uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Jing Feng

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Geography)

GRADE / DEGREE

Department of Geography

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Image-based Rendering on Mobile Devices

TITRE DE LA THÈSE / TITLE OF THESIS

Professor A. Boukerche

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Professor M. Benyoucef

Professor M. Elhadef

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Image-based Rendering on Mobile Devices

by

Jing Feng

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Systems Science

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-25768-5
Our file Notre référence
ISBN: 978-0-494-25768-5

Abstract

The research work of this thesis is motivated by a desire to render views of 3D environments in real-time on mobile devices. Considering the weak support for 3D hardware accelerations of mobile devices, it is hard to render geometry-based models at an interactive rate. The utilization of image-based rendering techniques in mobile devices is presented in this thesis. In order to have a systematical knowledge of image-based rendering, we briefly review different image-based rendering techniques. Although image-based rendering is advantageous to geometry-based rendering in certain circumstances, further tailoring is still required to immigrate it to mobile devices. This thesis presents a technique for running image-based rendering tasks on mobile devices. This technique comprises a lightweight rendering algorithm and a local cache management mechanism. Results have shown that the proposed algorithm requires less CPU cycles compared to other algorithms, and that the local buffer management mechanism is robust to communication errors.

Acknowledgements

I take great pleasure in thanking the many people who have helped me in my studies at the University of Ottawa. First, I would like to thank my supervisor, Professor Azzedine Boukerche, for his invaluable guidance and advice. I would like to thank Professor Morad Benyoucef and Professor Mourad Elhadef for their careful reviewing on my thesis and serving on the exam committee, as well as Professor Voicu Groza for serving as the chair of the exam committee. I also would like to thank Dr. Richard Pazzi Werner for granting me permission to use the Virtual Laboratory Model for my research work.

Finally, I would like to extend my thanks to my parents and my wife, whose love and encouragement have always sustained me.

Page iv-v missing.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Due to developments in the processing power and transmission bandwidth of wireless and mobile devices, interactive 3D applications have been made possible for these devices. However, the weak abilities of wireless devices and their dramatically varying times for rendering geometric models make real-time rendering a challenging task. As a result, different approaches to this problem have been proposed. Among these approaches, image-based rendering has attracted the most attention in recent years. Wireless devices have many limitations that should be taken into consideration when utilizing image-based rendering, such as small local storage size, large communication latency, and the high possibility of communication errors. On the other hand, there are many image-based rendering techniques that use a local data set that have been introduced in literature. How to select and migrate suitable image-based rendering techniques to a wireless environment is discussed in this thesis.

In this thesis, an architecture for utilizing image-based rendering techniques for virtual movement of the end users with mobile devices is introduced. Since the majority of the difficulties of utilizing image-based rendering techniques on mobile devices resides in two aspects: communication and rendering, the proposed architecture suggests different techniques to handle each of these two aspects. This architecture includes a general local reference buffer management scheme and a light-weight rendering algorithm. The suggested buffer management scheme can be used with most up-to-date image-based rendering algorithms and the proposed rendering algorithm allows the user to make virtual movement on the mobile devices with less CPU-cycle requests.

This thesis is organized as follows. In Chapter 2, we define and briefly survey different image-based rendering techniques that have beeen introduced in literature. Our defini-

tion and survey of image-based rendering is based on interpolation theory. Compared with existing surveys, we pay more attention to the potential usage of each category of image-based rendering techniques.

Chapters 3 and 4 are related to the local reference image buffer management on the terminal mobile devices. Remoteness is a major topic for utilizing image-based rendering techniques over wireless networks. Thus, an efficient local buffer management scheme is required for potential applications using these techniques over wireless network. We compare the performance of different buffer management policies in Chapter 3 and present an implementation of a buffer management mechanism in Chapter 4.

Considering the weak computing power of mobile devices, a light-weight image-based rendering algorithm is introduced in Chapter 5. This algorithm does not require hardware 3D accelerators and puts loose restrictions on source camera configurations. Moreover, it uses source information that is easy to obtain for both real and virtual scenes.

In Chapter 6, we present a conclusion for this thesis and briefly discuss the future work needed on this topic. Utilizing 3D image-based rendering over wireless networks is an attractive area of research and there are many issues still waiting to be investigated in the future. We do not intend to create a complete list of the potential topics in this area but list only present we can foresee at this time.

Chapter 2

Survey of Image-based Rendering Techniques

Image-based rendering techniques have a long history and are widely used in different areas. Although they look very different from their geometry-based counterparts, image-based rendering techniques are also used to generate novel views of a scene or an object based on certain formats of scene description. In this chapter, we briefly review various image-based rendering techniques and organize them into four categories based on the interpolation of the Plenoptic Function.

2.1 Review of Geometry-based Rendering

2.1.1 Rendering

Rendering means to generate views of an object or a scene based on descriptions of it in a certain format. It is a necessary part of many applications that involve multimedia content, such as video games, virtual reality, simulations, etc. Basically, there are two requirements to a rendering process. The first requirement is the quality of the rendering. For the objects and the scenes that exist in reality, people are familiar with what these objects or scenes should look like. Therefore, higher expectations for the low biased rendering results will be held by the viewers. Even with synthetic objects and scenes, the viewers can still create an image of the view based on their knowledge of shapes and illumination effects. Although it is difficult to formulize the expectations for the rendering quality because of the subjectiveness of the viewers' opinians, a good rendering result

is often described as photo-realistic. In other words, a good rendering result should not look like it has been achieved through a rendering process but instead like a picture taken by a camera.

The second requirement in a rendering process is speed. Most computer applications interact with the user through certain type(s) of user input and output, and this kind of interaction may require high rendering speed. For example, when the user changes his or her position in a game by typing on the keyboard, clicking the mouse, or operating a joystick, the rendering process should update the view for the user immediately after the user's input. It should be noted that a user making continuous movements will require a stream of views rendered at an interactive speed. This requirement is often called real-time rendering because the rendering process is asked to respond to the external event as soon as possible.

In summary, photo-realistic and real-time rendering are the goals of all rendering techniques. It is possible to sacrifice the performance of one aspect for the other in certain cases, for example, by creating roughly accepted views in a flight simulation application in order to allow for fast changes of position. Nevertheless, to fulfill both of the basic requirements is still an ultimate aim for general rendering techniques.

2.1.2 Geometry-based Rendering

The traditional approach for rendering is to express the object or the scene in various 3D geometric models and calculate the final view based on the interactions between the rays and the models. In order to enhance the realism, texture maps, environment maps, shading algorithms, and a combination of these techniques can be used on the surfaces of the 3D geometric models. Although the computer graphics community has achieved great improvements in geometry-based modeling and rendering techniques by developing more advanced display hardware, graphic accelerators, geometric representations, surface reflectance models, and illumination algorithms, photo-realistic real-time rendering is still a difficult task.

First of all, photo-realistic rendering requires an accurate model of the scene. The accuracy requirement of the model does not mean only the precise geometric positions and shapes of the elements in the scene but also stands for the correctness of the reflectance models of all the surfaces in the scene. However, creating an accurate geometric model for a complex scene is a time-consuming and talent-intensive task. The artists need to build detailed models of the scene piece by piece and designate the reflectance model of

each surface in the scene. Although many sophisticated softwares and other assisting techniques – such as 3D digital scanning – have been developed to ease the modeling process, it is still common to find that the real world is too complex to model, especially concerning the reflectance models of the surfaces.

The rendering speed is another pitfall for photo-realistic real-time rendering. In order to generate photo-realistic images, the illumination effects must be created to seem as natural as possible. The modern illumination algorithms require massive computing power which leads to the dedicate graphic accelerators in current PCs and workstations. Even with these advanced hardwares, the ever increasing expectations for rendering quality and speed make photo-realistic, real-time rendering still an intensive computing job.

In brief, as pointed out by Debevec [17], geometry-based models are hard to create and slow to be rendered. These two properties of geometric models prevent them from becoming the universal candidate for the solution to rendering problems.

2.2 Introduction to Image-based Rendering

2.2.1 Usage and Evolution

Image-based rendering is a set of techniques that produce views of objects and/or scenes based on a group of pre-taken images. The pre-taken images are called source images [11], original images [23], and reference images [33] by different researchers. In this thesis, the pre-taken images used by image-based rendering techniques are named reference images. The reference images can be pictures taken by cameras in real scenes or rendered images of synthetic scenes.

Image-based rendering techniques are used widely in different areas of research. In photogrammetry, efforts toward 3D geometric model determination have progressed with camera calibration, image registration, and photometrics. In computer vision, image understanding, object discrimination, and robot navigation are also approaches toward retrieving the 3D geometric models from images.

In the area of computer graphics, the evolution of image-based rendering techniques has three stages. In the first stage, images are used as part of geometric models to enhance the visual realism of the rendered result. In this stage, images are warped so that they can fit well to the surfaces they are attached to. The major part of the rendering tasks relies on the underlying geometric models [12, 21]. In the second stage, images are

used to describe the global illumination effects so that the modeling process can be made easier. This kind of technology is widely known as environment map. It uses images to store the environment surrounding the rendered object so that complex ray tracing can be avoided. It is a very efficient method of simulating a complex mirroring surface [4]. At this stage, images are used to replace part of the geometric descriptions and have shown their advantages in the rendering process.

The third stage of the evolution of image-based rendering techniques in the area of computer graphics is the focus of this thesis. In this stage, images constitute a significant part of scene descriptions. It is possible to generate views of a scene without a geometric model by using some image-based rendering techniques at this stage. Theoretically, image-based rendering techniques in this stage are closer to human vision systems than geometry-based rendering techniques. All of us have the experience that the world surrounding us is recognized and understood first by the images that comes in through our vision. Geometric models are created after the visual recognition of the world. As pointed out by Adelson and Bergen [1], the objects in the environment do not communicate with views directly, and the communication between the viewer and the elements in the environment occurs through the rays entering the eyes of the viewer.

2.2.2 Advantage

Image-based rendering has a higher efficiency than its geometry-based counterpart in both the model creating and view generating phases of the whole rendering process. In the model creating phase, the dominant polygonal modeling primitive is triangle. Triangles are convex in both 2D and 3D; therefore, it is easy to manipulate triangles through linear methods. Moreover, a triangle is an efficient representation of the model when it has a large size. By describing the properties of the three vertices of a triangle, the pixels inside it can be described by duplicating or interpolating the properties of the vertices. However, the efficiency of triangular expression comes into question when the size of the triangle decreases. At an extreme state, if all of the three vertices of a triangle project to the same pixel in the rendered view, the description efficiency of the model based triangles will be degraded to an unacceptable level. In order to enhance the accuracy of the geometric model, it is a common practice to use small triangles. Such practice results in not only a more intensive image generating process, but also in a larger model size. For example, Yoon [60] mentions a model with 1.7 million triangles and 1.3GB data size. On the other hand, an image is an efficient

sampling of a scene at a viewpoint. It trades accuracy and efficiency for the resolution of the image. Therefore, images can be more flexible than geometric models. In addition to this flexibility, image-based models are much easier to get than geometric models, especially for the real environments. The availability of inexpensive image acquiring hardwares, such as multi-mega-pixel digital cameras, can offer high resolution images as scene descriptions at a very low cost. Compared with the elaborate geometric model creation process, the image-based models are very attractive to industries.

In the view generating phase, geometric models rely on a multiplestage rendering pipeline that includes model transformation, view transformation, lighting, projection, clipping, hidden surface removal, and screen mapping. As a result, the computing cost of geometry-based rendering is dependent on the complexity of the objects or the scenes, especially the number of polygons or voxels used in the geometric description. The intensive computing of the geometric rendering pipeline leads to expensive and dedicated graphic hardwares in modern computer architecture. The absence of such hardwares in certain computers, such as mobile devices, may introduce difficulties in rendering high quality views on these computers. There is no fixed hardware pipeline for image-based rendering. First of all, image-based rendering produces the final view by warping or interpolating the reference images, which does not rely on dedicated hardware accelerators. Second, the diversity of image-based rendering makes it difficult to create special hardware accelerators. Moreover, the computing cost of image-based rendering tasks does not depend on the complexity of the scene. Instead, it is related to the resolution of the rendered result and the reference images. Thus, it is possible to control the cost of image-based rendering without any knowledge of the scene, which can only be found from the geometry-based rendering process. Meanwhile, modern CPUs are more likely to have multimedia instructions, such as 3DNow! of AMD [2], MMX of Intel [22], and Sparc VIS of Sun [49]. In addition to cover the possible high amount of computation, this technical trend can also support the diversity of image-based rendering techniques because the CPU itself can handle the computing of rendering independently.

In brief, image-based rendering is a set of techniques that generate views from a group of reference images. Image-based rendering techniques are used in various areas and have a long history of over three decades. Compared to its geometry-based counterpart, image-based rendering is easy to model and fast to render. A more detailed discussion of the history and usage of image-based rendering can be found in [35]. In [36], McMillan compares the differences between geometry-based rendering and image-based rendering in detail.

2.3 Definition of Image-based Rendering

Image-based rendering techniques can be further classified into two categories: image-based modeling and image-based rendering. The target of image-based modeling is to recognize and understand the environment of the viewer by creating an approximate or accurate model of the environment. This category of techniques is often considered in computer vision. On the other hand, image-based rendering techniques generate views of a scene based on a model created through image-based modeling techniques. These techniques are more likely to be in computer graphic. According to some researchers [36], image-based rendering is an interface between computer vision and computer graphics. In this thesis, the term image-based rendering covers both the model creating phase and the view generating phase.

2.3.1 Plenoptic Function

Due to the wide usage and diversity of image-based rendering, no general model existed for it until McMillan and Bishop [35] linked this area with the Plenoptic Function. The Plenoptic Function was first introduced by Aelson and Bergen [1] to describe the model of human vision. It combines many early models of vision systems from different fields. The Plenoptic Function is a 7D function that describes the view of a scene at any viewpoint (v_x, v_y, v_z), in any direction (θ, ϕ), at any time t, and by every possible wavelength λ. Therefore, the Plenoptic Function can be written as $P = f(v_x, v_y, v_z, \theta, \phi, t, \lambda)$. Figure 2.1 shows the model of the Plenoptic Function. The goal of the Plenoptic Function is to describe incoming rays to a viewpoint. Recalling that an image is a 2D array of rays, images can also be described by the Plenoptic Function.

2.3.2 McMillan and Bishop's Definition

Based on the Plenoptic Function, it is possible to define image-based rendering in a formalized way. McMillan and Bishop [35] define image-based rendering as the following: *Given a set of discrete samples (complete or incomplete) from the plenoptic function, the goal of image-based rendering is to generate a continuous representation of that function.* In this definition, the sample of the Plenoptic Function is thought of as an environment map of a given scene. When an environment map covers the full spherical surface surrounding a 3D point (v_x, v_y, v_z), the sample is said to be complete; otherwise, it is an incomplete sample. This definition is general because it does not put any restrictions on

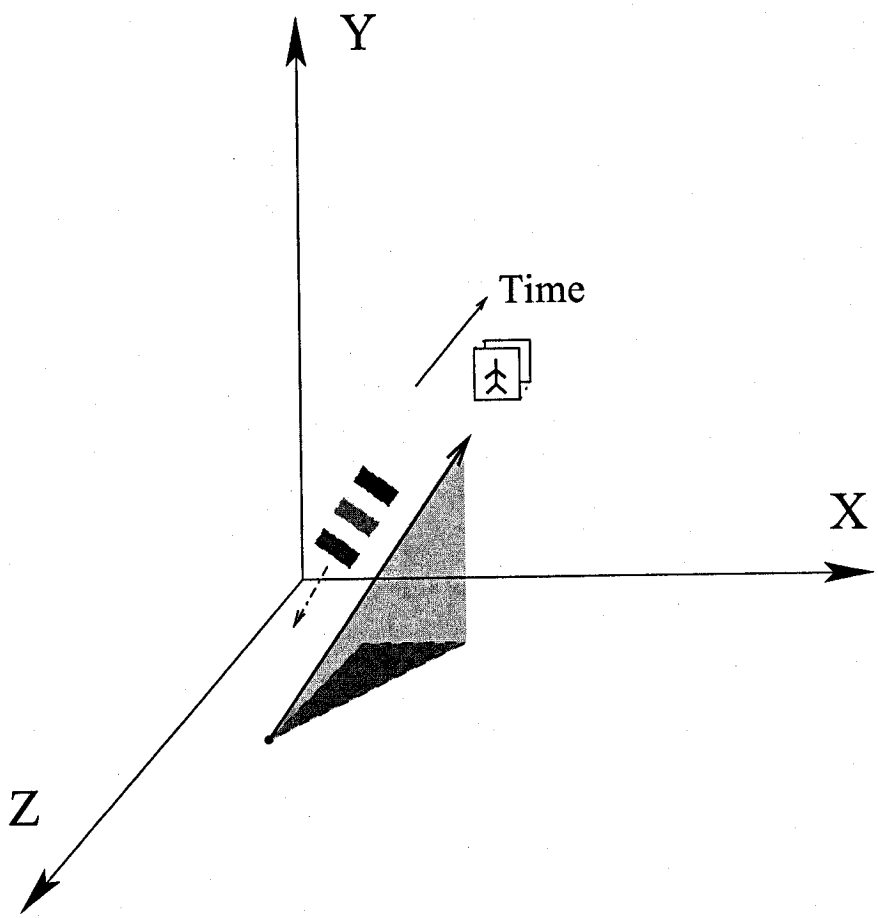

Figure 2.1: Model of The Plenoptic Function

the actual sampling and rendering method. Another advantage of this definition is that it links the rendering process to function sampling and reconstruction, which potentially enables the usage of many formal mathematical tools.

2.3.3 Zhang and Chen's Definition

Zhang and Chen [62] extend the definition of image-based rendering (IBR) as follow: *Given a continuous plenoptic function that describes a scene, IBR is a process of two stagessampling and rendering. In the sampling stage, samples are taken from the plenoptic function for representation and storage. In the rendering stage, the continuous plenoptic function is reconstructed with the captured samples.* It is clear that the major difference between this definition and that of McMillan and Bishop is that Zhang and Chen give more emphasis to the sampling phase of image-based rendering. Their definition clearly states that the sampling of the Plenoptic Function is part of the whole image-based rendering task. It is thus reasonable to include the sampling stage in image-based rendering process. Later analysis will show that there are many ways to sample the Plenoptic Function and that the view generating method changes with the specific sampling representation.

2.3.4 Interpolation Based Definition

Taking the complexity of the Plenoptic Function into consideration, it can be very difficult, if not impossible, to reconstruct even part of a continuous spectrum of the Plenoptic Function. For example, it is almost impossible to reconstruct a continuous spectrum along the wavelength axis of the Plenoptic Function on modern computer display devices. This is not only restricted by the ability of the display devices but also is determined by the human vision system [1]. When considering the situation along the time axis, video strips are natural samples of the Plenoptic Function. According to current television or video processing techniques, each video strip can be divided into a sequence of discrete frames. As a result, both the input and output of a rendering system are discrete frames.

