $\frac{\partial \Delta}{\partial x} = \frac{\partial \Delta}{\partial y} = \frac{\partial \Delta}{\partial z} = 0$. This is equal to [54].

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \bar{m} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{5.5}$$

Solving for $\bar{m}$ equal to

$$\bar{m} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{5.6}$$

- In the fourth step all the pairs are sorted.
- The final step is to iteratively remove the edge pair (m_1, m_2) which has the minimum cost. Cost of the all the valid pairs associated with m_1 are also updated.

5.2.4 Quad Tree Decomposition With Constant Blocks

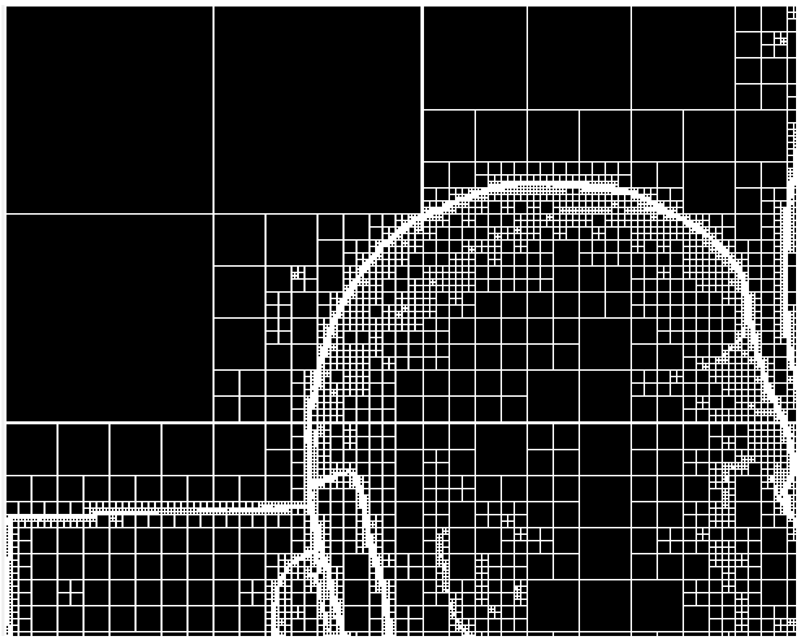
Despite its good performance, the computational complexity of the remeshing approach is prohibitive for realtime applications. Due to the quadratic asymptotic complexity, high resolution meshes require seconds or even minutes to process. The third type of depth meshing method takes regularity of the input depth map as well as their compressibility properties into account in order to speed up the calculations.

Quadtree based approach decomposes a depth image into homogeneous regions which are approximated with a constant value. Quadtree decomposition is an analysis technique that involves subdividing an image into homogeneous blocks [55]. Unlike the other decimation techniques in this approach the structure of the image is used to construct the blocks of homogeneous regions.

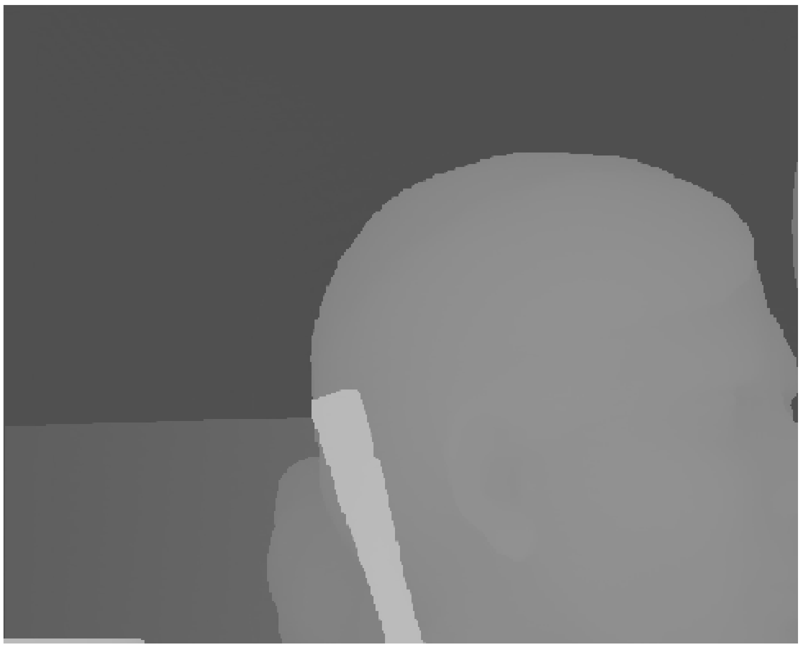
This algorithm divides the initial image into four equal sized blocks. Each block is checked if it meets certain criterion of homogeneity. Blocks meeting the criterion of homogeneity is retained as it is, and the blocks not meeting the criterion are further divided in four equal sized blocks. This process continues until each block meets the criterion. Algorithm can be explained in more details with five simple steps, as listed below.

First step is to divide the image to the required block size. Second step is to test the criterion. In this implementation each block is tested to check if difference between maximum value and minimum value is less than the user defined threshold. Third step divides the block if it does not meet the criterion or keeps the block otherwise. In the fourth step, the result is returned as a matrix, which holds the coordinate values of the depth map along with block sizes. In the fifth step, a special attention is paid to ignore the faces with discontinuous edges and construct the triangles using the matrix returned by the Quadtree decomposition. Value of the threshold is between 0 and 1,irrespective of the value of the image. If the image value is unsigned integer 8, then the threshold value is multiplied by 255 to determine the actual threshold; if the image value is unsigned integer 16 then the threshold value is multiplied by 65535 to determine the actual value of the threshold.

Typical Quadtree decomposition is shown in Figure 5.2(a).


(a)

(b)



(c)

Figure 5.2 *(a) Quadtree decomposition (b) Texture Image [2] (c) Associated Depth map [2].*

5.2.5 Quad Tree Decomposition With Plane Fitting

Peak Signal to Noise Ratio of the rendered image in previous approach is not good because in this case homogeneous blocks are approximated with constant value and not by planar area. However, Quadtree decomposition with plane fit approach, models every block of a decomposition as a planar area. Unlike block based Quadtree, this decomposes the depth image into homogeneous triangular areas. The algorithm is developed using a recursive technique.
Equation of the plane is given as $au + bv + c = z$. By taking several samples of depth within the current block one can estimate underlying depth plane parameters using Moore-Penrose pseudo-inverse

$$\begin{bmatrix} z_1 \\ z_2 \\ z_n \end{bmatrix}^T = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \\ v_1 & v_2 & v_3 & v_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} . \tag{5.7}$$

$$\begin{bmatrix} a & b & c \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \\ v_1 & v_2 & v_3 & v_4 \\ 1 & 1 & 1 & 1 \end{bmatrix}^+ \begin{bmatrix} z_1 \\ z_2 \\ z_n \end{bmatrix} . \tag{5.8}$$

Algorithm 1 Pseudo code of homogeneous Quadtree decomposition

```
Require: Depth Image D, Threshold T
 1: procedure QUADTREE($D, T$)
 2:     $blockDivisionSize \leftarrow [1024, 512, 256, 128, 64, 32, 16, 4, 2]$
 3:     $count \leftarrow 0$
 4:     for k in blockDevisionSize do
 5:         Divide the image as block
 6:         for each block do
 7:             if max(block) − min(block) <= T then
 8:                 NewImage(row,row+ blkSz,col, col+blkSz) $\leftarrow count$
 9:             end if
10:             $count \leftarrow$ count+1
11:
12:             if row < originalRow AND col < originalCol then
13:                 S(row,col) = blkSz
14:             end if
15:             if max(block) − min(block) > T AND blkSz <=2 then
16:                 NewImage(row,row+ blkSz,col, col+blkSz) $\leftarrow count$
17:             end if
18:             $count \leftarrow$ count+1
19:
20:             if row < originalRow AND col < originalCol then
21:                 S(row,col) = blkSz
22:             end if
23:         end for
24:     end for
25: end procedure
```

where + is the matrix pseudo inverse.
Slope m and intercept b of the line ab passing through $(1, 1)$ and (h, w) is given by $m = \frac{(h-1)}{(w-1)}$ and $b = \frac{w-1*(1-h)}{(w-1)}$.

In this binary tree approach is followed. First, the block is divided into half horizontally. For each block, triangles are formed by cutting the block in different diagonals as shown in Figure 5.3. A plane is fitted for these triangles. If the error produced by the triangles is less than a threshold then the block is not divided otherwise the block is further divided vertically. This process is followed until there are no more blocks to divide or the condition is met. Algorithm is listed in 2.

Triangles formed by fitting the planes after decomposition is shown in Figure 5.4.

Algorithm 2 Quadtree with plane fit

```
Require: Depth Image D, Threshold T
 1: procedure QUADTREE WITH PLANE FIT($D, T$)
 2: $Z_1 \leftarrow D(1,1)$
 3: $Z_2 \leftarrow D(1,w)$
 4: $Z_3 \leftarrow D(h,1)$
 5: $Z_4 \leftarrow D(h,w)$
 6: $\begin{bmatrix} a & b & d \end{bmatrix} \leftarrow \begin{bmatrix} 1 & w & 1 \\ 1 & 1 & h \\ 1 & 1 & 1 \end{bmatrix}^+ \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}^T$
 7: $\begin{bmatrix} b & c & d \end{bmatrix} \leftarrow \begin{bmatrix} w & w & 1 \\ 1 & h & h \\ 1 & 1 & 1 \end{bmatrix}^+ \begin{bmatrix} z_2 \\ z_4 \\ z_3 \end{bmatrix}^T$
 8: $\begin{bmatrix} b & d \end{bmatrix} \leftarrow \begin{bmatrix} \frac{1-h}{w-1} & \frac{w-1*1-h}{w-1} \end{bmatrix}$
 9: $\begin{bmatrix} a & b & c \end{bmatrix} \leftarrow \begin{bmatrix} 1 & w & w \\ 1 & 1 & h \\ 1 & 1 & 1 \end{bmatrix}^+ \begin{bmatrix} z_1 \\ z_2 \\ z_4 \end{bmatrix}^T$
10: $\begin{bmatrix} c & d & a \end{bmatrix} \leftarrow \begin{bmatrix} w & 1 & 1 \\ h & h & 1 \\ 1 & 1 & 1 \end{bmatrix}^+ \begin{bmatrix} z_4 \\ z_3 \\ z_1 \end{bmatrix}^T$
11: $\begin{bmatrix} a & c \end{bmatrix} \leftarrow \begin{bmatrix} \frac{h-1}{w-1} & \frac{1-(1-h)}{w-1} \end{bmatrix}$
12:     for $x = 1 \Rightarrow w$ do
13:         for $y = 1 \Rightarrow h$ do
14:             $z1_a \leftarrow a.x+b.y+d)$
15:             $z1_b \leftarrow b.x+c.y+d$
16:
17:             if y >(x.b) +d then
18:                 E1(y,x) = $abs(D(y,x) - z1_b)$
19:             else
20:                 E1(y,x) = $abs(D(y,x) - z1_a)$
21:             end if
22:             $z2_a \leftarrow a.x+b.y+c$
23:             $z2_b \leftarrow c.x+d.y+a$
24:
25:             if y >(x.a) +c then
26:                 E2(y,x) = $abs(D(y,x) - z2_b)$
27:             else
28:                 E2(y,x) = $abs(D(y,x) - z2_a)$
29:             end if
30:         end for
31:     end for
32: end procedure
```