The reference images are taken by cameras or rendered based on geometric models; therefore, there are some parameters of the real or virtual camera that determine the properties of the reference images. Some of the possible parameters are the following: the resolution of the image, the internal parameters of the camera (such as focus length, field of view, and non-linear distortion), and the external parameters of the camera (such the position and orientation) [64]. The similar parameters may also appear in

the virtual projection phase of an image-based rendering process. Therefore, both the input samples and output results may have certain properties determined by certain parameters. We will discuss in later sections that there is a certain type of image-based rendering techniques that generate rendering results with different properties than the input samples.

Based on the concern about the complexity of the Plenoptic Function and the preciseness of its samples, we define image-based rendering as the following: *Image-based rendering has two stages: sampling and rendering. In the sampling stage, discrete samples with definite preciseness are taken from the Plenoptic Function and described in a special format. In the rendering stage, an arbitrary discrete sample with certain projection parameters of the Plenoptic Function is generated based on the samples taken from the previous stage* [8].

The main differences between our definition of image-based rendering and previous ones in literature are the following. First, the goal of image-based rendering is to create a discrete sample of the Plenoptic Function instead of a continuous representation of that function. We have shown that it is impossible to reconstruct a continuous representation along the wavelength axis of the Plenoptic Function. With the same consideration, it is clear that creation of continuous representations along other axes of the Plenoptic Function is impossible too, due to the limitations of devices and the human vision systems. Second, we emphasize the preciseness of the samples and the rendering result. We will show in later sections that there are some image-based rendering techniques used to generate images with a different preciseness than the reference images.

From the definition of image-based rendering, we can easily find the similarity between image-based rendering and function interpolation. Interpolation is a method of constructing new data points from a discrete set of known data points. In a more formal way, given an arbitrary value of the variable(s) of a function, interpolation means to find the function value based on a number of samples of this function [48]. Recalling the definition of image-based rendering, the Plenoptic Function supplies views to the viewer at any time, through any wavelength, from any direction, and on any position. When doing image-based rendering, we want to find a view at a specific combination of time, wavelength, direction, and position based on the views taken at other combinations of the variables of the Plenoptic Function.

When considering interpolation in a linear space, we can express interpolation as $f(x) = \sum W_{n \in N}(x,\ n) f(n)$ [18]. At the left hand side of this equation, x represents an arbitrary point in the domain of the function and $f(x)$ is the value of the function at

arbitrarily given variable(s). At the right hand side of the equation, N is the set of points of the variable(s) of the function f and n represents any point in N. The samples used in the interpolation process are the function values at the point n and are represented as $f(n)$. In order to find the interpolated value, all samples are weighted and summed. The weights of the samples are expressed as $W_{n \in N}(x,\ n)$. It should be noted that the weights are evaluated in the set of N with respect to x.

In terms of image-based rendering, x is the specific combination of variables of the Plenoptic Function for which we want to generate the view $f(x)$. The reference images are represented by $f(n)$ where n is a special combination of the variables of the Plenoptic Function at which the corresponding reference image is taken. All reference images in the sampling set N are weighted by the weighting function $W_{n \in N}(x,\ n)$ with respect to x. In practice, the sampling and interpolation of image-based rendering are often done at the ray level. Considering images as 2D arrays of rays, the definition of image-based rendering and all the discussions of the relationship between image-based rendering and interpolation will remain valid. Using rays as the manipulation objects in image-based rendering will increase the rendering quality of the result because of the smaller grind size. This is a common approach to improving quality or accuracy in various areas.

2.3.5 Summary

Image-based rendering is defined on the Plenoptic Function. The first definition describes image-based rendering as a function reconstruction process, while the second one adds the sampling stage. Our proposed definition states that the goal of image-based rendering is to generate a discrete sample of the Plenoptic Function instead of a continuous representation. By this definition, we link image-based rendering with interpolation and emphasize the properties of both the input and output sample. With this definition, more techniques can be attributed as image-based rendering and we can organize all related techniques more clearly.

2.4 Classification of Image-based Rendering

Due to the diversity of image-based rendering techniques, many researchers try to organize them in a systematic way. There are different approaches to arranging the area of image-based rendering and classifying the various algorithms developed during their long history. In this section, we will briefly discuss these survey efforts and present our

method of classifying this area.

2.4.1 Previous Classifications of Image-based Rendering

There are several surveys of image-based rendering techniques in literature. They use different methods to organize various image-based rendering techniques. A common concern of these surveys is the extension of the usage of scene geometry that determines the size of the data set used by a specific image-based rendering technique [23].

In [35], McMillan and Bishop give an early survey of different image-based rendering techniques at that time. They list the different techniques in their chronological sequence. Movie-maps [30], image morphing [3], view interpolation [15], and pixel correspondence [25] are mentioned in the order of their appearance.

In 1996, many envisaged image-based rendering techniques were introduced in SIGGRAPH'96 such as view morphing [42], light field [28], and lumigraph [19]. These techniques triggered many related research works and Kang [23] surveyed these new developments by the pixel indexing scheme. All image-based rendering techniques are divided into four basic categories, namely, non-physically based image mapping, mosaicking, interpolation from the dense samples, and geometrically-valid pixel reprojection. The non-physically based image mapping category covers the techniques that transform one image into another without any consideration of 3D geometric information. These techniques are often used in the entertainment or advertising industry to create images with special vision effects. The mosaicking techniques are often used in the processing of aerial or satellite pictures so that a picture with higher resolution or a larger field of view can be achieved by combining more than one reference image. As the category name implies, the techniques in the third category synthesize an image based on multiple reference images of the same object or scene. In the last category, extensive geometric information about the pixels is used to reproject the pixels based on the parameters of a virtual camera. The geometric information can be recovered by using computer vision techniques or given as part of a scene description.

It is often noticed that image-based rendering techniques use various geometric information about a scene as an auxiliary. Actually, image-based rendering techniques cover a wide range of combined uses of images and geometry. Theoretically, if a detailed model of the scene or object can be retrieved from the reference images by some technologies, we can feed this model into a traditional geometry-based rendering pipeline to generate views. Despite the reality of this approach, it stands at an extreme of the path along

which different combinations of images and geometry appear. At the other end of the path, there should be techniques that are purely based on images which do not require scene geometry at all. A detailed discussion of different possible combinations of image and geometry can be found in [36].

The extent of the usage of geometry in image-based rendering is used by many researchers to classify different image-based rendering techniques. Shum and colleagues [45] organize different image-based rendering techniques into three distinct but not absolute discrete categories: rendering with no geometry, rendering with implicit geometry, and rendering with explicit geometry. In this approach to classification, implicit geometry means that 3D information is not given directly with the reference images; however, positional correspondences across images are given, and 3D information will be computed through these correspondences during the rendering process. On the other hand, the 3D information is given directly in the form of pixel depth values or 3D coordinates in the third category.

Oliveira [40] also states that image-based rendering techniques can be divided into two categories: pure and hybrid image-based rendering. The meaning of pure image-based rendering is straightforward, while various geometric descriptions, such as pixel depth and polygon meshes, are used by the techniques in the hybrid category. Zhang [63] also divides image-based rendering techniques into four categories: mosaicking, CAD-like modeling, correspondence-based rendering, and illumination modeling. The last three categories require scene geometry expressed as domain knowledge, pixel correspondences, and fixed illumination models.

Zhang and Chen [62] divide the image-based rendering techniques into two distinct categories: restraining the viewing space and introducing source descriptions. In the first category, different assumptions about the Plenoptic Function are introduced and scene geometry is not used. The authors discuss how these assumptions help to reduce the dimensions of the Plenoptic Function so that its sampling process of the Plenoptic Function can be simplified. As the price of simplifying the sampling process, the freedom of user movement is often limited. The second category covers the techniques that use geometric scene descriptions. In addition to direct geometric descriptions such as pixel correspondences, pixel depth, and mesh or volumetric models, texture maps and reflection models are also mentioned as auxiliaries of geometric description. The views are often rendered by the reprojection operations based on the given geometry.

2.4.2 Hierarchical Classification of Image-based Rendering

In this thesis, we propose a hierarchical classification method for image-based rendering based on the interpolation model of the Plenoptic Function. All of the seven dimensions of the Plenoptic Function can be divided into three classes: the chronological, the optical, and the geometric class. The time axis is in the chronological class; the wavelength axis is in the optical class; and, all the other five axis are in the geometric class. When interpolating the Plenoptic Function, the operations within certain class have no effect on the other class. However, different interpolation operations conducted within the same class may interfere each other. This is specifically valid for the geometric class that is the only class covering more than one dimensions of the Plenoptic Function. A similar assertion of the addibility of different assumptions of the Plenoptic Function is also discussed in [62].

Because of the independence of the different classes, the interpolation operations of the Plenoptic Function can also be classified into three independent categories at the first level: the chronological, the optical, and the geometric interpolations.

Chronological Interpolation

Chronological interpolations of the Plenoptic Function are often used in video processing. The samples used by this category of interpolations are the frames in the video sequence. In this section, two sub-categories of chronological interpolations are briefly introduced: motion compensation and frame cross-dissolve.

Various motion compensation technologies are used to remove the redundancy between consecutive frames in order to compress a video sequence. A well known usage of these technologies is in the MPEG-x video compression standards. As an add-on to DCT compression at the frame level, motion compensation techniques can provide a higher compression ratio in MPEG-x [50]. The frame group mechanism of MPEG-x standards is a good illustration of the chronological interpolation. In a group of frames, the first frame is often coded without motion compensation and is called I-frame (intra-coded-frame). Other frames in a frame group are called P-frames; a P-frame is predicted from the I-frame or the P-frame coming immediately before it. It is also possible to predict a frame from both chronological directions by introducing a B-frame into a frame group. A B-frame is predicted from the frames coming immediately before and after it. In terms of interpolation, the I-frame is the primary sample and the other frames are interpolated from this sample. The interpolated results can also be used as samples for future inter-

polation operations, such as predicting a P-frame or a B-frame using other P-frames and B-frames. In order to get a higher compression ratio and better interpolation quality, samples at the sub-frame level can be used in motion compensation, resulting in different motion compensation techniques such as global motion compensation, block motion compensation, variable block-size motion compensation, and overlapped block motion compensation. These techniques can also be explained by the interpolation model of image-based rendering with examples at a smaller grind level.

Frame cross-dissolve is a widely used technique in the entertainment industry. Given a starting frame and an ending frame, all intermediate frames are interpolated based on the distance from the two reference frames. Different from motion compensation, the intensity of each pixel in an interpolated frame is the simple weighted summation of the corresponding pixels in the reference frames. There is no vector description of the motion of blocks of pixels in frame cross-dissolve techniques. The weights of the interpolation operations can be designated manually or computed based on certain parameters. Frame cross-dissolve can often be seen in films to show the changing of illumination effects in a scene.

Optical Interpolation

The optical interpolation of the Plenoptic Function expresses various colours through a limited number of discrete colour channels. This interpolation mechanism is based on the character of the human vision system [1]. All colours are interpolated via the three discrete channels of the human vision system: red, green, and blue. This phenomenon leads to the RGB colour expressions in computer graphics, image processing, and computer vision [54]. There are more discrete colour expressions today; however, all these expressions are linear transformations of the RGB expression. In terms of interpolation, the samples are the channels and the weighting factors of the samples are determined by the optic sciences.

Geometric Interpolation

In the remainder of this chapter, we will focus on the interpolation operations of the last class, namely, the geometric interpolation. This interpolation class is the most complicated class of the three. All state-of-the-art image-based rendering techniques discussed by previous surveys can be thought of as belonging to this class. The techniques of this class are used for different targets and with different representations. Thus, it is

possible to organize the techniques according to many different methods.

In previous surveys of image-based rendering, the extension of the usage of scene geometry is a major focus [23, 40, 45, 62]. Image-based rendering is considered to be a convergence of computer vision and computer graphics. In both areas, geometric models play an important role as the output or input [27]. Because geometric models are often thought more compact than image-based models, the more scene geometry is used, the smaller the size to be of the models achieved [24].

We organize various image-based rendering techniques in this class according to different interpolation methods of the Plenoptic Function. From our point of view, there are four categories of interpolation methods: morph, mosaic, tabulation, and reprojection. Each category of these interpolation methods has its own effects on the rendering result and involves specific scene representations. This classification method is theoretically consistent with the methods used in previous surveys in that each category uses certain combination of images and geometry. However, the proposed classification method pays more attention to the interpolation process and the effects of the methods, and it will thus be helpful for application designers when selecting image-based rendering techniques based on the specific requirements of their applications.

2.4.3 Summary

A hierarchical classification of image-based rendering is proposed in this section. At the first level, the classification is based on different physical principles of the dimensions of the Plenoptic Function. There are three categories at this level: chronological, optic, and geometric interpolations. The interpolation operations in the first and second categories are simple because each category only handles one dimension of the Plenoptic Function. Because of the complexity of geometric interpolation, this category is divided further into sub-categories.

In previous surveys of image-base rendering, the extension of the usage of scene geometry is the major concern and main criterion for classifying the image-based rendering techniques within the geometric interpolation category. In order to facilitate technique selection, we organize geometric interpolation techniques into four sub-categories based on their interpolation methods: morph, mosaic, tabulation, and reprojection. Table 2.1 shows the proposed hierarchical classification of image-based rendering techniques. In the remainder of this chapter, we will discuss each sub-category of geometric interpolation in detail.

First Level	Second Level	Examples
Chronological Interpolation		frame cross-dissolve
Optic Interpolation		colour channels
Geometric Interpolation	morph	image morphing
	mosaic	QuickTime VR
	tabulation	light field rendering
	reprojection	3D warping

Table 2.1: The Hierarchical Classification of Image-based Rendering

2.5 Morph

2.5.1 Introduction

Morph or morphing operation is an image processing technique. It uses 2D pixel transformations to change an image into a second one. The image-based rendering techniques using morph do not consider the correctness of 3D geometry. These methods are often used in the education and entertainment industries to create special visual effects.

As morph proceeds, the original image is gradually distorted toward the target image and faded out. At the same time, the target image is gradually distorted in reverse from its initial state and faded in. The fade-in and fade-out is trivial, while the distortion is the key problem. All 2D affine transformations can be used to accomplish a specified distortion. These transformations include translation, rotation, scale, shear, and combinations of the basic operations. A mapping function that relates the coordinates of the two images is defined for every pixel. Since the transformations are affine, they can be done by matrix operations on homogeneous coordinates. Therefore, the mapping function can be expressed as a first-order polynomial [58].

2.5.2 Mesh Based Transformations

In order to enhance sensitivity to local deformations, Wolberg [58] presents a 2D spline mesh mapping. Instead of defining the pixel coordinates globally, they are expressed as a union of local functions. In this manner, a 2D mesh is defined for each image to control the mapping process. Each grid node on the mesh in the destination image has a set of reference grid nodes in the initial image. The displacements between grid nodes can be calculated by some form of interpolation. The interpolated images are composed of local

mesh patches and each patch is influenced by nearby control nodes. A 2-pass algorithm with linear interpolation is a common implementation. In the first pass, only transform actions in the x-direction or y-direction are computed, while the left direction is handled in the second pass. This 2D mesh-based approach applies more local controls than the feature lines based approach; as a result, it allows animators less flexibility. Moreover, the 2-pass algorithm may cause failure when the image is significantly rotated.

2.5.3 Feature Based Transformations

The most famous technique of this sub-category is likely feature-based image morphing [3]. Given two reference images, the initial correspondences are built manually by animators. Different from pixel correspondences, the initial correspondences are expressed as directed line segments that are thought of as features that link the two reference images. When rendering an intermediate view, a pixel is transformed based on its relationship to the defined features. According to specific situations, the transformation of a pixel may be accomplished by translation, rotation, scale, and/or a combination of these basic operations.

The most simple case of pixel transformation is one pair of features. In this case, the transformation of the pixel is determined by two parameters. The first parameter is the relative position of the pixel with respect to the feature line segment, while the other is the distance between the pixel and the feature line segment. Figure 2.2 shows a transformation by rotation and scale. Feature line PQ in the initial image transforms into $P'Q'$ in the destination image; correspondingly, the pixel x in the initial image transforms into the pixel x' in the destination image.

When there are multiple pairs of feature lines defined, the situation becomes more complex. A weighting factor is given to each pixel with respect to each pair of feature lines. In [3], the weight is determined by the distance from the pixel to the feature line. Figure 2.3 shows an example of transforming a pixel based on two pairs of feature line segments.

Compared with mesh-based morphing, the feature-based image-morphing allows the animator more flexibility in feature definition. This property makes feature-based image-morphing very useful for creating video sequences with special visual effects. Theoretically, any two unrelated objects can be transformed into each other through a set of well defined features. However, there are two problems to this approach. The first is the lack of local control and the second is the running speed. Because the features are defined

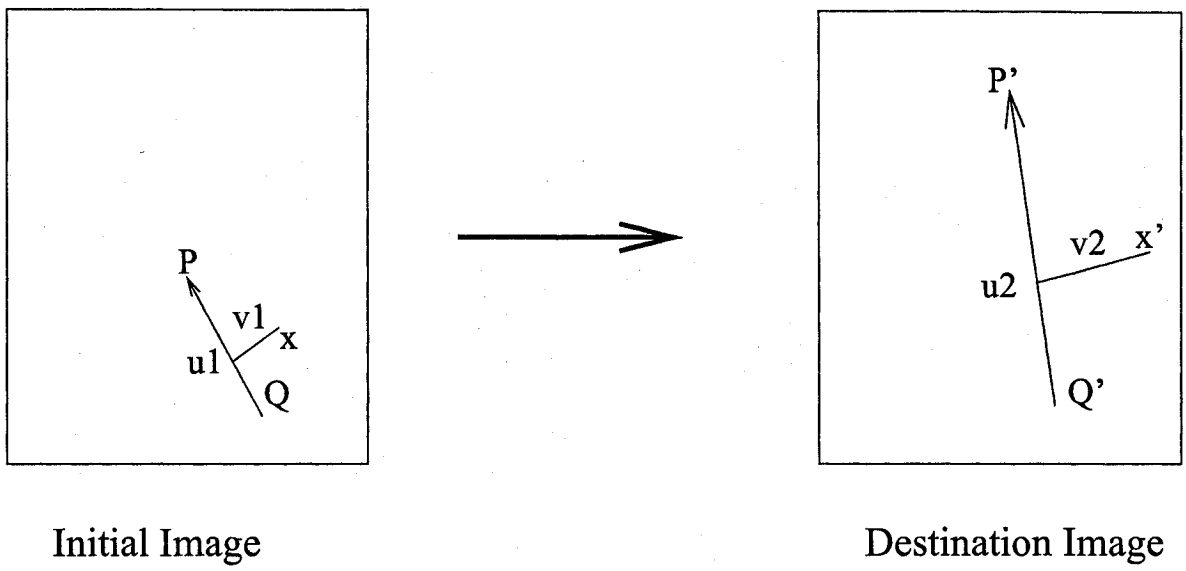

Figure 2.2: Example of One Pair of Feature Lines

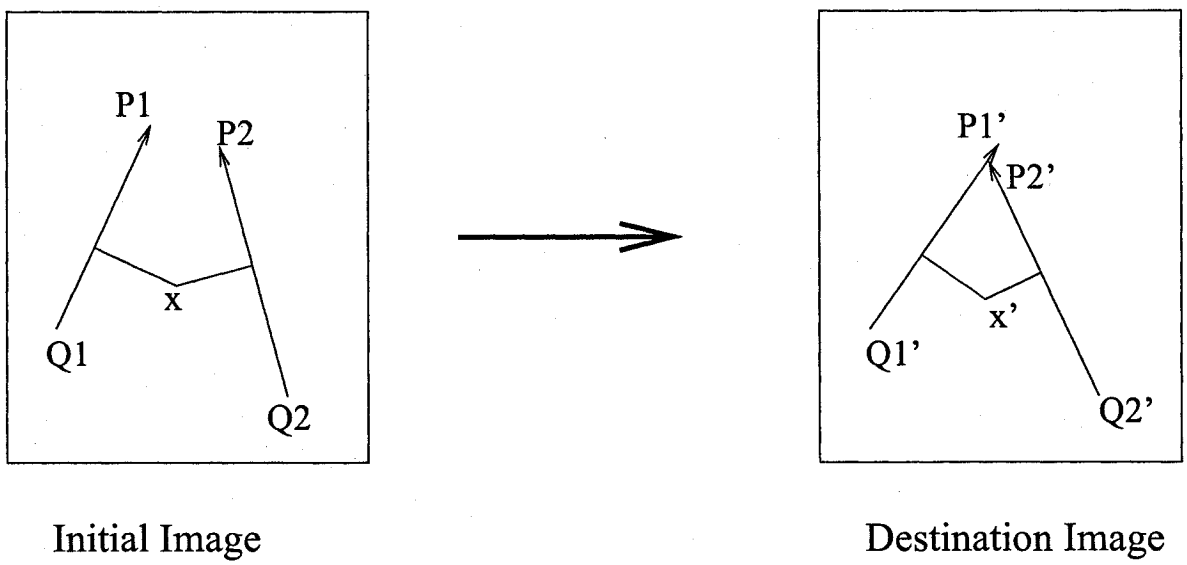

Figure 2.3: Example of Two Pairs of Feature Lines

globally, every pixel is influenced by all features, which may introduce artificial effects into the intermediate images and cause a waste of computing cycles. Thus, defining a group of well-fitted features is the key to the success of a morphing operation; however, this task does not only rely heavily on the talent and experience of the animator but also changes with the specific images used.