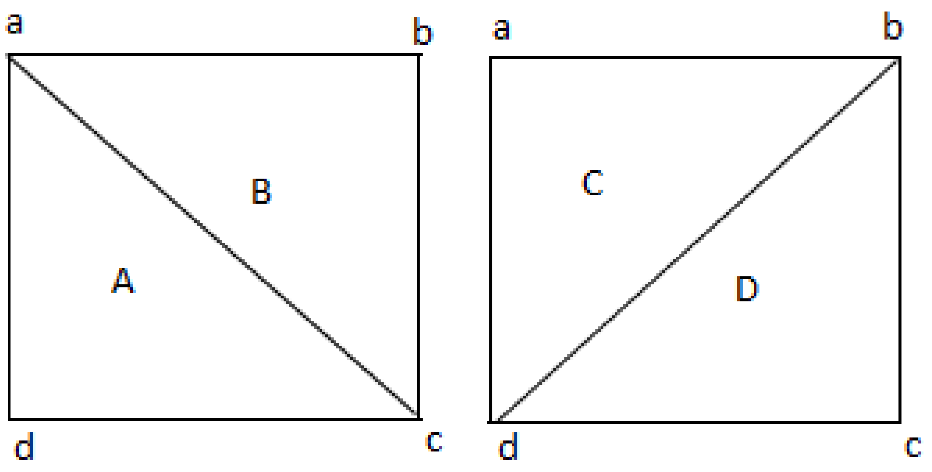
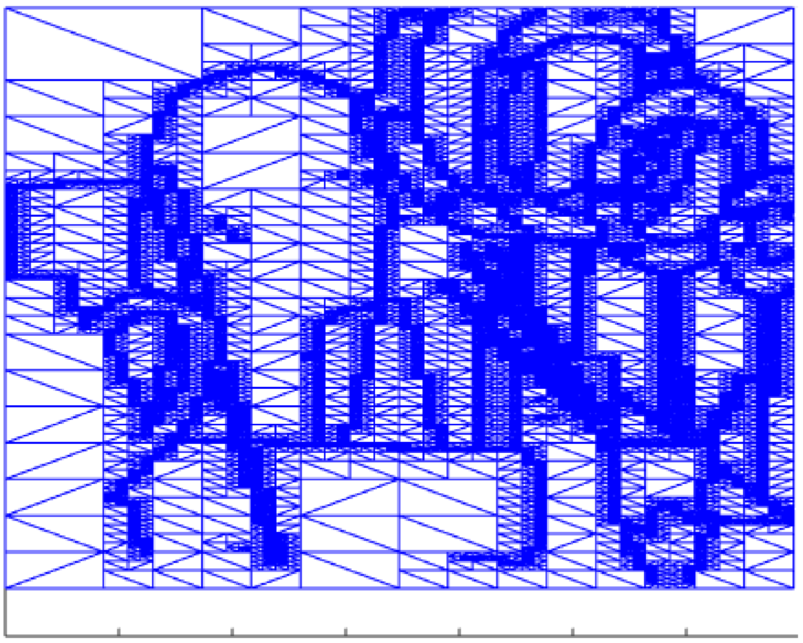
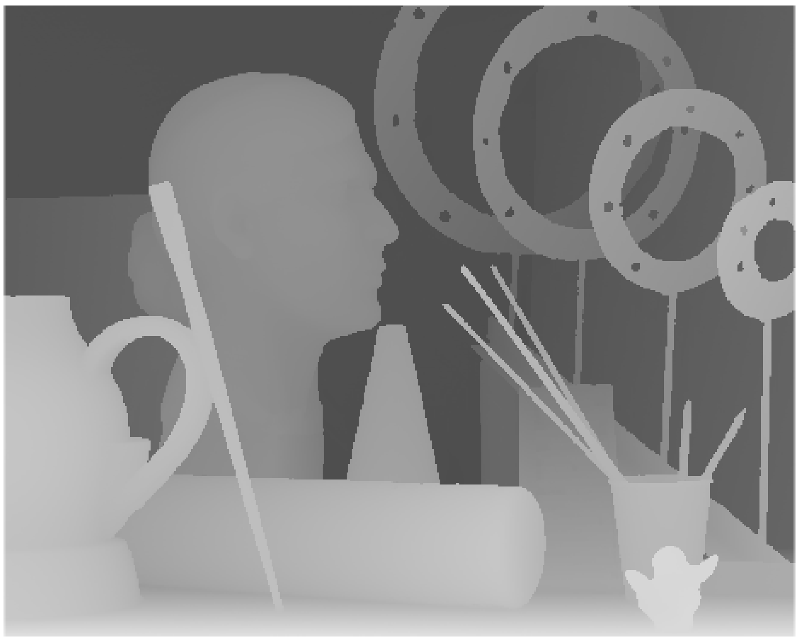


Figure 5.3 *Triangles showing plane cut in fit Plane Quad Tree decomposition.*


(a)

(b)



(c)

Figure 5.4 *(a) Quadtree decomposition with plane fit (b) Texture Image [2] (c) Associated Depth map [2].*

6. EXPERIMENTS AND RESULTS

"However beautiful the strategy, you should occasionally look at the results".

Winston Churchill

In this chapter, we analyse the performance of the depth simplification approaches and compare them in terms of objective quality.

6.1 Experiments

In order to properly evaluate the considered depth simplification approaches from both performance and complexity point of view, we conducted several experiments. For our experiments we have chosen four different data sets from Middleburry collage [2]. Depth image was semi manually preprocessed to remove possible holes. Data sets were carefully chosen to cater all typical scenarios in natural scenes such as different detail levels in both color and depth data, different resolutions, etc.

There are two quality measures used to evaluate the selected approaches. The first one is the PSNR between virtual views rendered using either original or processed(simplified) depth map. The second one is to measure the MSE between the disparity map of triangulated full mesh and the processed depth map after rasterization.

6.1.1 PSNR Of The Rendered View

Peak Signal to Noise Ratio (PSNR) is the ratio between the maximum possible power of a signal and the power of corrupting noise. PSNR is calculated from the MSE, it is expressed in logarithmic scale because many signals have wide dynamic range.

$$\begin{aligned} MSE &= \frac{1}{N}\sum_{i=1}^{N}(c_i - \bar{c_i})^2 \\ PSNR &= 10log_{10}\left(\frac{MAX_z^2}{MSE^2}\right) \\ &= 20log_{10}\left(\frac{MAX_z}{MSE}\right) \end{aligned} \tag{6.1}$$

Where c_i and $\bar{c_i}$ are refered and processed signal. N is the number of samples(pixels) and MAX_z is the maximum possible pixel value assuming minimum is zero.

PSNR is calculated to compare the quality of rendered view using processed depth versus that of using original depth [56]. Occluded areas are created in synthesized new view. In order to attain correct comparison, pixels that are dis-occluded during rendering process are excluded to make the comparison independent on the particular hole fitting approach. For color images, we calculate PSNR independently for each color channel and then calculate the mean between the three channels.

For PSNR calculation two depth maps are used. One is the original depth map and the other with warped view. Both are preprocessed for hole removal. These two depth images are used for left to right correspondence check to find the valid area. PSNR of the valid area of the rendered view is then calculated as given below.

$$MSE = \frac{\sum (I_r - I_g)^2 * ValidMat}{\sum ValidMat * 3} \tag{6.2}$$

$$PSNR = 10 * log_{10}\frac{256^2}{MSE} \tag{6.3}$$

where I_r is the rendered image and I_g is the ground truth image.

6.1.2 MSE of Rasterized Inverse Depth Map

In order to objectively measure error introduced during simplification or meshing of depth maps we chose two quality metrics aimed at different aspects of the rendering performance.

While the percentage of BAD pixels is one of the famous metric in research community for measuring quality of estimated depth maps, we consider the quantized nature of this metric to be too constrained for measuring smaller depth degradations.

In contrast, simple mean squared error estimated directly in the inverse depth domain can be more informative since it captures even smallest distortions. Here, inverse depth domain is chosen due to original depth data sets are defined in the so called "disparity" maps, where the signals are quantized in 8 bits integer values. In order to establish mapping between true depth and inverse depth values, focal length and other intrinsic camera parameters were used.

Given the focal length f and baseline value, the depth value Z of the rastered image is converted to inverse depth map D using the formula (6.4).

$$D = f * baseline./(Z) \tag{6.4}$$

6.2 Visual Performance

In this section, visual performance results of the different approaches are presented. Figure 6.1 shows the rendered images paired with the depth map of the rasteraized image. This representation helps to analyse the effect of depth decimation on rendered image and also the quality of the depth map after simplification. Since Art data has many intricacies this is chosen for illustrations.

Figure 6.1 (a) and (b) illustrate original depth map and the corresponding rendered color image using mesh-based rendering. Dis-occlusion artifacts are not filled. As can be seen from the figure, the rendered image shows sharp object boundaries and no rendering artifacts.