2.5.4 Summary

Morph is a special interpolation operation. When considering the interpolation model of image-based rendering at the image level, a sample of the Plenoptic Function is changed into another one without any external descriptions. However, at the ray level, each ray is transformed by a manually defined mapping function. Generally, the longer the morphing process, the more intermediate images are involved. Since the mapping function is defined manually by animators, the creation of the mapping function is a talent-intensive task. It is still very difficult to create a mapping function for a highly-smoothed morphing process, especially for longer ones. Morphing techniques are often seen in the movies to supply special visual effects. At the end of the film "Saving Private Ryan", a morphing process that changes the face of young Ryan to that of senior Ryan pulls the viewer back from fifty years ago.

2.6 Mosaic

2.6.1 Introduction

The goal of mosaic is to generate a larger image or an image with higher resolution by combining multiple images [23, 62]. The early stage of image mosaic involves creating a larger image from a group of aerial images. The mosaic has a larger field of view and is convenient for creating views within the whole area covered by the mosaic, especially for the boundaries of the reference images. There are two basic difficulties in image mosaicking: removing discontinuity and minimizing perspective distortion. Since the mosaic is generated by combining multiple reference images, discontinuity and perspective distortion can be introduced by many causes, for example, by camera position, device differences, and changing illumination effects. Moreover, when panoramic images or cylindrical panoramas are used in image mosaicking, perspective distortion is inherent because of the different projection modes of the reference images and the resulting

mosaic.

Removing Discontinuity

Although it is possible to create mosaics without removing visible discontinuities, there are a number of techniques for decreasing or removing the discontinuities. Milgram [37] presents a method for combining two images line by line so that the approximate optimal seam point for each line can be found and the introduced seam can be minimized. In order to handle clouds, snow, or ground water, Milgram [38] improves performance by choosing a best seam path with specified endpoints, and dynamic programming is used to reduce the computing cost. A brief discussion of the development of this area can be found in [47].

Minimizing Perspective Distortion

Generating mosaics for aerial or satellite pictures is fairly easy because of the small distortion in perspective. For nearby scenes, various techniques have been developed so that perspective distortion can be invisible. Generally, these techniques use reprojection or warping operations to generate the mosaic.

2.6.2 Larger Image

In spite of the discontinuities, creating an image mosaic from a group of aerial or satellite pictures is relatively easy because of the trivial distortions in perspective. However, many applications require image mosaics for nearby scenes. In this case, panoramic images or cylindrical panoramas are often used; therefore, noticeable perspective distortion is introduced and must be handled.

Hardware intensive systems also exist that can conveniently obtain a panorama [39]. However, the research focus in this area is still on how to construct cylindrical or spherical representations of a scene. Many techniques have been introduced to stitch multiple input references together, such as Quicktime VR [14] and plenoptic modeling [35]. In a usual case, the input images share the same center of projection. Thus, these images are related by 2D projection transforms. The transforms can be given or calculated. In order to calculate the 2D transform actions, a well-known technique is to designate four corresponding pixels for each image pair [20]; more recently, Szeliski and Shum [52] have used patch alignment to determine the transform action.

In practice, it is a common case that the source cameras of the multiple reference images do not share strictly the same centre of projection. Ghosting effects often happen in the resultant mosaic when there are more than one centre of projection. Several local alignment algorithms have been developed to eliminate such artifacts [52, 55].

Before conducting transformations on reference images, an image registration stage is required in the creation of an image mosaic. There are many methods for image registration, including the manual picking of corresponding points, automatic searching over a limited range, and phase correlation, etc. A good survey of this topic can be found in [65].

There is a notation that we should make at the end of this section. Although some techniques we mentioned in this section allow the user to move virtually in the scene [14, 35], we emphasized their ability to create images with a larger field of view than the reference images. This ability can be expressed as changing the properties of the samples, in terms of Plenoptic Function interpolation. The ability that allows the user to move virtually is discussed in later sections of this chapter.

2.6.3 Super-Resolution

The goal of the techniques in this sub-category is to enhance the spatial resolution of video sequences or still images by using multiple images of the same scene as the input. The similar, but not identical information in the multiple input images and given constraints enables the reconstruction of images with higher resolution. The techniques in this sub-category depend either on frequency domain principles or spatial domain principles.

The techniques that depend on frequency domain principles have theoretical simplicity and low computational complexity. It is easy to parallelize computing tasks based on these techniques. The techniques that depend on spatial domain principles, though computationally more expensive and more complex than their frequency domain counterparts, offer important advantages in terms of flexibility. A good survey of this area can be found in [5].

2.6.4 Summary

In this section, we discuss several techniques that generate images with different geometric properties than the input. In terms of the interpolation of the Plenoptic Function,

these interpolation methods create a discrete sample that has different geometric properties. The common feature of image morphing, discussed in 2.5, and image mosaic is the fixed viewpoint before and after the interpolation operation. Image morphing offers another image with different content at this viewpoint and image mosaic allows a viewer using another camera to view the same scene.

2.7 Tabulation

2.7.1 Introduction

As the category name implies, the image-based rendering techniques of this category have two stages. In the first stage, some form of lookup table is built up by taking many image samples of an object or a scene from many different camera viewpoints. Subsequently, the image associated with any given arbitrary viewpoint is synthesized by interpolating the samples in the table. The advantage of the methods in this category is that they allow the viewer to make virtual movements in a certain range of the scene. Because the rendering process is based on the interpolation of the lookup table, it is possible to render at a fast speed. However, these techniques often require a large data set and the calibration information of each camera.

2.7.2 Movie Map

An early approach in this sub-category is movie map [30]. This technique takes plain or panoramic pictures along the roads inside a small town and stores these reference images. The viewer is allowed to move along the predefined paths and turn at certain points along the road. When the viewer is moving, the closest reference image is selected and displayed for the viewer. The idea of this technique is straightforward and allows the virtual movement of viewers based on the scene description in images. In terms of Plenoptic Function interpolation, movie map uses the samples taken at a nearby position directly for the rendering position. This technique puts reference images into a one-dimensional table and extracts the image based on the distance between the rendering position of the viewer and the camera position. The idea behind this technique is used by many later approaches to geometric interpolation of the Plenoptic Function, such as [26, 29].

2.7.3 Mosaic Based

McMillan and Bishop [35] introduced the notion of plenoptic modeling with the five geometric dimensions of the complete Plenoptic Function. Because of the difficulty in spherical representation and implementation, a cylindrical panorama is used as the reference images in [35]. The novel views are generated by image mosaicking technologies discussed in Section 2.6. Geometric constraints are used to describe the relationship between the current rendering position and the input samples. The disparity distributions of multiple cylindrical panoramas can be found by stereo techniques. Figure 2.4 shows the relationship between the virtual rays of rendered cylindrical panorama V and the real rays of two input cylindrical panoramas A and B. This technique allows the viewer to move to any position inside the specified sampling area of the scene and to look in any horizontal direction at the rendering position. In order to offer a complete sample of the Plenoptic Function for a given viewpoint, spherical representation is introduced [52]. In practice, the viewer may move on a plane instead of the 3D space. This moving pattern reduces one geometric dimension and results in a manifold mosaic [41].

Another branch of the techniques in this sub-category is concentric mosaic [44]. Instead of taking cylindrical panoramas at different viewpoints, concentric mosaic samples the Plenoptic Function at a fixed viewpoint, but with a different radius. Each ray in the input cylindrical panorama is indexed by the radius, the angle, and the vertical position. Concentric mosaic is easy to capture. There is no difference between the capturing process of concentric mosaic and a traditional cylindrical panorama, except that a concentric mosaic requires more space. Vertical distortion often appears in the rendered image-based concentric mosaic; however, depth correction can alleviate this problem.

The techniques of this sub-category create tables for rays in the reference images based on three geometric properties: the projection centre of the cylindrical or spherical representation, the radius of the sampling surface, and the ray direction inside the representation. The advantage of the techniques in this sub-category is that they allow the viewer to have a full field of view in at least one dimension at any viewpoint. This advantage is achieved by an over-sampling of the Plenoptic Function, which creates a large data set and requires a complex image registration process.

2.7.4 Light Field and Lumigraph

Both light field rendering [28] and lumigraph [19] index the rays in the reference images by using two parallel planes. Each plane is expressed as a 2D mesh with discrete nodes.

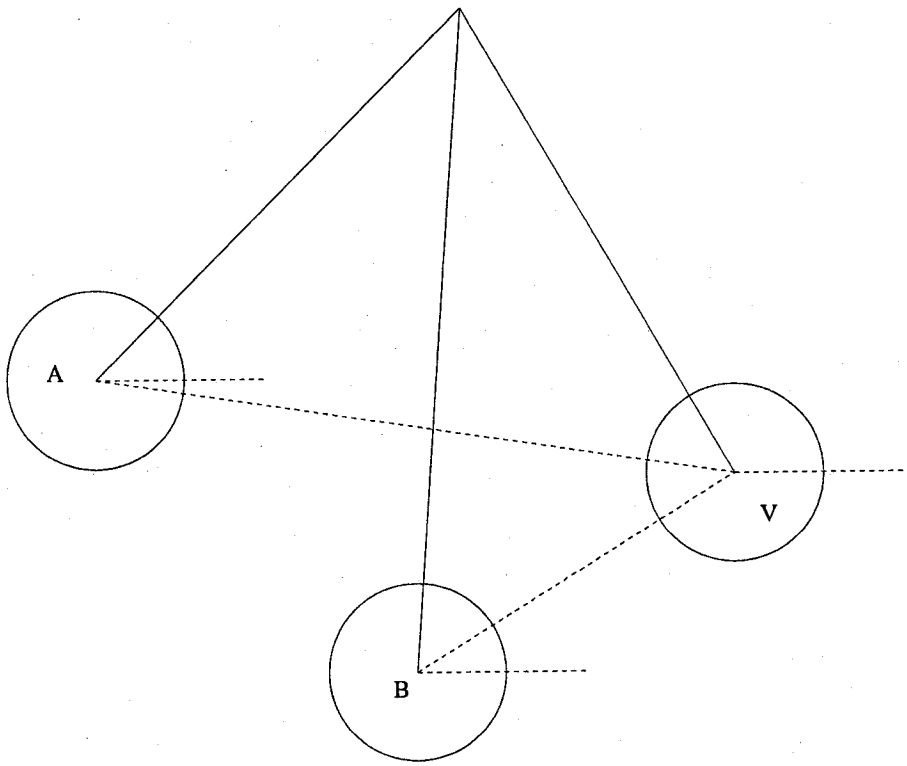

Figure 2.4: Cylindrical Interpolation of Plenoptic Modeling

Rays are indexed with the coordinates in the two parallel planes. A virtual ray is interpolated by the nearby sample rays. Figure 2.5 shows the ray index scheme of light field rendering and lumigraph. The difference between light field rendering and lumigraph is that lumigraph uses additional scene geometry to reduce the size of the data set [45, 62]. Recently, special hardware intensive systems have been developed to take samples for light field and lumigraph [57, 59].

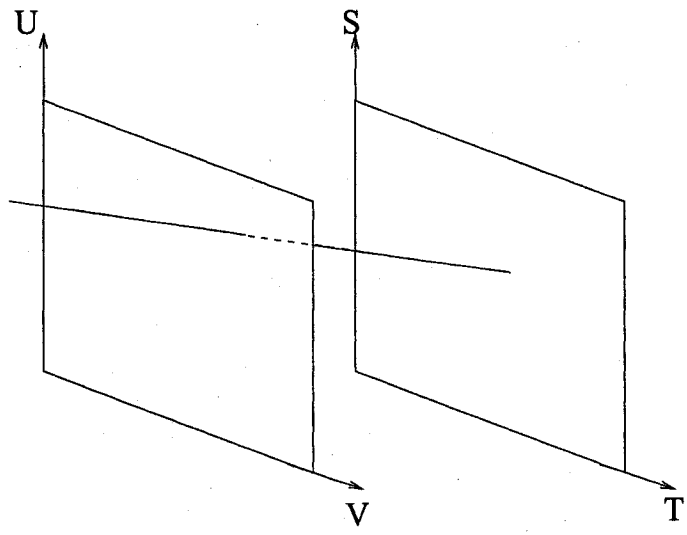

Figure 2.5: Ray Index Scheme of Light Field and Lumigraph

In order to simplify the sampling stage of light field rendering and lumigraph, Buehler and colleagues [10] introduce a rendering technique that does not restrict the planar camera array. On the other hand, a rough geometric proxy is used as the auxiliary to reference images. Zhang and Chen [61] develop this technique by using colour consistency checking and depth sweeping so that the geometric proxy can be avoided. Boukerche and Feng [8] improve the performance of this technique by using epipolar line search in the colour consistency check, which is introduced in detail in Chapter 5.

2.7.5 Summary

The techniques introduced in this section build look-up tables for the samples of the Plenoptic Function at the image or ray level. Various geometric parameters are used in the tables, including the distance between positions and angular difference. The interpolation process aims to find nearby geometric samples from the table and display

the sample or generate novel views based on the sample. These techniques allow viewers to move virtually in the scene. The range of the movement is determined by the sampling scheme.

2.8 Reprojection

2.8.1 Introduction

The techniques discussed in this section also allow the viewer to move virtually in a certain range of the scene. Compared with the techniques discussed in 2.7, these techniques use reference images with scene geometry. The novel views are rendered by reprojection operations. Because of the usage of scene geometry, the rendered result of these techniques can be geometrically accurate. Depending on the representation of scene geometry, the techniques in this category can be further divided into those using correspondences of reference images and those using scene geometry.

2.8.2 Using Correspondences

As the name implies, these techniques use positional correspondences across a small number of images to render novel views. These correspondences can be pixel correspondences or dense optic flows across different reference images. Novel views are interpolated by direct manipulation of these correspondences. Some techniques that fall under this class include view interpolation [15], view morphing [42], and dense pixel correspondence [25].

View Interpolation

View interpolation [15] is used to generate novel views of arbitrary viewpoints given two input images and a dense optical flow between them. This technique has a good performance when two input views are nearby so as to avoid visibility issues. Building a dense optical flow is a challenging task when utilizing this technique. If the scene is real, computer vision techniques can be employed to find correspondences across the reference images. For virtual scenes, the flow can be obtained from the given scene geometry. Because the optic flow is required for every pair of reference images, this technique is expensive for large scenes.

View Morphing

Unlike view interpolation, when given two reference images, view morphing [42] does not rely on dense optic flow. Instead, it restricts the viewpoints of novel views to be along the line linking two optical centers of the source cameras. Moreover, the view direction of the source and virtual cameras is perpendicular to the camera motion line. Another assumption is that the camera calibration information, including both external and internal parameters, is known. With these restrictions, the intermediate views are generated by reprojection operations using the linear interpolation of the projection matrices of the two source cameras.

Dense Pixel Correspondences

View interpolation and view morphing are special cases of dense correspondences. All these methods are characterized by the use of a relatively small number of images with the geometric constraints to reproject image pixels at a given rendering position. The correspondences across multiple images can be expressed as pixel correspondences, optic flow, and epipolar constraints.

Laveau and Faugeras [25] use a collection of reference images, weak correspondences between each pair of reference images, and the fundamental matrix between each pair to render novel views. Every pixel in the rendered view is interpolated from corresponding pixels in the reference images. A searching stage is required to find the corresponding pixels during the rendering. The constraints of epipolar geometry and image disparities are used to limit the searching range.

2.8.3 Using Scene Geometry

Using scene geometry allows a virtual camera to reproject the pixels with high accuracy. The most widely used scene geometry is the depth value of the pixels. The image with a depth value for each pixel is often called a depth map. For virtual scenes, the depth map can be obtained by enabling a Z-buffer when rendering. Range scanners can be used to generate a depth map for real scenes. 3D warping [34] first projects each pixel in the depth maps to its proper 3D position and reprojects the pixel to the rendered image using the virtual camera. Holes may appear in the rendered image of 3D warping. There are two causes that may result in holes. The first reason is the difference in resolution between the reference and rendered images, while the other is that the invisible part

of the reference image becomes visible in the rendered image. To fill in the holes, a common method is to map one pixel in the reference image to multiple pixels in the rendered image. Layered Depth Image (LDI) [43] is proposed by Shade and colleagues. In addition to storing what is visible in an image, LDI also keeps what is behind the visible surfaces. Therefore, the holes caused by visibility issues can be eliminated by LDI.

2.8.4 Summary

The techniques discussed in this section also allow the viewer to make virtual movements in a scene, which is similar to the techniques discussed in 2.7. The main difference between these techniques and those discussed in the previous section is that they use intensive geometric information about the scene. For synthetic scenes, it is easy to obtain various scene geometry from the geometric models that define the scene. However, it is still difficult to retrieve accurate geometric information from pictures of real scenes.