Results displayed here are for the reduction rate in the range of 35-45. Reduction rate is the reduction of number of vertices/faces compared to the full mesh.

As predicted, the results of the simple decimation as in Figure 6.1 (c) and (d) shows significant artifacts followed by surface simplification and Quadtree with constant block. This is because as reduction factor increases, number of samples decimated is higher resulting in under sampling scenario, where as the filter depth map gives slight improvement due to smoothing.

Basic Surface simplification used in thesis, does not take boundary constraints [57] into account, as a result of this near the boundaries (Figure 6.1 (i) and (j)) unacceptable artifacts are visible.

Adaptive remeshing (Figure 6.1 (g) and (h)) on the other hand shows better rendered quality of the image, because in this approach not only the geometry is preserved but also the visual appearance [30], even with slightly higher reduction rate (45). Furthermore, it poses very few discontinuous edges still connected.

Quadtree with constant block approach shown in Figure 6.1 (k) and (l) exposes artifacts at the boundaries, and also shows many discontinuous edges still connected together. This is because in this approach homogeneous area is identified only by comparing the gray values of the depth image, discontinuous edges fall in within certain range of threshold is still considered as one whole block. Besides, Quadtree with constant block approach also exhibits bigger holes in the rendered image.

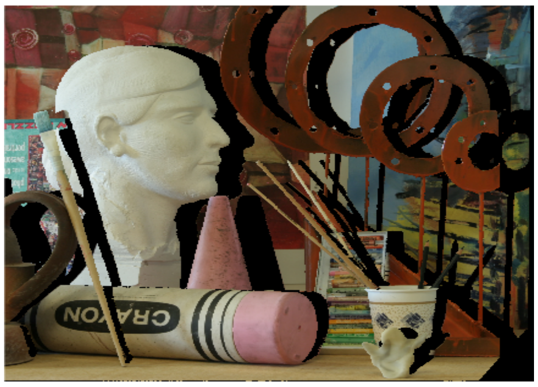
Quadtree with plane fit shown in Figure 6.1 (m) and (n)) on the other hand shows good quality of the rendered image with very few artifacts. The visual quality of the rendered image is very close to the rendered image with true depth map. This approach stands as the best option followed by adaptive remeshing.

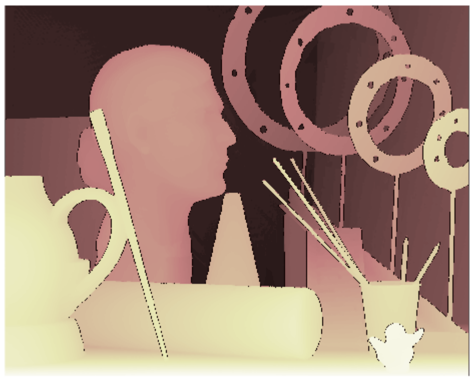
From the inverse depth map, it can be seen that Quadtree without plane fit shows holes in the form of triangles at higher reduction rate mainly along the edges. Surface simplification shows similar artifacts, that it is unable to preserve the edges at higher reduction rate.

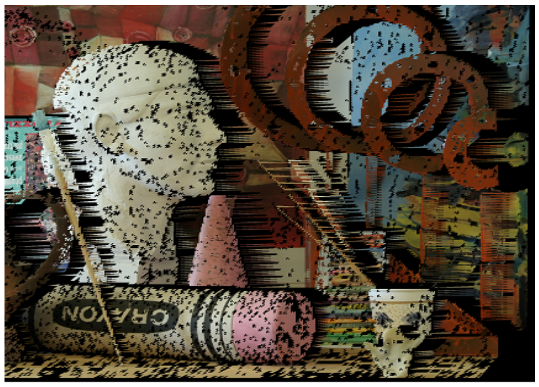
6.3 Numerical Performance

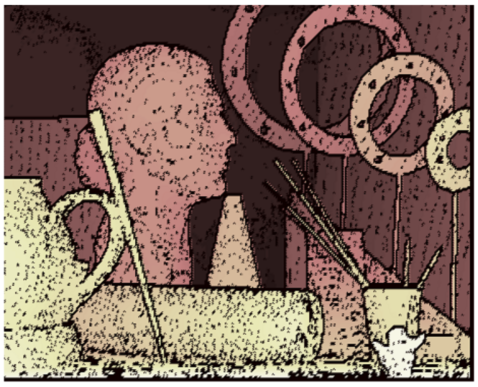
Numerical results, namely PSNR of the rendered color image and MSE of the rasteriazed depth image are presented for four different data sets. For clarity, PSNR and MSE of the images are illustrated together. From Figure 6.2 it is clear that results are consistent across the data sets with very slight variations.

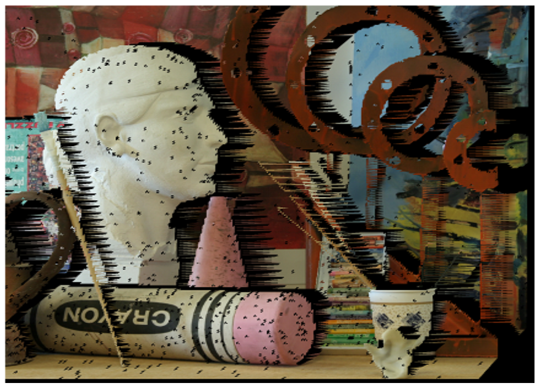
It is clear from Figure 6.2 that Quadtree with plane fit has the best PSNR. Furthermore, PSNR grows slightly while increasing the reduction rate. This technique allows a finer control of the reduction rate at the upper bound as compared to the lower bound. At the lower side with this method the maximum achievable reduction factor is 30. However, for most of the applications this is quite sufficient.

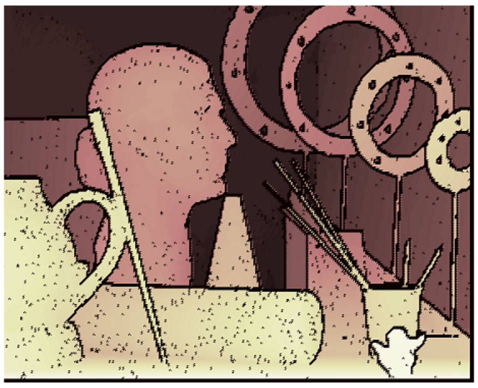

(a)

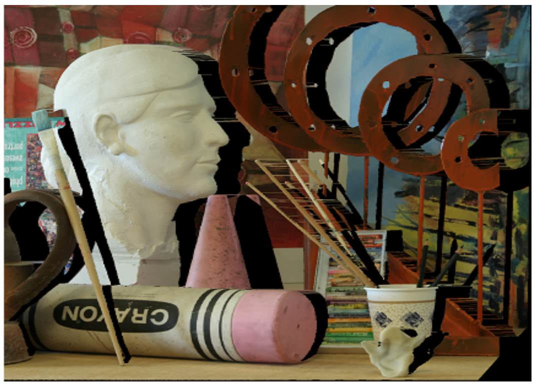

(b)