2.9 Conclusion

In this chapter, we define image-based rendering as an interpolation operation of the Plenoptic Function. Based on this definition, a hierarchical classification of image-based rendering is proposed. At the first level, all seven dimensions of the Plenoptic Function are divided into three classes. Chronological, optic, and geometric interpolation are introduced for each class at the first level. Due to the complexity of the geometric class, this class of interpolation operations is further divided into four categories: morph, mosaic, tabulation, and reprojection. Compared with previous surveys of image-based rendering, we focus more attention on the potential usage of each category of geometric interpolations. The proposed classification method is helpful for application designers in enabling them to choose specific image-based rendering technique based on certain requirements. In brief, Table 2.9 summarizes the suitable usages of each category of geometric interpolation operations.

Table 2.2: Suitable Usages of Image-based Rendering Techniques

Technique	Suitable Usage
Morph	Special visual effects
Mosaic	Larger field of view or higher resolution
Tabulation	Virtual movement in real and/or synthetic scenes
Reprojection	Virtual movement in synthetic scenes

Chapter 3

Local Buffer Management Policies

As an alternative way of rendering novel views of real and/or virtual environments to geometry-based models, 3D image-based rendering is getting more attention in recent years. Taking the weak hardware acceleration support of mobile devices into consideration, image-based rendering is a good choice for interactive 3D applications for such devices. However, because of the small storage size of mobile devices, the large size of the data set used by most modern image-based rendering techniques often introduces difficulties in utilizing image-based rendering on mobile devices. Therefore, a reference image buffer management mechanism is critical to the success of image-based rendering techniques on mobile devices. Different buffer management policies may result in different quality-of-service. In this chapter, we compare the performance of several buffer management policies based on customized QoS factors for image-based rendering techniques. In order to make our experiments general, we show that image-based rendering techniques are interpolations of the Plenoptic Function and define the QoS factors based on the interpolation theory. The results of the experiments show the impacts of different buffer management policies on customized QoS factors. Because the QoS factors do not depend on any special algorithms, this result keeps true for all modern image-based rendering algorithms.

[1]This chapter appears in the Proceedings of the 31st IEEE Conference on Local Computer Networks [7].

3.1 Introduction

Rendering novel views of 3D real and/or virtual environments in real-time is a key function of 3D interactive multimedia applications. With the increase of processing power and connection bandwidth of mobile devices, demands for 3D interactive multimedia applications have emerged from different areas, such as entertainment, distributed simulation, and e-learning.

Traditionally, a novel view of an environment is rendered based on the descriptions of selected geometric properties of the scene, such as position of vertices, reflection rate of surfaces, and characteristics of light sources. Because of the geometric nature, many geometric properties are expressed as matrices or vectors; and, this kind of expression leads to a large number of matrix operations during the rendering process. In order to render in real-time, 3D hardware accelerators are introduced. Presently, 3D hardware accelerators have become an integral part of our desktop PCs and workstations. However, compared with desktop PCs and workstations, the 3D hardware acceleration support for mobile devices is very weak. Mobile devices either do not have 3D hardware accelerators or are equipped with 3D hardware accelerators which have weak performance. The weakness of 3D hardware acceleration support on mobile devices makes real-time geometry-based rendering a challenging task on such devices.

Image-based rendering is an alternative method to render novel views of a scene. Different from geometry-based models, image-based rendering uses reference images as scene descriptions and produces the novel view by performing warping or interpolation operations on the reference images. Compared with geometry-based rendering, image-based rendering requires less CPU power and does not rely on hardware acceleration. The lightweight computing property makes it a good candidate rendering algorithm for 3D interactive multimedia applications on mobile devices. However, it is difficult to download to mobile devices the entire data set used by image-based rendering technique [23] due to the large size of the data set, the limited storage capability of the devices, and the large latency of wireless communication. Thereafter, reference image buffer management is critical to the success of image-based rendering on mobile devices.

In this chapter, we compare several buffer management policies for image-based rendering on mobile devices. These policies include FIFO, LRU, CLOCK, and Priority Queue. In order to measure the performance of different management policies, we define two QoS factors for image-based rendering algorithms: the order statistic of reference images and the number of reference image requests. These two QoS factors do not depend

on a particular image-based rendering algorithm; therefore, the result of our comparison remains true for all up-to-date image-based rendering algorithms.

This chapter is organized as follows: Section 3.2 briefly reviews previous work conducted on this topic. Buffer management policies and QoS factors are discussed in Section 3.3 and 3.4, respectively. Section 3.5 presents certain experimental results and is followed by a conclusion.

3.2 Previous Work

Two pieces of information are critical to the success of an interpolation process. One is the samples used during the interpolation process which are the bricks for building the interpolated value; the other one is the weights of these samples which determine how to interpolate. For the interpolation of the Plenoptic Function, different image-based rendering algorithms use different methods of sampling Plenoptic Function and weighing samples. For example, Seitz and Dyer [42] use distances between the centres of source cameras and the virtual camera as the weighing factor. Buehler et al. [10] interpolate virtual rays based on the angle difference between the sample rays and the virtual rays. Although it fully covers the 2D projected view, the Plenoptic Function, like other vision models, cannot retrieve the full geometry of a scene; i.e., a 3D point cannot be located solely through the Plenoptic Function due to the function's lack of depth information. Therefore, many image-based algorithms use additional geometric information for interpolation; various types of geometric information have been used explicitly and/or implicitly. The most widely used geometric descriptions are depth information of pixels [13, 53], camera calibration information [42, 61], and pixel correspondence between reference images [25].

In recent years, many studies have suggested approaches that utilize image-based rendering techniques for client/server-based interactive 3D applications. Lei et al. [26] describe a client-server system that allows a user to travel forward or backward along a predefined path, as well as change paths at certain points, virtually. Li et al. [29] use a sequence of images, panoramas, and concentric mosaics as scene representations. Users are allowed to move on a horizontal plane. Chang and Ger [13] and Thomas et al. [53] use depth maps to render novel views on wireless devices. A hybrid approach to render new views over the network based on both a geometric model and rendered images is described in [60]. Archeoguide integrates an artificial model of ancient architectures into current environments and allows users to travel virtually by using wireless devices [56].

In order to allow users to take different viewpoints and/or view directions, geographical information and GPS signals are used as scene geometry.

On the other hand, buffer management in multimedia communication over wireless networks also attracts the interest of many researchers. Various scheduling policies and their impacts on QoS have been discussed [6, 31]. However, the discussions in this area mainly focus on the general packet scheduling policy for multimedia communication on base stations of wireless networks. Nevertheless, the buffer management policy for image-based rendering on mobile devices is not widely discussed. Thomas et al. [53] briefly introduce a way of organizing cached images into two arrays in order to avoid having rendering blocked by communication. The reference images are pushed to the client by the server. As far as the authors know, there is no detailed discussion concerning the impact of different buffer management mechanisms on QoS of image-based rendering on mobile devices.

3.3 Buffer Management Policies

Reference images used by image-based rendering are cached in the buffer slots on the mobile devices. Different buffer management policies manipulate the buffer slots in different ways. Since image-based rendering is an interpolation process of the Plenoptic Function, a management scheme should keep the samples which contribute more to the final interpolation result in the local buffer to maintain a good interpolation quality. This is the general requirement for all buffer management policies. From the discussion of Section 2, we noticed that the actual image-based rendering algorithm often evaluates a reference image by the position or orientation difference between this image and the rendering position. The nearby reference images often contribute more to the final interpolation result than the distant ones. Therefore, the general requirement for buffer management policies can be expressed so as to keep the most nearby reference images in the local buffer.

Every actual image-based rendering algorithm has its own method to use the reference images to create the novel view and the actual rendering algorithm requires reference images based on its own mechanism. To make our comparison general, we will focus on the sample replacing policy of each buffer management mechanism when there is no free slot in the buffer. From our point of view, the sample replacing policy is critical to not only the rendering quality but also the communication overhead.

3.3.1 First-In-First-Out

FIFO is the most common and straight forward management policy and it is widely used in many applications. FIFO manipulates the buffer slots as a queue. When a new reference image arrives and there are free slots, the newly arrived reference image will be appended to the end of the queue. If there is no free slot when a new reference image arrives, a reference image is removed from the head of the queue to free a slot. Therefore, the removed reference image is the one with the longest residing time among all cached reference images. It is easy to implement FIFO policy and there is almost no overhead introduced by FIFO. Since FIFO manages the samples based on the sequence of their arrival time, it does not pay attention to the respective weight of the cached samples. When the reference images arrive in the same order as the occurrence of the corresponding requirements, FIFO can keep good locality of the samples because the new arrival samples are near to the rendering position. If the reference images arrive out of order, FIFO may discard samples with higher weight and these discard operations will decrease the QoS.

3.3.2 Least-Recently-Used

LRU keeps a timestamp for each buffer slot. This timestamp is updated for each access of the corresponding slot. It records the time of the last usage of the reference image cached in the corresponding buffer slot. When a reference image arrives and there are free slots in the local buffer, the newly arrived image is inserted into a free slot without special restriction. When a reference image arrives and there is no free slot in the buffer, the slot with the oldest timestamp will be freed for the newly arrived reference image. Compared with FIFO, LRU replaces the samples based on their last access time instead of their arrival time and LRU introduces an additional sorting action when removing a cached reference image. As we discussed in the beginning of this section, image-based rendering algorithms often give the nearby reference images higher weight than the distant ones; if the user moves continuously and in the same direction, LRU always discards the furthest cached reference image which guarantees the best rendering quality. However, if the user is allowed to take other movement instead of just moving forward, such as moving backward, rotation, or jumping, LRU may remove the sample which will be used in the interpolation process soon after. Therefore, LRU is not suitable for applications which allow the user to take complex movements.

3.3.3 Clock

Clock policy[46] manages the buffer slots as a circular queue. A global pointer is used to label the next free slot or the starting slot of searching action. Each slot is associated with a status indicator which has two values: KEEP and DISCARD. At the initial state, the global pointer points to any free slot in the local buffer. Because the slots are treated as a circular queue, any free slot can be selected for the first arrival of a reference image. When a free slot is allocated to a newly arrived reference image, the associated status indicator is set to be KEEP and the global pointer points to the successor of this slot in the circular queue. When the reference image stored in the slot is used by the last interpolation operation, the status indicator of the slot is set to be KEEP if it is not in KEEP state. When a reference image arrives and there is no free slot in the local buffer, a searching action will be taken. During the searching action, the global pointer moves forward in the circular queue. When the global pointer meets a slot whose status indicator is in the KEEP state, the status indicator of this slot is changed to the DISCARD state. When the global pointer meets a slot whose status indicator is in DISCARD state, this slot is used for the newly arrived reference image. The worst case of the searching action is to search the circular queue of buffer slots one round plus one step; in the first round, every slot changes its status from KEEP to DISCARD; and, the starting slot of the searching action is used for the newly arrived image in the first step of the second round. However, it is guaranteed that a slot can be found for the newly arrived image. In the searching action, the global pointer acts similar to the rotating hand of clock, giving this policy its name. This policy is also called second-chance replacement because the reference image is replaced the second time it is pointed to by the global pointer. Figure 3.1 shows a free slot searching operation of the clock policy. The slots coloured in light gray are labeled with the KEEP state and those coloured in white are in the DISCARD state or free. Before the searching operation, the slot currently pointed to by the global pointer is coloured in black. The slot with a brick-wall surface is the one used by the newly arrived reference image. The searching operation changes the state of the slots along the searching path from KEEP to DISCARD and uses the first slot in the DISCARD state or free to cache the image.

3.3.4 Priority Queue

As discussed in Section 2, image-based rendering is an interpolation process of the Plenoptic Function and it relies on the weight of the reference images to produce the novel view.

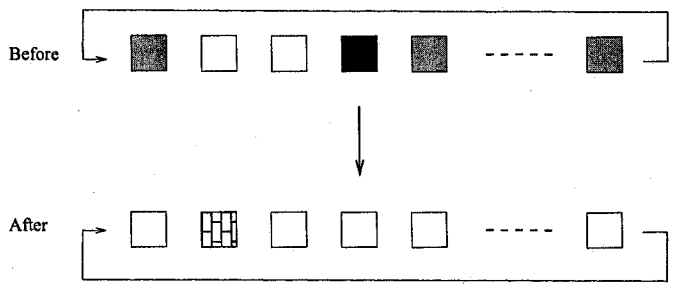

Figure 3.1: Clock Policy

It is natural to organize the buffer slots into a priority queue based on the weight of the reference images cached in them. When a new reference image arrives and there are free slots in the local buffer, the newly arrived image can be inserted into any free slot without restriction. When a reference image arrives after the buffer is full, the reference image with the lowest weight in the local buffer will be removed from the corresponding slot to make room for the newly arrived image. What should be noted here is that the weights of the reference images are calculated for the interpolation process and the buffer management just uses the calculated weights; therefore, there is no overhead for computing the weights. The overhead of this policy is the extracting operation from the priority queue. Priority queue policy has very good performance for continuous movements, such as translation, rotation, one-direction, and random two-direction. The only pitfall for this policy is jumping. Jumping will change the rendering position discretely; therefore, the weights of the reference images also change in a sudden way, compared to the status before jumping. This sudden change of weights may cause priority queue policy to discard a reference images which would be used in the near future.

3.4 QoS Factors

Different services have different measures of quality-of-service. In this section, we will define the QoS factors customized for 3D image-based rendering on mobile devices. Although there are many different image-based rendering algorithms, we try to make QoS factors independent of any particular algorithms. From our point of view, there are two aspects to how the QoS of image-based rendering should be measured. The first aspect is the communication process and the second aspect is the quality of the rendering result. In the remainder of this section, we will define two QoS factors to measure the two

aspects respectively and show the independence of these QoS factors. For the communication aspect, we select the number of reference image requests as the QoS factor so that we can avoid details in the communication process but manage the samples of the interpolation process at a higher level. For the rendering quality, we select the average order statistic of the reference images as the QoS factor; this measurement enables us to generalize the quality of the samples being used in the interpolation process for all modern image-based rendering algorithms.

3.4.1 Number of Reference Image Requests

A good buffer management policy should reduce the quantity of communication traffic between the server and client. In the case of image-based rendering, the volume of data varies with the algorithm and the compression method. In order to make our QoS factor independent to any special algorithm and compression method, we measure the communication traffic by the number of reference images transferred over the network rather than the volume of transferred data.

When the user is moving virtually in the scene, the weight of each reference image changes according to the virtual position and orientation of the user. To get a good rendering quality, all image-based rendering algorithms need to update the weights of the reference images, request reference images with higher weights from the server, and discard the images with lower weights in the local buffer. Due to the randomness of the movements of the user, it is difficult to precisely predict changes of the weights before the movements take place. Therefore, it is possible for any buffer management policy to discard an image with lower weight which may be used by the rendering process soon after. Such discard operations will make the rendering algorithm require the discarded image again from the server; and, this will increase the number of reference image requests and increase the actual data volume when using the same rendering algorithm and compression method. Thus, the performance of a buffer management policy can be measured by the number of the reference image request operations; a better buffer management policy can reduce this number by avoiding re-transmissions of certain cached images. From the perspective of the users, reducing the communication traffic can improve the interaction between the user and the application because of the shorter communication latency. Therefore, the reference image request quantity is a good measurement of the QoS that can be offered to the end user. Moreover, it depends only on the interpolation theory; therefore, it can be used with any existing rendering algorithm.

3.4.2 Average Order Statistics

The most direct way of evaluating the result of an image-based rendering operation is to take a picture at the rendering position and compare this picture with the produced novel view to check the similarity. The image comparison procedure involved in this method is a subjective process and is difficult to implement automatically [51].

Another way to evaluate the rendering result is based on the interpolation theory. We assume that the interpolation operation can get a result with good quality if it uses a set of samples with good quality. Obviously, the evaluation of the quality of the samples also involves image quality evaluation which, as we mentioned, is difficult to automate. Therefore, we use the weight of every reference image during the interpolation as an alternative measure of the actual quality of the image. Because the rendering result is a weighted summation of the samples, using a set of samples with higher weight will produce a more accurate result; in other words, the rendering result will be better if it is produced by using reference images with higher weights.

Although using the weight of the reference images as quality measurement enables us to avoid direct evaluation of image quality, the various kinds of weights used by different image-based rendering algorithms [62] make it algorithm-dependent. In order to make our QoS factor algorithm-independent, we use average order statistics [16] of the used reference images as the QoS factor. All reference images can be viewed as a priority queue, using the weight as the priority key. Thereafter, the rendering algorithm should extract the N reference images with highest weights from the local buffer to produce the novel view if it uses N images to render. Each of these selected images has an order statistics in the priority queue of reference images. The average of these order statistics shows the relative quality factor of the cached samples and can be used as a performance measurement of the buffer management policy. Because the average of order statistic does not rely on any weighting mechanism, it is algorithm-independent. The testing result based on the average of order statistic remains valid for all image-based rendering algorithms based on a remote data set.

What should be noted here is that the average of order statistic will always get the ideal value if it is calculated after the reference image request operation. The cause of this scenario is that the ideal reference images can always be downloaded to the local buffer if communication error is not considered. In order to correctly compare the performance of the buffer management policies, we calculate the average of order statistic before the image request operations in our experiments. In real life, the average of order statistic

calculated before the image requiring operations can be used to determine whether to render a novel view based on cached reference images to shorten the response time or to wait for the ideal reference images downloaded to the local buffer.

3.5 Experimental Results

In this section, we compare the performance of the buffer management policies mentioned in Section 3 based on the QoS factors defined in Section 4. In the experiments, the user makes single direction movement from the starting point to the ending point along a pre-defined path. When the user reaches the ending point, he or she turns back and goes to the starting point. The reference images are taken by cameras along the defined path and labeled with their positions, respectively. These cameras point to the direction perpendicular to the defined path that is also the view direction of the user. A novel view is rendered after each step of the user. With regard to the simple camera configuration, only the distance between the camera centre and the virtual position needs to be considered when calculating the weight of a reference image. In our experiments, the inverse of the distance between the camera centre of a reference image and the user's current position is used as the weight in the interpolation process.

It is noticed that there are other factors having effects on the performance of the buffer management policies. In our experiments, the effects of two factors on the performance of the different buffer managing policies are examined. The first factor is the ratio between the step length of the user and the distance between neighbouring cameras. For this factor, we define three scenarios. The step length is smaller than, equal to, and greater than the distance between the neighbouring cameras in each scenario, respectively. The second factor is the ratio between the number of reference images used for rendering a novel view and the size of the local buffer. We also define three scenarios for this factor. The number of the slots in the local buffer is twice, four times, and eight times the number of reference images used for rendering.

3.5.1 Number of Reference Image Requests

Table 3.1 shows the number of reference image requests made by a user during a 300-step movement. The ratio between the step length of the user and the distance of neighbouring cameras is fixed as 0.5 during these experiments. The number of buffer slot is twice, four times, and eight times the number of reference images used for rendering in the second,

Table 3.1: Reference Image Requirements

Policy	Case 1	Case 2	Case 3
FIFO	145	137	121
CLOCK	144	137	121
LRU	143	135	119
PQ	143	135	119

Table 3.2: Reference Image Requirements

Ratio between Steplength and Camera Distance	Case 1	Case 2	Case 3
0.5	145	137	121
1.0	279	259	219
2.0	528	484	400

third, and fourth columns, respectively. We assume that each rendering operation uses two reference images. Investigating the experimental results of using other ratio values, we can get a similar table with only scalar differences.

Table 3.2 shows the number of reference image requests made by a user during a 300-step movement, using FIFO as the buffer management policy. The ratio between the user step length and the camera distance changes from 0.5 to 2. The number of buffer slots is twice, four times, and eight times the number of reference images used for rendering in the second, third, and fourth columns, respectively. We assume that each rendering operation uses two reference images. The results shown in Table 3.2 remain valid for other policies.

From the above two tables, we can draw the following conclusions. First, compared with the ratio between user step length and camera distance, the ratio between the buffer size and the number of reference images used for rendering has a minor effects on the number of reference image requests. The difference between the data of each column in the first row in Table 3.1 is much smaller than the difference between the data of each row in a column. Therefore, it is better to reduce the user step length when the distance between cameras is fixed.

The second conclusion is that the number of reference image requirements is more sensitive to the ratio between the buffer size and the number of images used in rendering when user step length increases. From Table 3.2, it is clear that the difference between the image request number using different buffer sizes is smaller when user step length is shorter. Thus, short user step length is a better when we want to keep communication traffic stable.

The third conclusion is that there is no large difference among different buffer managing policies. From the data shown in Table 3.1, although the performance of the Priority Queue policy and the LRU policy is slightly better than the performance of the FIFO policy and the CLOCK policy, the advantage is very small. The reason for this result is the simplicity of the movement. The one-dimensional continuous movement makes the reference images arrive and leave almost in the same sequence with different policies. Therefore, the communication traffic stays the same for different policies.

In summary, from the experimental results analyzed in this section, a small ratio between the user step length and the camera distance is a better choice than a large ratio when we want to reduce communication traffic and/or keep traffic volume stable. Moreover, different policies contribute little in reducing the traffic volume when the movement of the user is simple. Thereafter, when the virtual movement of the users of a specific application is simple and the communication overhead is the only concern of this application, any policy can be used.

From our point of view, when the user moves in a shorter step, the weights of the cameras change more continuously than when a larger step is used. The continuous changing of the camera weights results a better locality of the samples used in each interpolation operation. This locality minimizes the performance difference of the different policies. When the user makes complex movement which breaks the continuous changing style of the image weights, such locality will not exist and the performance difference of the different policy will be more obvious.

3.5.2 Average of Order Statistics

In the experiments, the local buffer always has more slots than the reference images used in each rendering operation. In order to accurately reflect the quality of the used images in the rendering process, we calculate the average of order statistics of these used images instead of all cached images in the ordered priority queue of all reference images. It should be noted that only the reference images used in the rendering are counted.

In order to investigate the effect of the ratio between user step length and camera distance, a group of experiments are conducted. Figure 3.2, Figure 3.3, and Figure 3.4 show three cases of the average of order statistics, changing with the movement process of the user. In all three cases, the ratio between the user step length and the camera distance is 0.5, 1, and 2 in case 1, 2, and 3, respectively. The ratio between the buffer size and the number of reference images used in rendering is 2. In these experiments, the clock policy is used in these experiments.

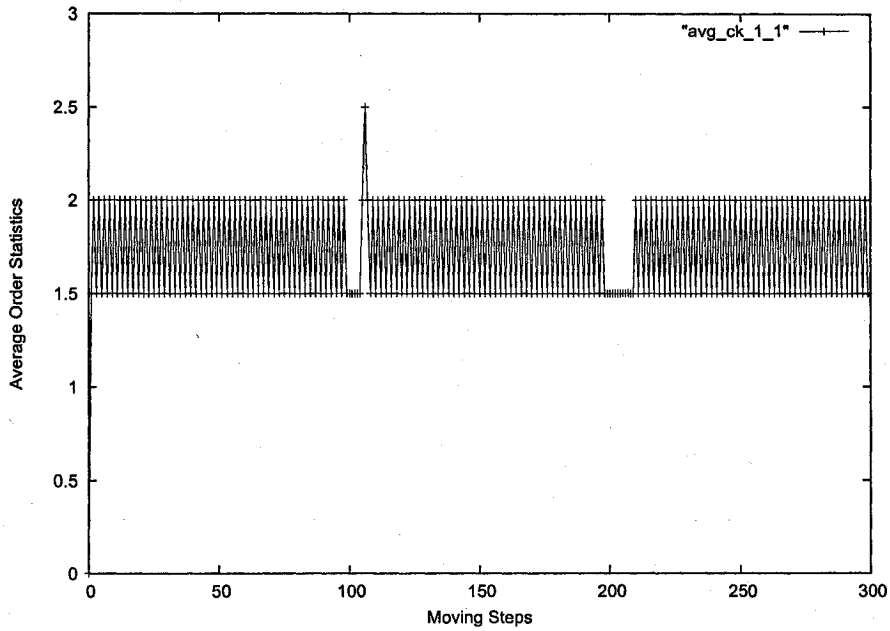

Figure 3.2: Average of Order Statistics, Case 1

The above figures show that the average of order statistics of the images used in rendering becomes least sensitive when the user step length is equal to the camera distance. However, Table 3.3 shows that smaller user step length can result in more stable average of order statistics. This table shows different statistics of the data shown in Figure 3.2, Figure 3.3, and Figure 3.4.

Based on the data shown in Table 3.3, a smaller user step length can result in the smallest expected value, the smallest variation of QoS, and the smallest max value of the changing average of order statistics. Recall the definition of the average of order statistics, the smaller value of this parameter reflects the better relative quality of the samples and the smaller variation range means the more stable quality. In other words, a

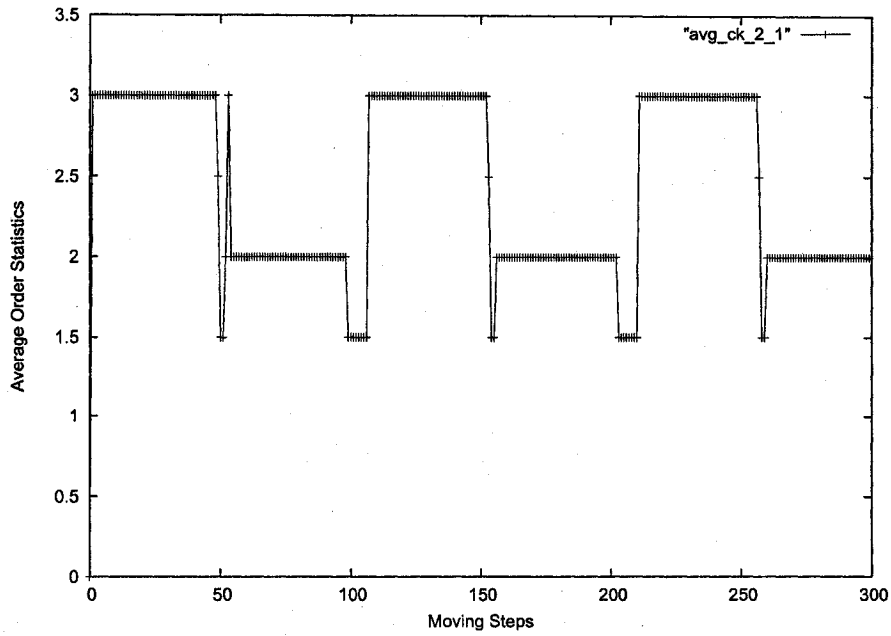

Figure 3.3: Average of Order Statistics, Case 2

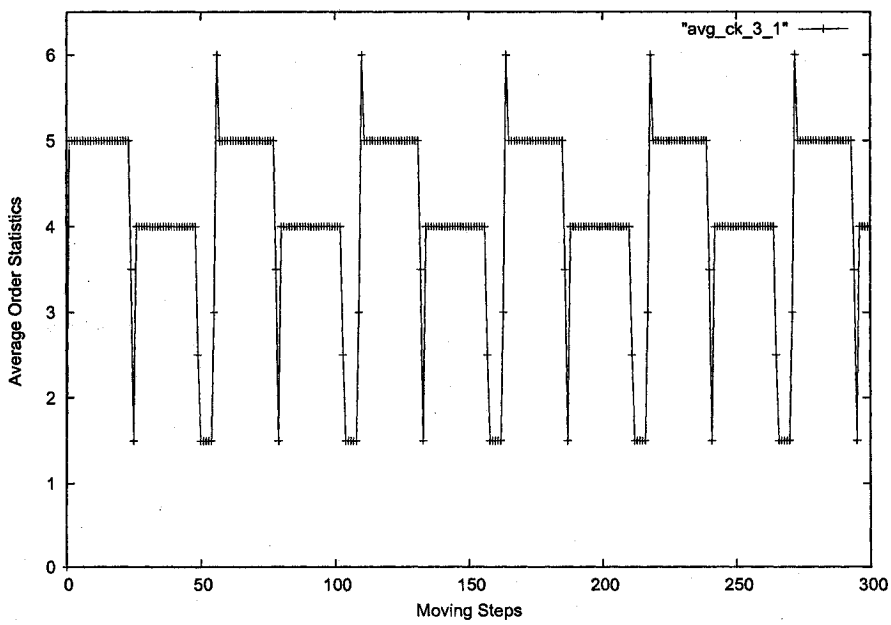

Figure 3.4: Average of Order Statistics, Case 3

Table 3.3: Average of Order Statistics with CLOCK Policy

Statistics	Case 1	Case 2	Case 3
max	2.5	3.0	6.0
mean	1.73	2.43	4.14
var	0.07	0.32	1.24

Table 3.4: Max Average Order Statistics

Policy	Case 1	Case 2	Case 3
FIFO	2.50	4.00	7.00
CLOCK	2.50	3.00	6.00
LRU	2.00	3.00	5.00
PQ	2.00	3.00	5.00

smaller user step length achieves better average quality, the smaller variation of quality, and better quality in the worst case.

In our investigation, we find that the effect of the ratio between the buffer size and the number of images used in the rendering process is small when the user step length is fixed. Therefore, a larger buffer size cannot offer major improvements in rendering quality, compared with user step length.

Considering effects of different buffer management policies, we find the Priority Queue policy and the LRU policy can achieve better performance in the worst case, especially when the user step length is large. Table 3.4 shows the max value of the average of order statistics of different policies. The ratio between the user step length and the camera distance is 0.5, 1.0, and 2.0 in each case, respectively.

When the user step length increases, there are more discrete changes in the weights of the reference images. Since the Priority Queue policy and the LRU policy explain the weight of the reference images better, they are more adaptable to the changing style. It should be noted that these discrete changes happen more frequently when the user is allowed to make complex movement. Therefore, the Priority Queue policy and the LRU policy are better than the FIFO policy and the CLOCK policy for applications that allow the user to move randomly.

3.6 Conclusion

3D image-based rendering is an emerging solution to interactive 3D applications on mobile devices. Although the lightweight computing property of image-based rendering techniques makes it attractive to mobile devices, the large data set used by these techniques lays barriers for utilizing them on mobile devices due to the limited storage capacity and the high wireless communication latency. In this paper, we discussed different buffer management policies that can be used by the image-based rendering algorithms and compared the performance of these policies based on customized QoS factors. In our experiments, other factors that effect QoS factors are also considered.

Based on our experimental results, the Priority Queue policy and the LRU policy are more suitable to the applications that allow random user movement. Moreover, the user step length has a more significant effect on the QoS factors than the buffer size. A shorter user step length can achieve less communication volume and better rendering quality at the same time.

Chapter 4

Local Buffer Management Implementation

In order to utilize 3D image-based rendering technology on mobile devices, an efficient local buffer management schema is required to cope with the large set of data used by image-based rendering algorithms, communication errors, communication latency, and virtual movements of the user. In this chapter, we present a robust mechanism for managing a local reference image buffer on mobile devices. Our proposed technique is generic because we do not put any restrictions on the actual rendering algorithm. Experiment results show that the proposed technique is robust toward communication errors. Based on the discussion of Chapter 3, we select priority queue as the local buffer management policy.

4.1 Introduction

In Chapter 2, we linked image-based rendering techniques with function interpolation theory and we discussed the impact of different local reference image buffer management policies in Chapter 3. In this chapter, we present a buffer management mechanism for running 3D image-based rendering on mobile devices. Our proposal is robust to communication errors and latency. In addition to dealing with communication uncertainties, our proposed mechanism optimizes the organization of cached reference images so that the most appropriate reference images available can be used by the rendering process.

[1]This chapter appears in the Proceedings of the The 4th ACM International Workshop on Mobility Management and Wireless Access. [9]

This chapter is organized as follows: the architecture and implementation details of our proposed technique are introduced in Sections 4.2 and 4.3, respectively. Section 4.5 shows certain experimental results and is followed by a Conclusion.

4.2 A Two-Level Buffer Model

Our proposed local reference image management architecture is based on the interpolation model of image-based rendering. It uses a two-level buffer to parallelize the rendering and communication sub-tasks. Reference images are required by the client based on their respective weight of interpolation. The local buffer is re-organized with the position change of the virtual viewpoint and the arrival of reference images.

Buffer areas are organized as buffer slots. One cached reference image, whether fully or partially arrived, will occupy one slot. The number of slots in each buffer area is determined by different factors. For the texture buffer, this number is determined by how many reference images are used for interpolating a novel view; it is a scene-determined parameter. On the other hand, the size of the communication buffer is determined by the storage and communication abilities of the device. The number of slots in each buffer area can be managed dynamically. The texture buffer holds the reference images for texture mapping during the rendering process. The communication buffer is used for the intermediate stages between the arrival of a reference image and its entrance into the texture buffer. Cached reference images will be exchanged between the texture buffer and the communication buffer based on their appropriateness for the current virtual rendering position. In order to avoid large memory area operations, a pointer array is employed during buffer slot swapping. Moreover, reference image meta data and its pointers are used to label the random status of reference images and buffer slots.

4.3 Implementation Details

Although there are many different image-based rendering techniques, the proposed two-level buffer management algorithm can theoretically support all of these techniques when the source data set is remotely located. The reason for this is that the management algorithm does not rely on the absolute weight of a sample, as this may change according to the technique used. However, the algorithm relies on the relative order of the reference images. Different interpolation algorithms use different sampling and weighing schema.

Nevertheless, for any interpolation algorithm, samples can be ranked in an order based on the algorithm's respective weighing mechanism. Although the ordered position of a sample is not directly useful for the interpolation process, it is valuable in terms of buffer management. For a buffer with limited size, it is ideal to hold a number of samples with a larger weight than others. If the samples are set in a certain order, whether ascending or descending based on their respective weight, it is possible to manage them in a buffer using their ordered position. The proposed two-level buffer management algorithm is based on an ordered array of the reference images; thus, it can support various image-based rendering algorithms that run remotely. In the remainder of Section 4, the implementation details of the proposed algorithm will be discussed.

4.3.1 Parallel Sub-tasks

The operations at the client side can be divided into three parallel sub-tasks: view rendering, buffer management, and reference image requiring. View rendering renders the current novel view by using the reference images cached in the texture buffer. Buffer management takes care of the contents of the local buffer so that the most appropriate reference images can be used for rendering or can be required if they are not inside the local buffer. Reference image requiring is controlled by the buffer management sub-task and requires reference images from the server or other peers. POSIX multi-thread is employed to process these tasks in parallel. Signals are used as a method of communication between the rendering and buffer management threads. The buffer management thread creates and cancels image-requiring threads.

4.3.2 Reference Image States

Due to the remoteness of the data set and the proposed two-level buffer architecture, a reference image may have several different states. Figure 4.1 shows the possible states of a reference image and the actions that can change its state.

The actions described in Figure 4.1 move a reference image from one state to another. Depending on their respective weights, certain images will be more appropriate than others for the current novel view during the rendering process. These images are referred to as "ideal images" in this work. The number of ideal images for the current novel view is the same as the number of slots in the texture buffer. It is ideal that the texture buffer should hold only the ideal images. Nevertheless, it is possible that non-ideal images can stay in the texture buffer because of the movement of the rendering position and

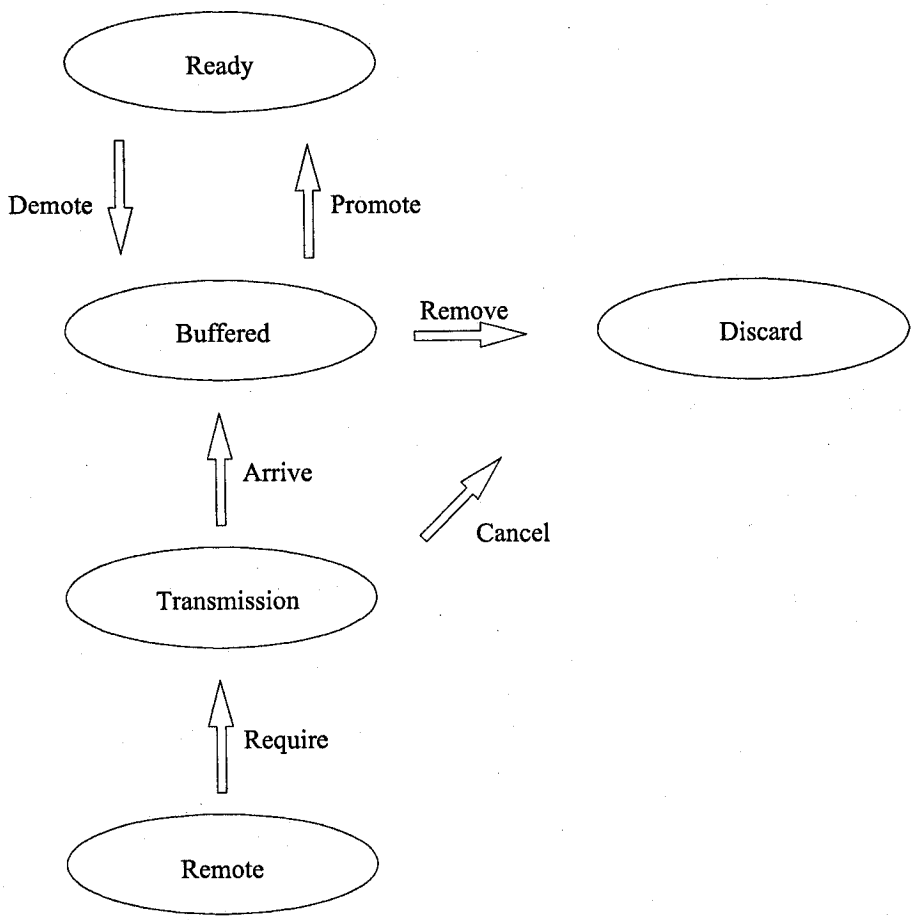

Figure 4.1: Reference Image Status

communication latency. Due to the possibility of having cases that are not ideal, the actions described in Figure 4.1 are designed to make sure that the rendering thread can use the most appropriate images available in the local buffer to render the novel view. Moreover, if an ideal image is not available in the local buffer, it can be required from the server. The following list explains the condition for each action in Figure 4.1.