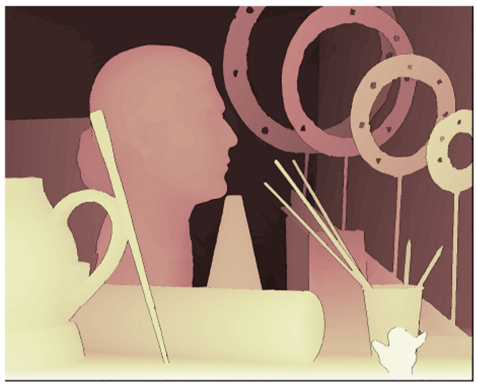

(c)

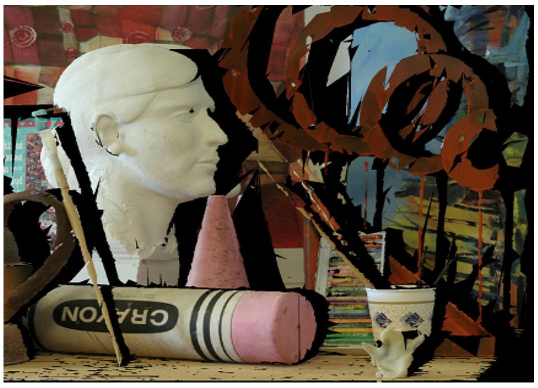

(d)

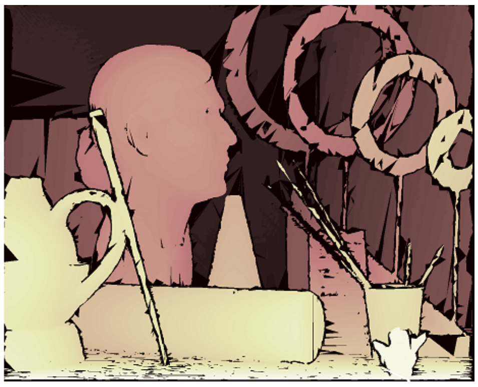

(e)

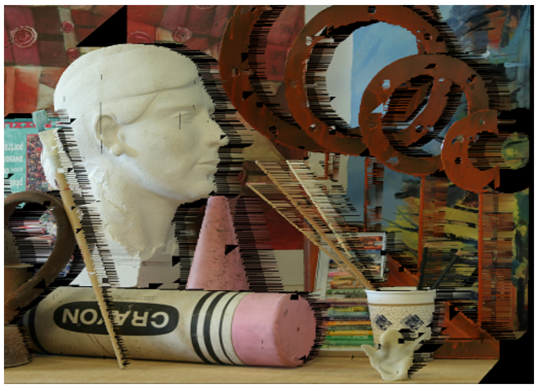

(f)

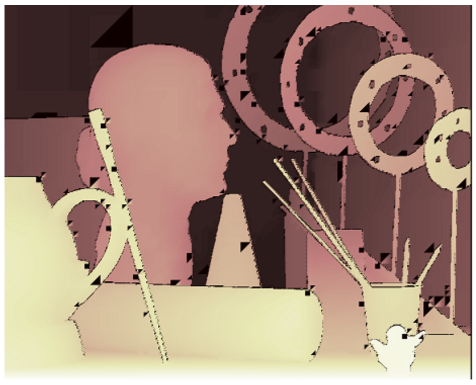

(g)

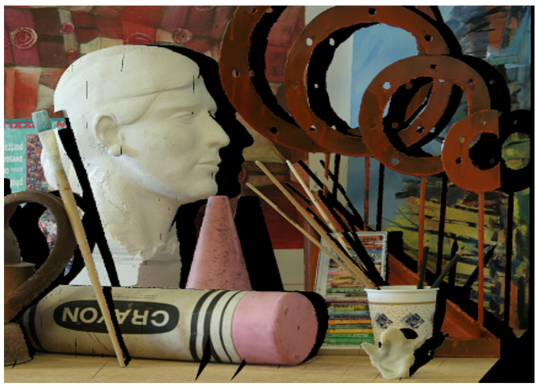

(h)


(i)


(j)


(k)


(l)


(m)

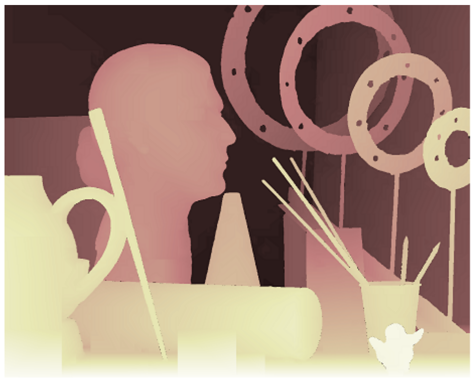

(n)

Figure 6.1 *Left column shows the visual comparison results of rendered image and right column shows its corresponding inverse depth map. (a) rendered image with full mesh; (b) full depth map; (c) rendered image of simple decimation; (d) inverse depth map of simple decimation; (e) rendered image of simple decimation with median filter; (f) inverse depth with median filter; (g) rendered image of adaptive remesh; (h) inverse depth map of adaptive remesh; (i) rendered image of surface Simplification; (j) inverse depth map of surface simplification; (k) rendered image of Quadtree with constant approximation; (l) inverse depth map of constant Quadtree with constant block approximation; (m) rendered image of Quadtree with plane fit; (n) inverse depth map of Quadtree with plane fit are shown*

Adaptive Remeshing on the other hand gives finer control of the reduction rate on the whole range. This works on the full mesh and reduces the vertex/face depending on the user defined threshold. Similar to Quadtree with plane fit, in this approach PSNR grows slightly while the reduction rate increases. Moreover, there is no degradation in the quality even when simplification rate is high which is the peculiar feature of this approach. However, PSNR of this approach is slightly lower compared to the Quadtree with plane fit. This is because, adaptive remeshing performs the decimation by collapsing the shortest edge each time. Collapsing the shortest edge some times causes the facets to be non-planar. This causes reduction in accuracy of the depth map, which in turn results in artifacts in rendered image causing reduction in PSNR. It is noticeable that both computational performance and speed wise Quadtree with plane fit is is better choice compared to adaptive re-mesh.

PSNR of the Quadtree with constant block is slightly less compared to surface simplification. This slight gain PSNR is due to the use of sophisticated quartic error metric for collapsing edges. However, surface simplification takes significant amount of time for simplification of meshes for no apparent gain in PSNR. The longer time taken by this approach is due to the use of complex error metric and simplification criteria used for simplification. It is worth noting in the results that, in Quadtree with constant block approach, simplification of the mesh can go up to the reduction factor of 70 on the upper bound as opposed to the Quadtree with plane fit. However, such high simplification may lead to degraded rendered quality of the image and may not be suitable for practical purpose.

PSNR of simple decimation is comparable with surface simplification and Quadtree in the beginning. However, the PSNR decays exponentially as the reduction rate increases. There is a negligible increase in the PSNR of the filtered depth map compared to the simple decimation. This small increase in PSNR is due to the use of median filter which helps in smoothing the depth map which otherwise leads to abrupt discontinuities.

MSE of rastered image shows inconsistent results across the data set. Quatree with plane fit and adaptive remesh shows lowest MSE. However after reduction rate of 30, MSE of plane fit shoots significantly.

From the MSE it is clear that surface simplification method works best on the data set with few details as compared to the larger details. Surface simplification performs worst on Art and Reindeer data set which have larger details. Surface simplification algorithm does not preserve the topology, may be due to this data set with finer details shows higher MSE.

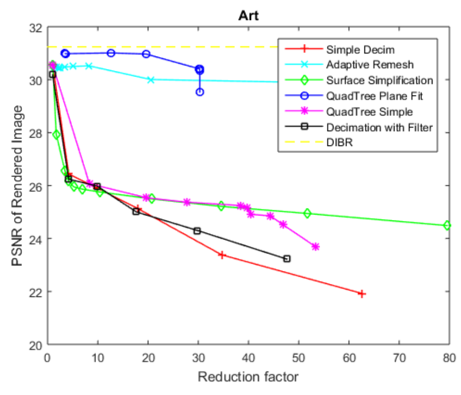
Filtered depth map shows considerably smaller MSE as opposed to unfiltered one even though there is no much significant improvement in PSNR.

As can be seen from Figure 6.2 (g) and (h), Surface simplification and Adaptive remesh fail to work on Cone data set because of its high resolution (1800x1500).

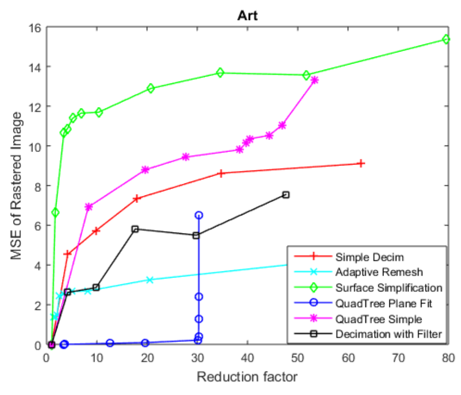
All approaches were compared against DIBR. DIBR shows slightly higher PSNR compared to Quadtree with plane fit. However, this minimal improvement in gain is at the cost of high rendering cost.

6.4 Computational Efficiency

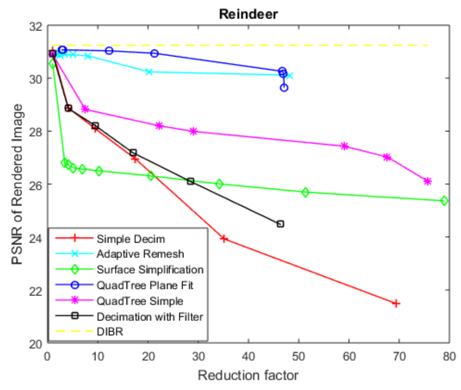
The processing speed of the selected approaches was tested on a computer running Intel core i5, 1.6 GHz. Surface simplification computes optimal vertex position which is associated with error metric and is defined as the sum of squared distances to its incident planes. Due to this as can be seen in Figure 6.3, surface simplification method takes significantly large amount of time especially with less reduction rate. Moreover, PSNR of this approach is not high. Adaptive remesh on the other hand takes slightly more time compared to simple decimation, and other Quadtree approaches. Quadtree with plane fit takes ideal time with high PSNR. Simple decimation, as predicted takes shortest time due to direct decimation of pixels but the PSNR of this approach is not good.



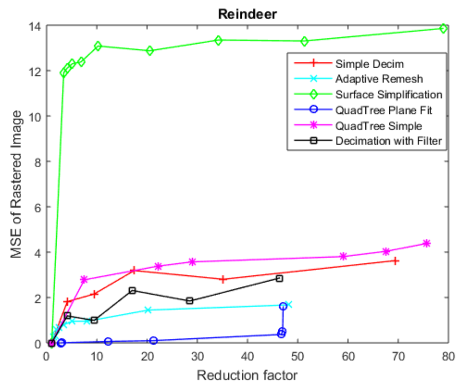
(a)



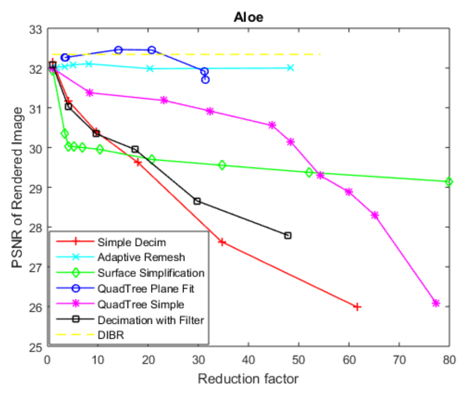
(b)



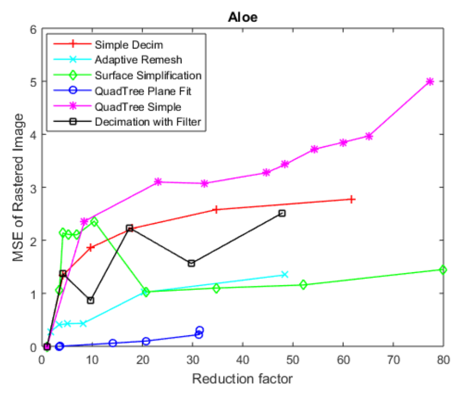
(c)



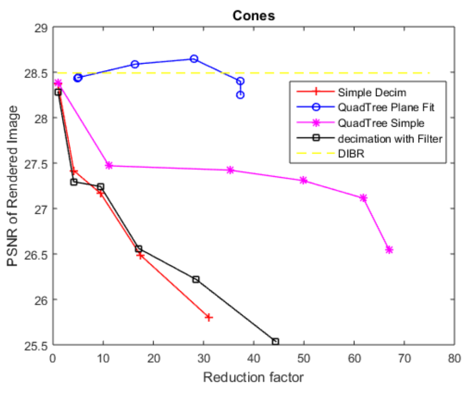
(d)



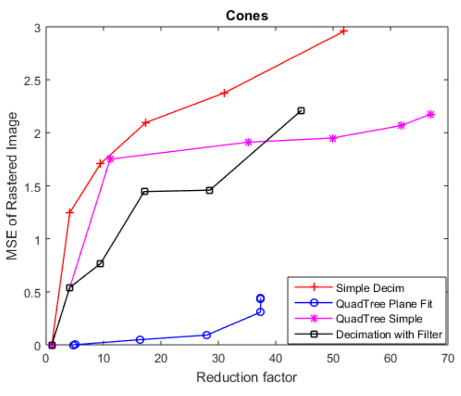
(e)