1. Require: If a reference image is ideal for the current rendering position but is not in the client device's local buffer, and is not being required, this image will be required from the server.

2. Arrive: This action moves a reference image from the Transmission state to the Buffered state. This means that a reference image is successfully transferred from the server to the client.

3. Promote: This action moves a reference image from the communication buffer to the texture buffer. When a fully arrived reference image in the communication buffer is found to be more appropriate for the current view than any images in the texture buffer, this image should be promoted to the texture buffer for future use by the rendering thread.

4. Demote: This action moves a reference image from the texture buffer to the communication buffer. When a reference image in the texture buffer is less appropriate for the current view than any images in the communication buffer, this image should be demoted to the communication buffer. The promotion and demotion actions occur simultaneously when swapping two reference images.

5. Remove: This action removes a reference image from the communication buffer. A slot in the communication buffer is freed for future use if the reference image that is cached within it is removed. Any removal should take place in the communication buffer in order to keep texture buffer consistent.

6. Cancel: This action can be taken by the client device either actively or passively. When it is taken actively, a reference image requirement is canceled by the client device and the corresponding communication thread is killed by the buffer management thread. When cancellation happens passively, a reference image cannot be successfully transferred from the server to the client or its colour information cannot be correctly decoded. In either case, the respective communication buffer slot is freed for future use.

4.3.3 Data Structure

When the application runs, a reference image can be randomly placed in any state due to the changing rendering position. The goal of the data structure design is to map the random status of reference images with respect to their unique identifiers in the data set. In this paper, it is assumed that each reference image has sequential image ID as identification in the data set. All of the information about a reference image can be obtained through this image ID. Figure 4.2 shows the data structure design based on this assumption. In this data structure, two arrays are used to hold the related information of reference images and two arrays are used to hold pointers for other arrays. The following list shows the use of each array.

1. Buffer Slots Array: Each member of this array holds the colour information of a reference image. In order to identify the respective reference image, image ID should be kept. If there are other concerns, other image-related information can be added. The size of this array is determined by the storage ability of the device. It is not required that each reference image has a slot in this array. Instead, the slots are shared by all reference images.

2. Buffer Slot Pointers Array: This array has the same size as the buffer slots array. Each member of this array holds a pointer for a member of the buffer slots array. This array is used for two purposes. First, it avoids large memory area moving operations by swapping pointers. The second use of this array is to distinguish the texture and the communication buffers from one another. There is no difference between the buffer slots of the texture and communication buffers. The entire buffer is allocated as a continuous area of memory. The organization of different buffers is implemented through this array: if there are N slots in the texture buffer, the buffer slots pointed to by the first N pointers will be used as texture buffer slots; other pointers in this array point to communication buffer slots.

3. Meta Data Array: Each reference image in the data set of the current scene has a corresponding member in this array. This array should be organized in such a way that each reference image can be located without searching through the array. Members of this array do not hold the colour information of corresponding reference images but instead hold their meta data. Meta data describes how an image will be used during rendering. It includes the current weight of the image, its status, the geometric information for calculating its weight, and the index in the buffer slot

pointer array if the image is in the local buffer. The current weight and status of a reference image will change according to the movement of the rendering position and the communication process. In addition, the index of the buffer slot pointer array may change when the image moves between the texture and communication buffers. Thereafter, the related information in this array must be in order so that it can remain up-to-date. One of the considerations for using this array is locating which member of the array should be updated.

4. Meta Data Pointers Array: This array is of the same size as the meta data array. Each member of this array points to a member in the meta data array. The Meta Data Pointers Array is used to sort reference images into an suitable order based on their current weight. The sorting action affects only this array: after sorting, the i^{th} pointer points to the meta data of the reference image on the i^{th} position of the sorted array. This will keep the meta data array unchanged after each sorting action. Therefore, no difficulty will be involved in reference image information updating. Furthermore, the sorting results can be conveniently utilized for operations discussed in the remainder of this section.

4.3.4 Weighing Reference Images

The way in which reference images are weighed with respect to the current rendering position is based on the actual rendering algorithm. Any specific algorithm can define its appropriate geometric information as meta data of reference images and calculate the current weight of reference images based on this information. Since there are no restrictions on geometric information, any type of information can be defined as meta data. Therefore, any actual rendering algorithms can be used with our proposed buffer-managing algorithm. In the implementation of the experiments, an improved view morphing [8] is used as the actual rendering algorithm and reference images are weighed based on the distance between their camera centres and the rendering position.

4.3.5 Sorting Reference Images

The meta data pointer array is employed during sorting action to keep the organization of the Meta Data Array unchanged after sorting. There is no requirement concerning the initial state of the sorting action; therefore, the sorting action can take place fol-

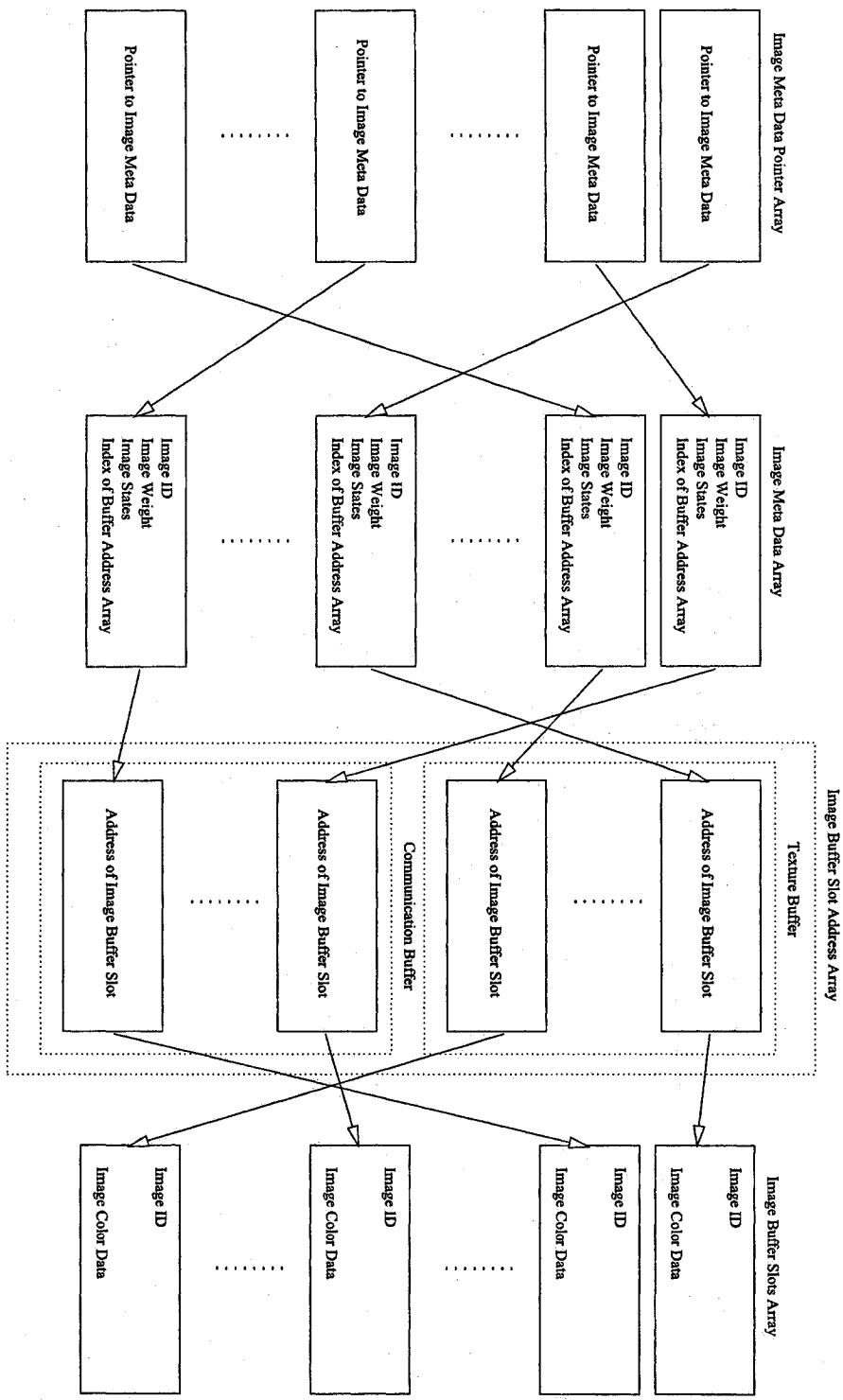

Figure 4.2: Data Structure

lowing another such action. Any suitable sorting algorithm can be employed. In the implementation of the experiments, counting sort is used.

4.3.6 Re-organizing the Local Buffer

Buffer re-organizing means optimizing the local buffer for the current virtual viewpoint by promotion and demotion. It may be triggered by two kinds of events: virtual viewpoint changing and reference image arrivals. This process is based on the sorting result of the reference images. In addition, buffer re-organizing requires the state of each reference image. The buffer re-organizing swaps the reference images between the texture and communication buffers if necessary so that the most appropriate reference images available in the local buffer can be used by the rendering thread. During buffer re-organizing operations, the pointers in the Buffer Slot Pointers Array are swapped to avoid large-memory-area swapping operations.

4.3.7 Updating the Current State of Reference Images

The current weight of a reference image changes with respect to the current rendering position. However, the reference image requiring operations and the communication process between the server and the client move a reference image among different states. This is shown in Figure 4.1. These two kinds of changes occur randomly. In order for information about the reference images to remain up-to-date, the Meta Data Array and the Buffer Slots Array are employed. Because the reference image sorting operation does not change the organization of the Meta Data Array, it is easy to locate the meta data record of a reference image in order to update the current weight of the respective reference image. The local buffer managing thread goes over the Buffer Slots Array to update the state changes caused by the reference image requiring operations and the communication process.

4.4 Experimental Results

In the implementation of our experiments, improved view morphing is selected as the actual rendering algorithm. By using this rendering algorithm, a novel view is interpolated using a minimum of two reference images. The data set of this improved view morphing algorithm contains plain images and camera calibration information about them. The

sampling process of the Plenoptic Function is complex. In order to concentrate on our proposed buffer management mechanism, the configuration of the cameras is made easy. The process is described as follows. All cameras have the same view direction and are positioned along a line perpendicular to the view direction. Users are allowed to move in the area behind the cameras. Figure 4.3 illustrates this camera configuration. When users move, they can take any one of the four directions shown in Figure 4.3. A user's step length is configurable for different experiments. The simulation of communication errors is implemented at a high level. For each experiment case, a failure probability of each reference image requiring operation is given as a parameter. When an experiment case runs, every reference image requiring operation will fail with the given rate. The configuration of local buffer is as follows. The number of slots in the texture buffer is two. This number does not change for different experiment cases. The size of the communication buffer is obtained by multiplying the size of the texture buffer by a configurable parameter that changes for different experiment cases.

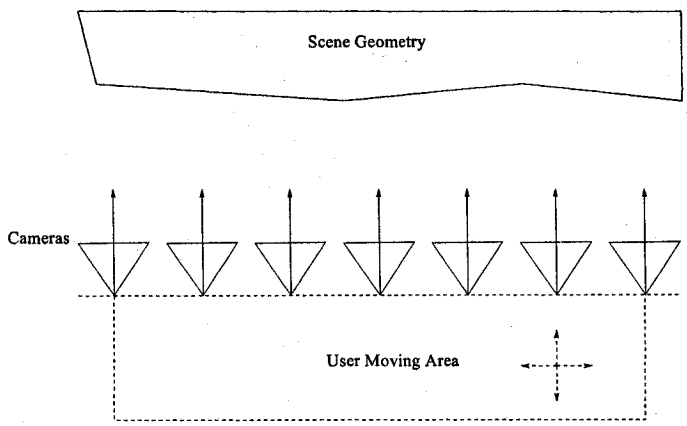

Figure 4.3: Camera Configuration for Experiments

The first aspect examined is the validity of our proposed buffer management algorithm. During these experiments, certain reference images of the scene are intentionally made unavailable to the client side so that novel views must be rendered with poorly appropriate reference images. This set of experiments is to check whether a novel view with an acceptable quality can be rendered by poorly appropriate reference images. Figure 4.4 shows a view rendered based on the most appropriate reference images; Figure 4.5 shows a view rendered with a number of poorly appropriate reference images. Although there are some artificial effects in Figure 4.5, the quality is still acceptable. Based on

our experiments, increasing the density of sampling may reduce the artificial effects in the rendering result.

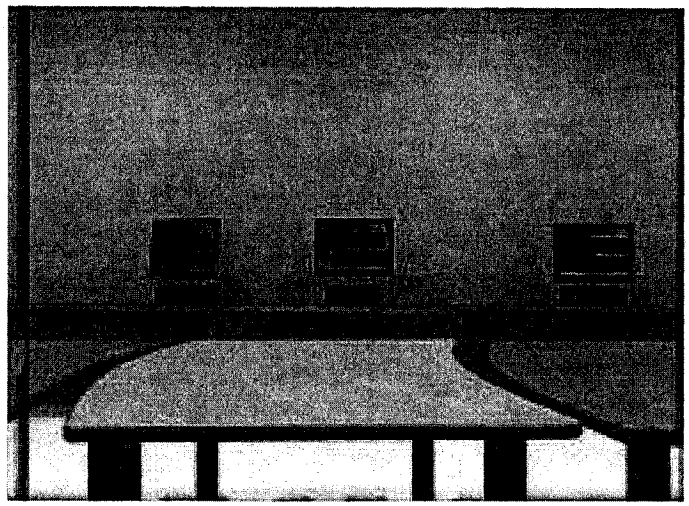

Figure 4.4: View Rendered with the Most Appropriate Reference Images

Figure 4.5: View Rendered with Poorly Appropriate Reference Images

The second aspect examined is the performance measured by the number of reference image requiring operations. In these experiments, we make the user move 1000 steps in the virtual environment and record the number of the reference image requiring operations. Different experiment cases are given different possibilities of failure in terms of

reference image requiring operations. Figure 4.6 shows the results. Although the number of reference image requiring operations increases with the rate of failure, the number of reference image requiring operations has a much slower pace. When the failure rate increases from 2% to 30%, this number only increases by approximately 3%.

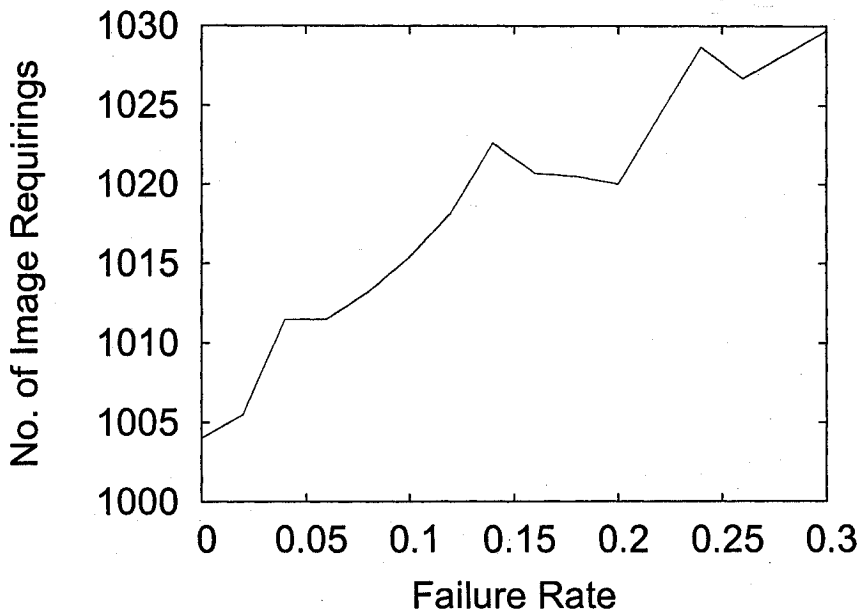

Figure 4.6: Reference Image Requirements vs. Failure Rate

The test cases in the third category examine the performance measured by the average value of the ordered position of the reference images in the texture buffer during 1000 steps of the user's movement. In the best cases, the measured performance should be 1.5 or 1.0. Recall that there are always two reference images in the texture buffer. If the average value of the ordered position has a value as 1.5, the nearest two reference images have different distances from the current rendering position; if the average value of the ordered position has a value as 1.0, these two images have the same distance from the current rendering position; otherwise, the novel view is rendered using reference images that are not optimal for the current rendering position. Figure 4.7 shows the change of the average value of the ordered position of the reference images in the texture buffer with the communication failure rate. Based on these results, it is obvious that novel views of acceptable quality can be obtained in the long run for the user's virtual traveling.

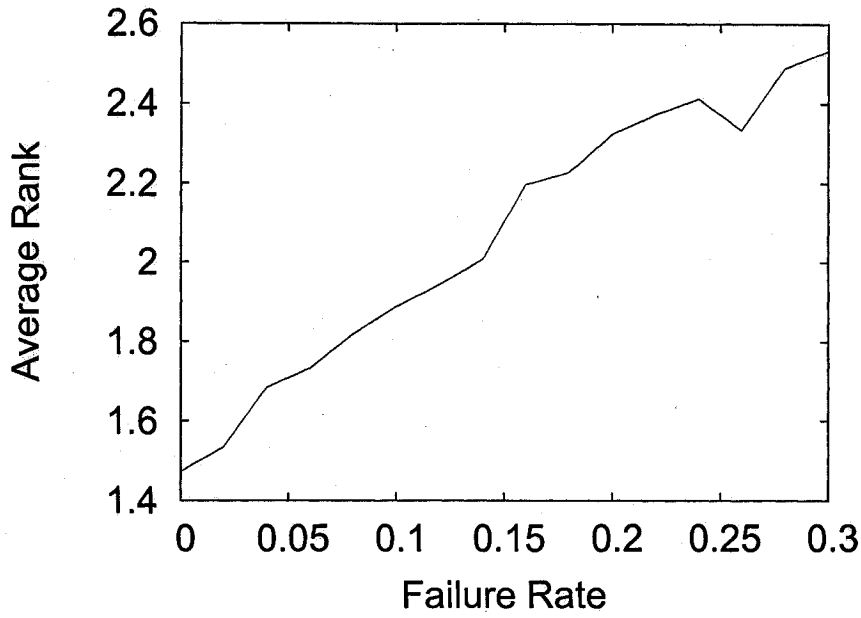

Figure 4.7: Average Rank vs. Failure Rate

4.5 Conclusion

Considering the increasing demand of interactive 3D application on mobile devices and the weak hardware support for geometry-based rendering on such devices, the 3D image-based rendering technique is treated as a promising solution to running interactive 3D applications on mobile devices. Among many barriers to utilize image-based rendering technique on mobile devices, the large size of the data set used by most up-to-date image-based rendering algorithm is an important one. This chapter presents a robust local buffer management technique based on the interpolation model of image-based rendering. Our proposed technique puts a loose restriction on the actual rendering algorithm; therefore, it is able to support all up-to-date image-based rendering algorithms. Three aspects of our proposed technique have been tested using experiments.