(f)

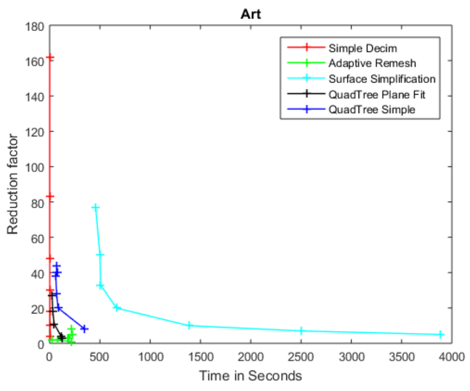


(g)

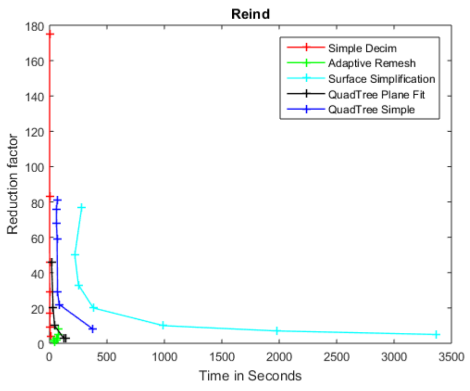


(h)

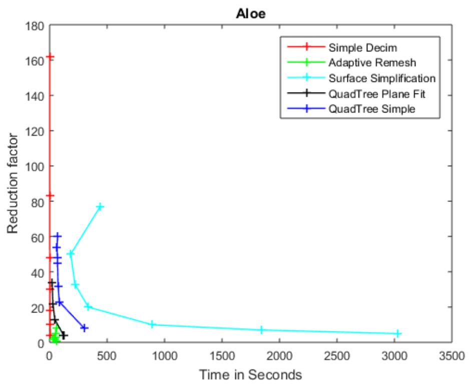
Figure 6.2 *Figure shows (a) PSNR results of Rendered image for art data set; (b) MSE of the rastered image for art data set; (c) PSNR results of Rendered image for aloe data set; (d) MSE of the rastered image for aloe data set; (e) PSNR results of Rendered image for reindeer data set; (f) MSE of the rastered image for reindeer data set; (g) PSNR results of Rendered image for cones data set; (h) MSE of the rastered image for cones data set;*



(a)



(b)



(c)

Figure 6.3 *Computational efficiency plot for (a) art; (b) reindeer and (c) aloe data sets are shown*

7. CONCLUSIONS

"To finish a work ? To finish a picture ? To finish it means to be through with it, ………"

Pablo Picasso

Virtual reality applications are getting wide range of popularity. Two key aspects for this type of applications are the rendering speed and the quality of the rendered images. Rendering of natural scenes with high quality can be problematic due to high resolution of underlying depth maps. In order to reduce the computational cost of rendering, complexity of the scene must be reduced without negatively affecting visual appearance. Depth images have many constant regions which are the potential redundant areas for the simplification. In this thesis we experimented several algorithms to reduce the complexity of the depth map mainly emphasising rendering on low capability device. It is very important to preserve the object boundaries, visual fidelity and the depth discontinuities while simplifying the redundant part of the depth map.

In the beginning, a study has been made on acquisition of depth maps, view plus depth representations, different image-based rendering such as DIBR and mesh-based rendering. Suitable quality evaluation matric for depth decimation is also studied. We took several high quality data sets for our experiments to coverall the typical scenarios.

In order to simplify the depth map, several existing depth decimation techniques were studied and evaluated. Some of the state of the art methods were modified to suit for our scenario. Along with this, we have introduced some of our own approaches. Experiments have shown that, all the approaches have their own merits and demerits and not all the approaches are capable to satisfy required aspects such as, depth discontinuities, visual appearance and object boundaries while simplifying the depth map.

Simple decimation being the fastest produces several artifacts and poor rendered image quality. Slight improvement is observed in filtered depth map, however, it is negligible. Surface simplification on the other hand maintains the rendered quality even with high reduction rate but it is unable to preserve the boundaries. Another demerit of this approach is, it fails to work on high resolution images. Quadtree with constant block exposes larger holes as the simplification rate increases. Adaptive remeshing is the second best choice which produces high rendered quality even with high reduction rate. Quadtree with plane fit is one of the best approaches and outperforms all the other approaches. It produces the highest quality of the rendered image in terms of PSNR. The limitation of this approach is, it cannot go beyond certain reduction factor. However for most of the application simplification beyond certain reduction is impractical.

Comparison of these approaches with DIBR shows that plane fit performs at par with DIBR and for some data sets even better. The performance analysis in terms of speed, shows that the surface simplification using quadric matric is one of the poor performing approaches and rendered quality of the image is not significantly good. Quadtree with plane fit computationally competitive to work on low capability device such as mobile phone.

Conclusion is that, with respect to quality and performance, Quadtree with plane fit outperforms all other methods and can be considered as one of the best approaches among the selected methods.

BIBLIOGRAPHY

[1] Gen2009. Availble at: http://gen2oo9.deviantart.com/art/The-Red-Dress-3D-xeye-158275596.

[2] D. R Szeliski Scharstein. Middlebury Stereo Vison Page. Availble at: http://vision.middlebury.edu/stereo.

[3] NL. UU. "Delaunay Triangulations Computational Geometry Lecture 12". Availble at: http://www.cs.uu.nl/docs/vakken/ga/slides9alt.pdf.

[4] D. P. Luebke. "A developer's survey of polygonal simplification algorithms". *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001.

[5] C.K. Shene. " Mesh Simplification ". *Computer Graphics Forum*, 15(3):C77–C86, 1996.

[6] Scratch Pixel. "3D Viewing Pinhole Camera". Availble at: https://www.scratchapixel.com/lessons/3d-basic-rendering/3d-viewing-pinhole-camera.

[7] M. Levoy and P. Hanrahan. "Light field rendering". *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, pages 31–42, 1996.

[8] W. Sun, Oscar C Au, Sung Him C, and Chun Wing K. "An Overview of Free Viewpoint Depth-Image-Based Rendering (DIBR)". (December):1023–1030, 2010.

[9] T. French. "The Ultimate History Project". Availble at: http://www.ultimatehistoryproject.com/3d-photography.html.

[10] Virtual Reality Society. "History of virtual reality". Availble at: http://www.vrs.org.uk/virtual-reality/history.html.

[11] O. Suominen. "SGN-34006 3D and Virtual Reality", lecture notes on 3D and Virtual Reality.

[12] A. Smolic. "Multimedia Communications 3D Video and Free Viewpoint Video".

[13] C. Fehn. "A 3D-TV approach using depth-image-based rendering (DIBR)". *Visualization, Imaging, and Image Processing*, pages 482–487, 2003.