Chapter 5

Light-weight Rendering Algorithm

This chapter presents a light-weight 3D image-based rendering algorithm for mobile devices. The proposed algorithm places loose restriction on source camera placement and does not depends on hardware 3D accelerators. Results have shown that the rendering performance of the proposed algorithm is better than other known algorithms described in literature.

5.1 Introduction

With the development of processing power and transmission bandwidth of wireless devices, interactive 3D applications on wireless devices have been made possible. However, the rendering ability of wireless devices for traditional polygon based geometric models is weak compared to desktop personal computers or workstations, which can integrate dedicated rendering acceleration hardware. Moreover, since the rendering time for geometric models depends on the complexity of the scene, it may vary dramatically. These make real time rendering for geometric models on wireless devices a challenging task. In order to overcome the limitations for the emergence of interactive 3D applications on wireless devices, different approaches have been proposed, which can be classified into two categories:

1. collaborative rendering tasks between server and wireless devices; and

2. reduced computing power requirement for the rendering task on wireless devices.

[1]This chapter appears in the Proceedings of 2006 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks [8]

For the first category, two strategies can be further applied: 1) rendering the new view on the server side and transferring the resulting image to the wireless devices; and 2) splitting the rendering task to both server and client side. The first strategy has a potential scalability issue - it is very difficult for commercial servers to render photo-realistic images for a large amount of clients in real time. For the second strategy, partial images are rendered on the server side (first phase rendering), and transferred to the client side for the second phase rendering. This solution employs image-based based rendering on the client side.

Image-based rendering is a classic problem in computer graphics and computer vision areas [36], which has attracted the attention of many researchers in recent years [62]. In literature, methods for using image-based rendering technology on wireless devices have been introduced. Most of current approaches to image-based rendering migrates existing image-based rendering methods from local data set to wireless environment [26, 53].

Although processors in wireless devices are increasingly more powerful and image-based rendering does not require as much computing power as geometry-based rendering, a light weight rendering algorithm solution should always be pursued. This chapter presents a novel image-based rendering algorithm for wireless devices. The proposed algorithm puts loose restriction on source camera placement and does not require hardware 3D accelerators. This algorithm allows the viewer make movements based on any sampling scheme of the Plenoptic Function.

This chapter is organized as follows. The algorithm and the experimental results are presented in Section 5.2 and Section 5.3, respectively. Section 5.4 makes conclusion for this chapter.

5.2 Rendering Algorithm Description

There is no difference between the actual rendering method for local data set and remote data set; thus, any rendering algorithm which can be used for local data set can also be used for remote data set. However, considering the limited processing power of most wireless devices, the rendering algorithm for such devices should be lightweight; and, at the same time, an acceptable rendering speed can be sustained. In the next sections we describe the foundations for our lightweight rendering algorithm.

5.2.1 Selection of the Source Information

Since image-based rendering is an interpolation of the Plenoptic Function with additional geometric information, the selection of source information decides the algorithm. Various types of Plenoptic Function samples have been used in different ways by various image-based rendering approaches, such as image and panorama. The selection of what type of geometric information is used also differs. Before deciding on which source information to select, the following requirements have to be considered:

1. The source information, expressed as image or geometry, should be easily available for both real and virtual scenes;

2. When sampling Plenoptic Function, special devices and special configuration of source cameras should be avoided in order not to limit potential future applications;

3. The registration process of the source images should be simple, in order to simplify also the implementation of the data set.

By considering the above requirements, images taken by commercially available cameras and camera calibration information are selected as source data for our algorithm. The reason why images are used is obvious: they are the easiest samples of the Plenoptic Function. Although panoramic images cover a wider field of view than common images, its speciality (special lens, need for continuing camera rotation, which introduces large non-linear distortion etc.), makes it unfavorable. For the selection of auxiliary geometric information, it is clear that depth information is hard to get for real scenes. Both pixel correspondence and camera calibration information require user input, such as locating correspondence and setting up checkboard. However, the pixel correspondence describes the relationship between two images. When facing remote data set, the availability of source images cannot be determined beforehand. As a result, a large number of images correspondence descriptions is needed. At the worst case, the number is $N * (N-1)/2$, where N is the number of source images in the data set. Conversely, with a reasonable assumption that the internal parameters of a camera is kept constant, calibration information independently describes the geometry aspects of an image what makes it favorable to our solution.

5.2.2 View Morphing Concepts

Based on the selection of the source data, we build our rendering algorithm: view morphing based on epipolar geometry. This method is based on view morphing, but avoids using depth information and pixel correspondence. This section describes the View Morphing concept, followed by the description of our algorithm. View morphing is described by Seitz and Dyer in [42]. Its main goal is to keep geometric correctness while synthesizing intermediary views along a line which combines the center of two source cameras. The principle of view morphing is as follows: if a 3D point is seen by two different images, when synthesizing a new view from a point on the line segment connecting the two camera centers, the camera matrix can be written as $\Pi_s = (1-s)\Pi_0 + s\Pi_1$, where Π_0 and Π_1 are camera matrices of image 0 and image 1, respectively. With the interpolated matrix, it is possible to re-project this 3D point to the image plane of the novel view. This is shown in figure 5.1.

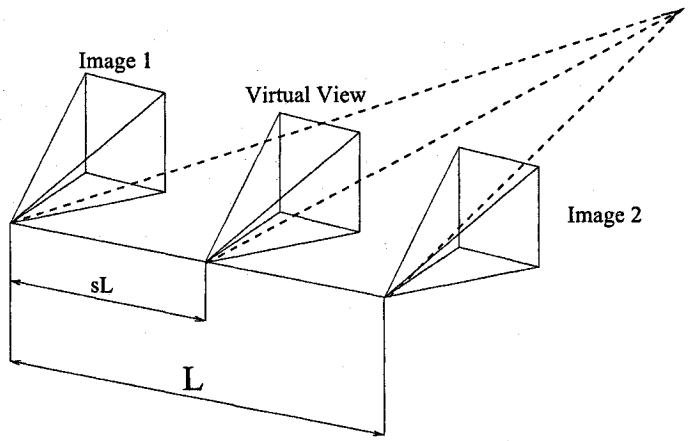

Figure 5.1: Principle of View Morphing

It is clear that depth information or pixel correspondence is needed for using view morphing as an additional geometric information. With this additional geometric information and camera matrices, the projection of a 3D point can be found after getting the camera matrix of the virtual view Π_s, e.g., $p_s = \Pi_s P$. Similarly, the intensity of pixels in the new view can also be calculated by using the same linear interpolation method.

5.2.3 View Morphing Based on Epipolar Geometry

Our algorithm supports interpolation of a new view by combining more than two images. At the same time, in order to make it light-weighted, epipolar geometry is employed instead of projecting a 3D point by using the interpolated camera matrix.

Epipolar Line Searching

Since the depth information and pixel correspondence are not always available, an alternative way to find the necessary additional geometric information is required. Yang et al. [59] suggest a plane sweeping method, which is shown in figure 5.2.

This method assigns an array of assumed depth values to each virtual ray and re-projects the 3D points along each virtual ray with assumed depth values to each source image. When the colour difference between small patches around corresponding pixels get minimum value, the correspondent depth value is accepted. For a wireless devices, this method is very heavy in computing aspects. The re-projection operation, expressed as a 3*4 matrix multiplying a 4*1 vector, requires 16 multiplying and 4 addition operations for each vertex with each assumed depth values, on each source image. Moreover, colour similiarity detection requires calculating color difference in a 2D area for each pixel.

To make this method better fit for wireless devices, we employ epipolar geometry[32] for reducing computing load. Epipolar geometry describes geometric relationship between two projections of a same 3D point. All constraints of epipolar geometry can be encapsulated into fundamental matrix F, which is a 3*3 matrix. Given fundamental matrix F between two views of a same point, expressed as V and V', a pair of correspondent projections p and p', expressed in homogeneous coordinates, satisfies $p'^T Fp = 0$. Therefore, given a pixel (u, v) in V, the correspondent pixel (u', v') in V' should satisfy $u'(f_{11}u + f_{12}v + f_{13}) + v'(f_{21}u + f_{22}v + f_{23}) + (f_{31} + f_{32} + f_{33}) = 0$. It is clear that the correspondent pixel can be locate on a line segment, which is called epipolar line. This property is shown in figure 5.3.

There are many ways for calculating fundamental matrix F. An intuitive method is to use the property $p'Fp = 0$. If the correspondence relation between a pair of pixels is known, it is clear that a linear equation can be set up by the property $u'(f_{11}u + f_{12}v + f_{13}) + v'(f_{21}u + f_{22}v + f_{23}) + (f_{31} + f_{32} + f_{33}) = 0$, with known u', v', u, and v. With eight pairs of correspondent pixels, a linear function group can be created. By solving this function group, F can be solved at a scalar level. This method can be trace back to early 80's and is widely used in computer vision community. However, this method

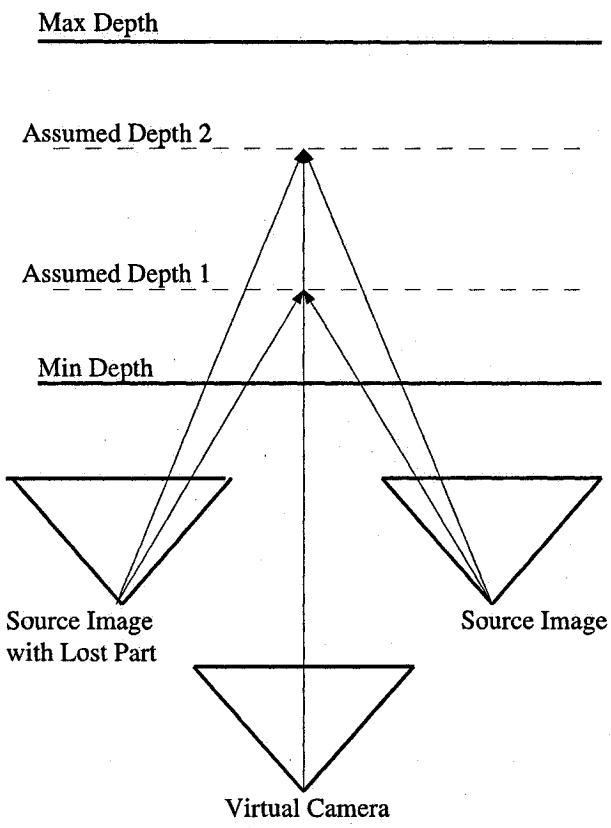

Figure 5.2: Plane Sweeping

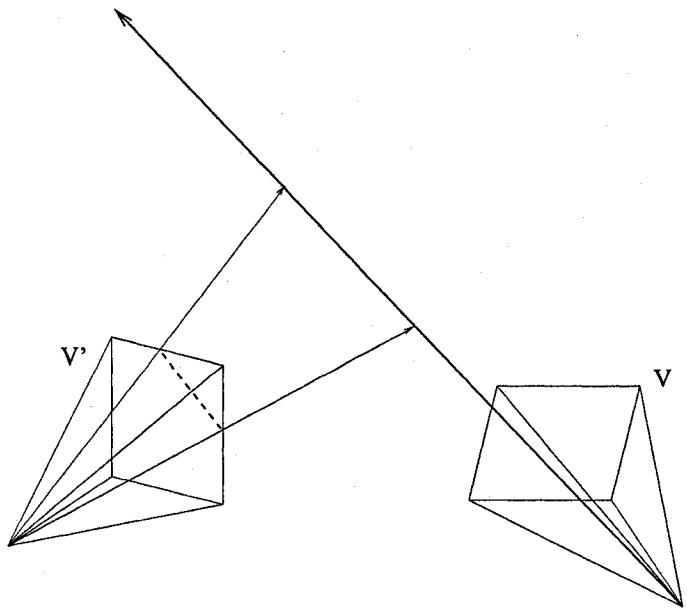

Figure 5.3: Epipolar Geometry

requires building pixel correspondence, which we want to avoid. Instead, a method based on camera calibration information is employed in our proposed algorithm. A method to compute fundamental matrix from camera matrices is presented in [32]. Given camera matrix P and P', which integrated internal and external parameters, fundamental matrix F can be given as $F = [P'C]_X P'P^+$, where C is centre of camera P, $[P'C]_X$ is the corresponding skew-symmetric matrix of vector $P'C$, and P^+ is $P^T(PP^T)^{-1}$. The core of this method is to recognize the fact that the projection of camera P centre must be the join point of all epipolar lines, which is called epipole, as shown in figure 5.4. From this, we can calculate fundamental matrix F with known matrice of source images and the virtual camera, which are required as additional geometric information by our approach. This calculation happens at each arrival of a new source image. It provides the epipolar line for any pixel in novel view at every arriving source image.

Given epipolar lines in two or more source images for new pixels, their correspondent pixels can be searched along those line segments. With the epipolar constraint, we can easily find the end points of an epipolar line segment - this will reduce the search range. Geometrically, epipolar line segment reflects the fact that only a limited depth range of a virtual ray can be projected to another view. As shown in figure 5.3, epipolar search is the same as plane sweeping described in [59]. However, the operations are reduced from

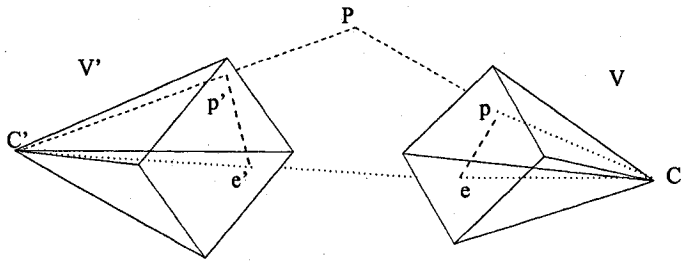

Figure 5.4: Epipole

16 multiplying and 4 addition to 1 multiplying and 1 addition. As as example, searching u' along v' can be represented as $u' = [-(f_{31} + f_{32} + f_{33}) - v'(f_{21}u + f_{22}v + f_{23})]/(f_{11}u + f_{12}v + f_{13})$. Moreover, colour similarity detection can be performed along 1D epipolar line. This reduction occurs for each pixel, at each assumed depth value, and in each source image which obviously reduces the computing load.

Texture Projection

To texture the novel view, a 2D mesh with depth information on its vertices is created. This mesh can reduce the calculation costs during rendering, because it enables texture projection piece by piece instead of pixel by pixel. After building the 2D mesh, we can find the correspondent pixels and acceptable depth value for each vertex of the mesh in the source images through epipolar search. With this depth value, we can weight reference rays in the source images in alpha channel for interpolating the interested virtual ray. The weighting mechanism used in our algorithm is similar to the one described in [10] the difference is that we use the cosine of the angle between the reference ray and the virtual ray. Figure 9 shows our weighting functions. By using the cosine of an angle instead the angle itself makes calculation easier. The cosine value can be calculated from the dot product of the two vectors representing the rays. All weighting factors are normalized to give every vertex of the mesh the same intensity after the weighting process. In the texture projection process, OpenGL ES is used as the graphic system, benefiting from the optimized rendering pipeline. In order to conform to the specification of OpenGL ES, uncompressed source images with each colour channel described by a GL_UNSIGNED_BYTE was used in our solution.

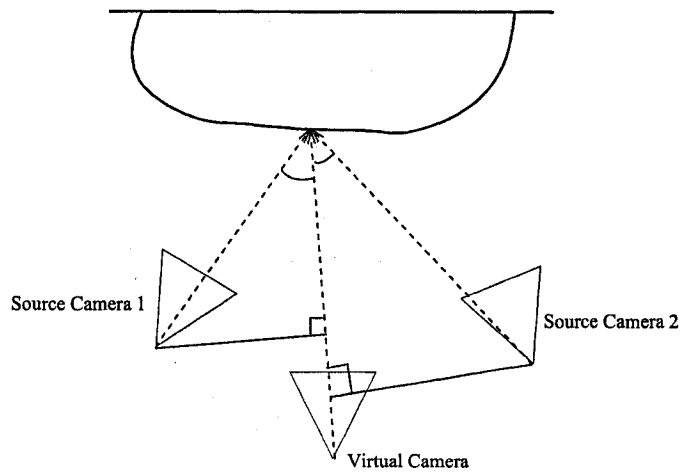

Figure 5.5: Weighting Functions

5.3 Experiment Results

In order to validate the effectiveness of our epipolar searching approach, its rendering speed is compared with the rendering speed of plane sweeping. The experiments environment is described as follows: three scenes are selected as the testing scenes. For each testing scene, thirty new views are rendered by both the epipolar search method and the plane sweeping method on same platform. The average of the thirty trials is accepted as test result. Table 5.1 shows the obtained results. For each row in the table, column two and column three show the rendering time taken by epipolar search and plane sweeping, respectively, in ms (0.001 second). The last column shows the ratio between rendering time used by epipolar search and plane sweeping. From this table, it can be seen that epipolar search is much faster than plane sweeping - it can render in near real-time speed.

Figures 5.6 to 5.11 show the rendering result of the testing scenes. From these images, it can be seen that the rendering quality of plane sweeping is a little bit better than epipolar search. The quality difference is caused by the inherent difference between these two methods: for epipolar search, the test depth is along the ray, connecting the center of the virtual camera and interested vertex of a 2D mesh. This makes the depth interval not uniform along the norm direction of the image plane. Conversely, plane sweeping uses uniform depth interval along norm direction of the image plane. Therefore, it is more accurate when computing disparity. However, considering wireless devices hardware and network limitations (including display low resolution), as well as real-time requirements

Table 5.1: Rendering Time Comparison

Scene No.	Epipolar Search	Plane Sweeping	Ratio
1	169.98 ms	210.32 ms	0.8
2	75.48 ms	165.48 ms	0.46
3	180.64 ms	235.16 ms	0.76

for 3D interaction, epipolar search can be a good image-based rendering solution for 3D rendering in wireless devices.

5.4 Conclusions

This paper describes an architecture for the rendering of 3D scenes. The proposed algorithm significantly reduces rendering time compared to Plane Sweeping, a well known rendering algorithm. Moreover, the presented solution takes data availability into consideration: it separates the data requesting task from the rendering task. As a result, these two tasks can run in parallel and communicate to each other with minimum overhead. Considering the results obtained so far, our algorithm can be a promising solution for 3D rendering in wireless devices.