[14] Sergey Smirnov, Atanas Gotchev, and Karen Egiazarian. Methods for depth-map filtering in view-plus-depth 3D video representation. *EURASIP Journal on Advances in Signal Processing*, 2012(1):25, 2012.

[15] J. Zhu, L. Wang, R. Yang, and J. Davis. "Fusion of time-of-flight depth and stereo for high accuracy depth maps". *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.

[16] P. Biber, H. Andreasson, T. Duckett, and A. Schilling. "3D Modeling of Indoor Environments by a Mobile Robot with a Laser Scanner and Panoraic Camera". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 3, 2004.

[17] D. Scharstein and R. Szeliski. "High-Accuracy Stereo Depth Maps Using Structured Light". *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1(June):I–195–I–202, 2003.

[18] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Proceedings - IEEE Workshop on Stereo and Multi-Baseline Vision, SMBV 2001*, (1):131–140, 2001.

[19] L. De-Maeztu, S. Mattoccia, A. Villanueva, and R. Cabeza. "Linear stereo matching". *Proceedings of the IEEE International Conference on Computer Vision*, (1):1708–1715, 2011.

[20] M. Campen and L. Kobbelt. "Practical Guide to Polygon Mesh Repairing". *Eurographics*, (1):1–55, 2012.

[21] H. J. Qi. "Finite Element Analysis". *Unpublished lecture notes*, 2006.

[22] P. Frey and P.-L. George. "Mesh generation". 32:1–39, 2010.

[23] J. R. Shewchuk. "Lecture Notes on Delaunay Mesh Generation". page 139, 2012.

[24] M. Sarkis, W. Zia, and K. Diepold. *"Fast Depth Map Compression and Meshing with Compressed Tritree"*, pages 44–55. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[25] M. Sainz, R. Pajarola, and Y. Meng. "Depth-Mesh Objects: Fast Depth-Image Meshing and Warping". *Ukpmc.Ac.Uk*, (03), 2003.

[26] Delany Mathwork. Availble at: `https://se.mathworks.com/help/matlab/math/delaunay-triangulation.html`.

[27] J. Arvo. "Graphics Gems 2". 59(2):157–167, 12 2015.

[28] W. J. Schroeder, J. a. Zarge, and W. E. Lorensen. "Decimation of triangle meshes". *ACM SIGGRAPH Computer Graphics*, 26(2):65–70, 1992.

[29] M. Garland and Paul S. Heckbert. "Surface simplification using quadric error metrics". *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97*, pages 209–216, 1997.

[30] H. Hoppe. "Progressive meshes". *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, pages 99–108, 1996.

[31] P. Lindstrom and G. Turk. "Fast and memory efficient polygonal simplification". *Proceedings Visualization '98*, pages 279–286, 1998.

[32] P. Lindstrom. "Image-Driven Simplification". Availble at: `http://www.gvu.gatech.edu/people/peter.lindstrom/papers/tog2000/`.

[33] R. Hartley and A. Zisserman. *"Multiple View Geometry in Computer Vision, 2nd Edition (2004)"*.

[34] Y. Morvan. *"Multi-view video coding"*. PhD thesis, 2009.

[35] Opengl Transformation Pipeline. Availble at: `http://www.songho.ca/opengl/gltransform.html`.

[36] S. Kyle. The extrinsic matrix. Availble at: `http://ksimek.github.io/2012/08/22/extrinsic/`.

[37] Computer Science RPI. "The Traditional Graphics Pipeline". Availble at: `https://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S09/lectures/15_Graphics_Pipeline.pdf`, 2003.

[38] S.B. Kang, Y. Li, X. Tong, and H.-Y. Shum. "Image-Based Rendering". *Foundations and Trends® in Computer Graphics and Vision*, 2(3):173–258, 2006.

[39] M. M Oliveira. "Image-Based Modeling and Rendering Techniques : A Survey". *Revista de Informática Teórica e Aplicada*, IX(2):38–66, 2002.

[40] S J Gortler, R Grzeszczuk, R Szeliski, and M F Cohen. "The Lumigraph". *Proceedings of ACM SIGGRAPH*, pages 43–54, 1996.

[41] J. Yu, L. McMillan, and S. Gortler. "Scam light field rendering". *Proceedings - Pacific Conference on Computer Graphics and Applications*, 2002-Janua:137–144, 2002.

[42] V. Verma and E. Walia. "3D Rendering - Techniques and challenges". *International Journal of Engineering and Technology*, 2(2):29–33, 2010.

[43] H.-Y. Shum and S. B. Kang. "A review of image-based rendering techniques". *Proc. SPIE Visual Communications and Image Processing*, pages 2–13, 2000.

[44] S. Eric Chen and L. Williams. "View Interpolation for Image Synthesis". *Acm Siggraph*, 27:279–288, 1993.

[45] S. M. Seitz and C. R. Dyer. "View morphing". *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, pages 21–30, 1996.

[46] M. Lhuillier and L. Quan. "Image Interpolation by Joint View Triangulation". *Proceedings of the Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado, USA*, 2:139–145, 1999.

[47] L. Azzari, F. Battisti, and A. Gotchev. "Comparative analysis of occlusion-filling techniques in depth image-based rendering for 3D videos". *Proceedings of the 3rd workshop on Mobile Video Delivery - MoViD '10*, page 57, 2010.

[48] J. Overes. "Occlusion filling in depth-image-based rendering". 2009.

[49] Y. Wexler, E. Shechtman, and M. Irani. "Space-Time Completion of Video". *Pami*, 29(3):463–476, 2007.

[50] A. Efros and T. Leung. "Texture synthesis by non-parametric sampling". *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2(September):1033–1038, 1999.

[51] R. Pajarola. "Large scale terrain visualization using the restricted quadtree triangulation". *Proceedings Visualization '98 (Cat. No.98CB36276)*, pages 19–26.

[52] R. Pajarola. "Overview of Quadtree based Terrain triangulation and Visualization". (02):1–16, 2002.

[53] K. J. Oh, S. Yea, A. Vetro, and Yo S. Ho. "Depth reconstruction filter and down/up sampling for depth coding in 3-D video". *IEEE Signal Processing Letters*, 16(9):747–750, 2009.

[54] A. Guéziec. "Surface simplification inside a tolerance volume". *Computer*, 20440(90191):58, 1997.

[55] MathWorks. "Quad Tree Decomposition". Availble at: `http://se.mathworks.com/help/images/quadtree-decomposition.html`.

[56] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Müller, P. H N de With, and T. Wiegand. "The effects of multiview depth video compression on multiview rendering". *Signal Processing: Image Communication*, 24(1-2):73–88, 2009.

[57] M. Garland and P.S. Heckbert. "Simplifying surfaces with color and texture using quadric error metrics". *Proceedings Visualization '98 (Cat. No.98CB36276)*, pages 263–270, 1998.