Figure 5.6: Scene 1 (rendered using plane sweeping)

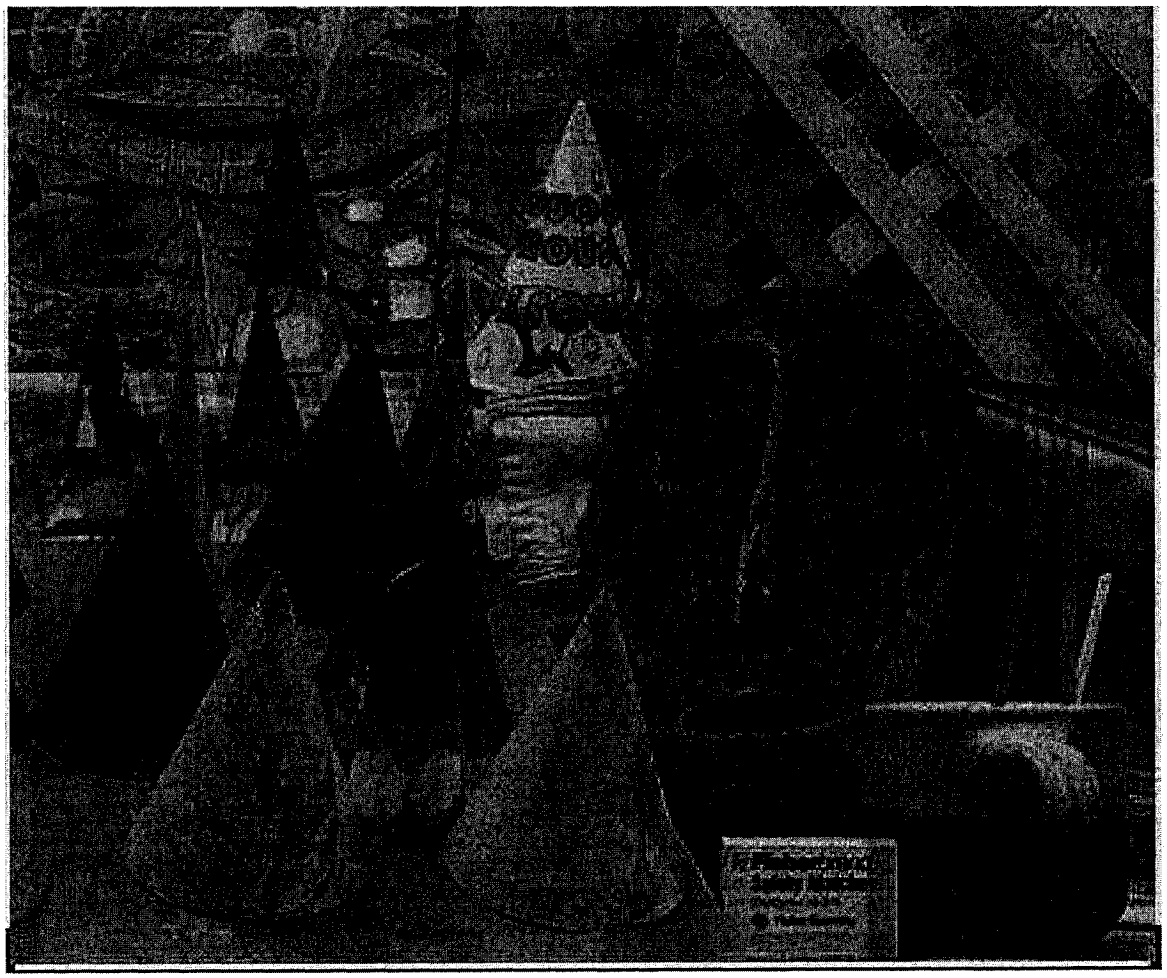

Figure 5.7: Scene 1 (rendered using epipolar search)

Figure 5.8: Scene 2 (rendered using plane sweeping)

Figure 5.9: Scene 2 (rendered using epipolar search)

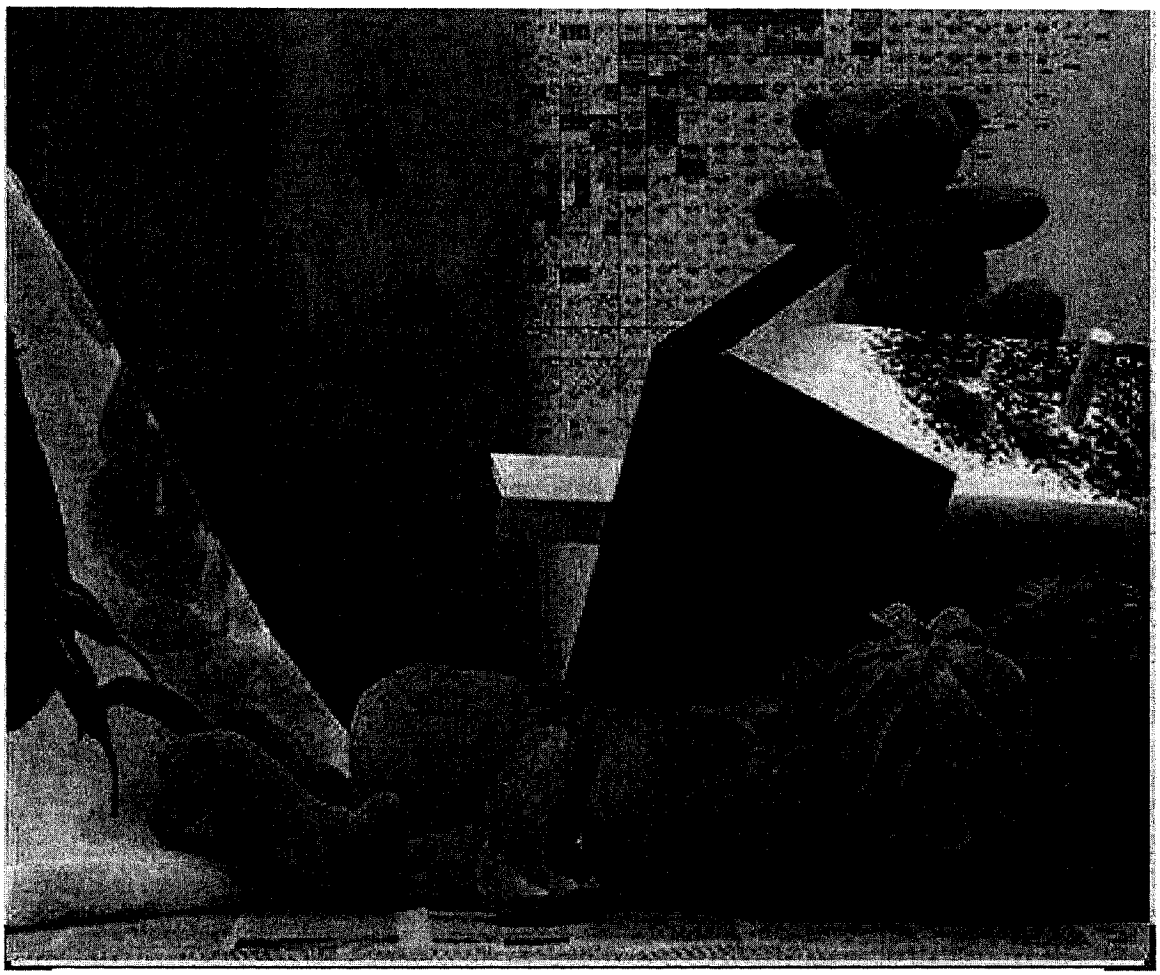

Figure 5.10: Scene 3 (rendered using plane sweeping)

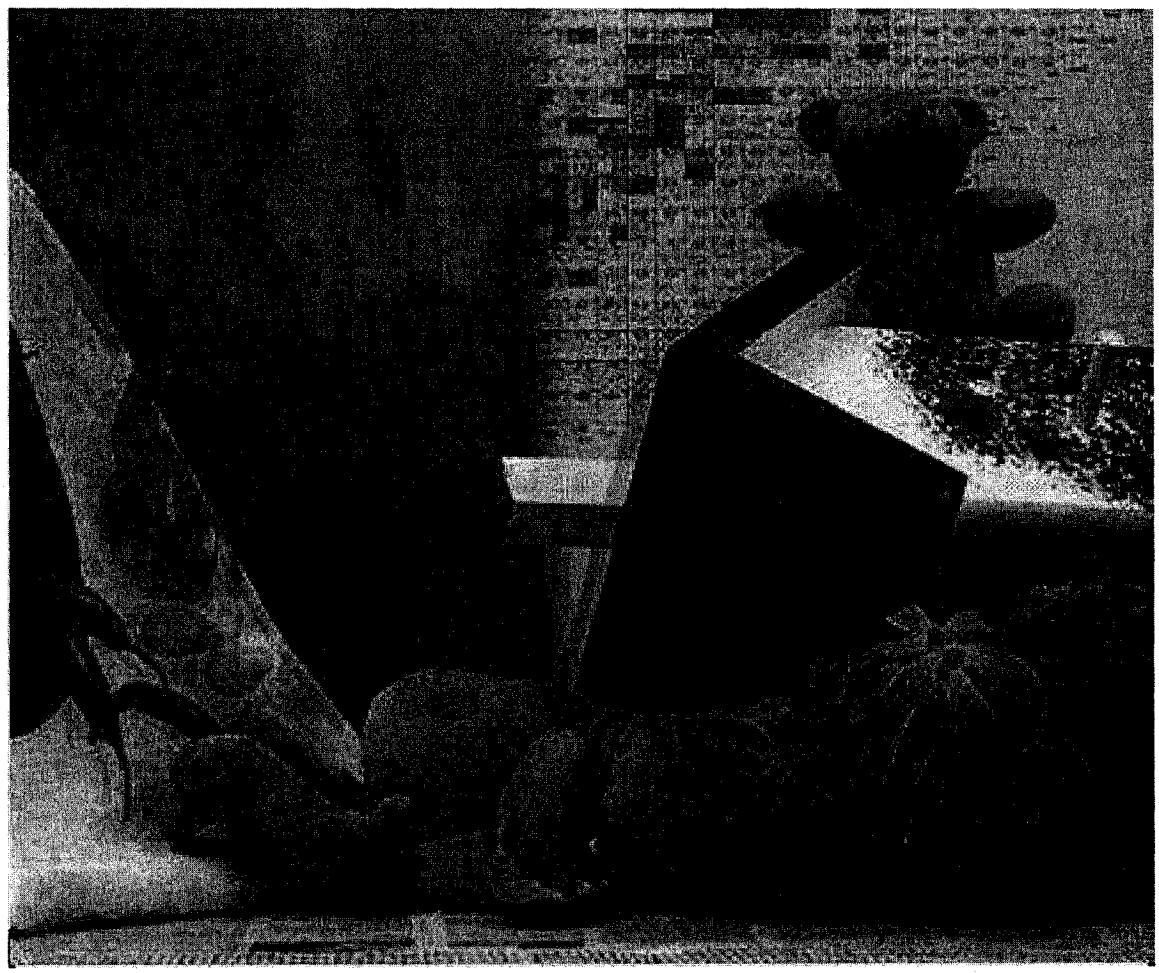

Figure 5.11: Scene 3 (rendered using epipolar search)

Chapter 6

Conclusion and Future Work

In this chapter, we summarize the contributions of this thesis and discuss briefly what future research work is required for this topic. The contributions of this thesis lay in three aspects: a new definition and classification of image-based rendering, a local buffer management scheme, and a light-weight rendering algorithm.

In Chapter 2, we define image-based rendering techniques using the interpolation theory. Therefore, all image-based rendering techniques can be considered as interpolation operations of the Plenoptic Function. Based on this definition, a hierarchical classification of image-based rendering is proposed. At the first level, different principles of physics are utilized so that all seven dimensions of the Plenoptic Function are divided into three classes and corresponding interpolation operations are introduced. There are three classes of interpolation operations at the first level, namely, chronic, optic, and geometric interpolations. Chronic and optic interpolations are relatively easy because both of them handle only one dimension of the Plenoptic Function. However, on the contrary, geometric interpolation is difficult because five of the seven dimensions are covered by this class. Unlike previous surveys, we organize techniques in this class into four categories with more respect given to their suitable usages. A summary is given in Chapter 2 to highlight the conclusion of the survey.

When utilizing image-based rendering techniques over wireless networks, the remoteness of the reference data set is critical to performance. In order to handle this remoteness, Chapters 3 and 4 suggest a local buffer management scheme. In Chapter 3, the performance of different buffer management policies is compared based on customized QoS factors. The experimental result shows that priority queues have the best performance when the viewer makes simple movements. The advantage of the priority queue

policy is inherent from the interpolation model of image-based rendering techniques. In Chapter 4, we introduce a two-level buffer management mechanism using the priority queue policy.

Taking the weak computing power of most mobile devices into consideration, a light-weight rendering algorithm is introduced in Chapter 5. The proposed algorithm does not rely on special camera configuration and scene geometry. Instead, camera calibration information, which is easier to obtain than scene geometry for real scenes, is used as the auxiliary to colour information of the reference images. Epipolar constraint is employed in the colour consistency check and correspondence searching. The experimental result shows that the proposed algorithm has a better performance than other similar algorithms described in literature.

Utilizing 3D image-based rendering techniques over wireless networks is a challenging task. In this thesis, local buffer management and rendering algorithms are discussed. However, there are more technical issues that need to be investigated in future research. In the rest of this chapter, we briefly discuss some topics for future research that can be foreseen at this time.

The proposed buffer management scheme handles the reference images at the image level, while the rendering algorithm also relies on whole images. In order to enhance performance, it is better to manage and render at the sub-image level. Therefore, the question of how to define reference data at sub-image level, how to organize the local buffer at this level, and how to render using sub-images are interesting topics for future research work.

Compared with image-based rendering using a local data set, reference images arrive at the client terminal progressively when the data set resides remotely. It is possible to render with progressively arrived images so that the viewer can have a quick view at the initial stage and decide whether to change position or orientation.

Another interesting topic is related to compression. It is a common case that much redundant information exists in the data set used by image-based rendering techniques. Transmitting all of this redundant information through wireless networks may be expensive and time-consuming. Thus, an effective encoding/decoding system should be developed for the transmitting of reference images.

In a summary, 3D image-based rendering over wireless networks is a very attractive area of research. With the development of image-based rendering techniques and wireless communication, more and more new techniques are expected to appear.

Bibliography

[1] Edward H. Adelson and James R. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, pages 3–20. MIT Press, October 1991.

[2] AMD. *3DNow! technology manual*, March 2000.

[3] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '92)*, pages 35–42, July 1992.

[4] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, October 1976.

[5] Sean Borman and Robert L. Stevenson. Super-resolution from image sequences-a review. In *Proceedings of 1998 IEEE Midwest Symposium on Circuits and Systems*, pages 374–378, August 1998.

[6] Azzedine Boukerche and Trivikram Dash. Performance evaluation of a generalized hybrid tdma/cdma protocol for wireless multimedia with qos adaptations. *Computer Communications*, 28(12):1468–1480, July 2005.

[7] Azzedine Boukerche and Jing Feng. Buffer management for 3d image-based rendering over wireless network with qos adaptation. In *Proceedings of the 31st IEEE Conference on Local Computer Networks*, pages 55–62, November 2006.

[8] Azzedine Boukerche, Jing Feng, and Regina Borges de Araujo. A 3d image-based rendering technique for mobile handheld devices. In *Proceedings of IEEE 2006 International Symposium on a World of Wireless Mobile and Multimedia Networks*, pages 325–331, June 2006.

[9] Azzedine Boukerche, Jing Feng, and Richard Pazzi Werner. A buffer management technique for 3d image-based rendering on mobile devices. In *Proceedings of 4th ACM International Workshop on Mobility Management and Wireless Access*, pages 158–163, October 2006.

[10] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*, pages 425–432, August 2001.

[11] Brian Cabral, Marc Olano, and Philip Nemec. Reflection space image based rendering. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*, pages 165–170, August 1999.

[12] Edwin E. Catmull. *A subdivision algorithm for computer display of curved surfaces.* PhD thesis, University of Utah, December 1974.

[13] Chun-Fa Chang and Shyh-Haur Ger. Enhancing 3d graphics on mobile devices by image-based rendering. In *Proceedings of Advances in Multimedia Information Processing - PCM 2002*, pages 1105–1111, December 2002.

[14] Shenchang Eric Chen. Quicktime vr an image-based approach to virtual environment navigation. In *Proceedings of the 22th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*, pages 29–38, August 1995.

[15] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93)*, pages 279–288, August 1993.

[16] Herbert Aron David. *Order statistics.* Wiley, 1970.

[17] Paul E. Debevec. Image-based modeling and lighting. *ACM SIGGRAPH Computer Graphics*, 33(4):46–50, November 2000.

[18] Sergey Fomel. *Three-dimensional seismic data regularization.* PhD thesis, Stanford University, March 2001.

[19] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pages 43–54, August 1996.

[20] Ned Greene and Paul S. Heckbert. Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(6):21–27, June 1986.

[21] Paul S. Heckbert. Fundamentals of texture mapping and image warping. Master's thesis, University of California, Berkeley, June 1989.

[22] Intel. *Pentium processor with MMX technology*, June 1997.

[23] Sing Bing Kang. A survey of image-based rendering techniques. In *Proceedings of SPIE - Volume 3641, Videometrics VI*, pages 2–16, December 1998.

[24] Sing Bing Kang, Richard Szeliski, and P. Anandan. The geometry-image representation tradeoff for rendering. In *Proceedings of 2000 International Conference on Image Processing, Volumn 2*, pages 13–16, September 2000.

[25] Stéphane Laveau and Olivier Faugeras. 3-d scene representation as a collection of images and fundamental matrices. Technical Report RR-2205, The French National Institute for Research in Computer Science and Control (INRIA), February 1994.

[26] Yu Lei, Zhongding Jiang, Deren Chen, and Hujun Bao. Image-based walkthrough over internet on mobile devices. In *Grid and Cooperative Computing - GCC 2004 Workshops: GCC 2004 International Workshops, IGKG, SGT, GISS, AAC-GEVO, and VVS*, pages 728–735, October 2004.

[27] Jed Lengyel. The convergence of graphics and vision. *Computer*, 31(7):46–53, July 1998.

[28] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pages 31–42, August 1996.

[29] Jiang Li, Yiying Tong, Yong Wang, Heung-Yeung Shum, and Ya-Qin Zhang. Image-based walkthrough over the internet. In *Proceedings of International Workshop on Very Low Bitrate Video Coding, VLBV01*, October 2001.

[30] Andrew Lippman. Movie-maps: an application of the optical videodisc to computer graphics. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '80)*, pages 32–42, July 1980.

[31] Songwu Lu, Vaduvur Bharghavan, and R. Srikant. Fair scheduling in wireless packet networks. *IEEE/ACM Transactions on Networking*, 7(4):473–489, August 1999.

[32] Quan-Tuan Luong and Olivier D. Faugeras. The fundamental matrix: Theory, algorithms, and stability analysis. *International Journal of Computer Vision*, 17(1):43–75, January 1996.

[33] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pages 369–374, August 2000.

[34] Leonard McMillan. *An image-based approach to three-dimensional computer graphics.* PhD thesis, University of North Carolina at Chapel Hill, March 1997.

[35] Leonard McMillan and Gary Bishop. Plenoptic modeling: an image-based rendering system. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*, pages 39–46, August 1995.

[36] Leonard McMillan and Steven Gortler. Image-based rendering: a new interface between computer vision and computer graphics. *ACM SIGGRAPH Computer Graphics*, 33(4):61–64, November 2000.

[37] David L. Milgram. Computer methods for creating photomosaics. *IEEE Transactions on computers*, C-24(11):1113–1119, November 1975.

[38] David L. Milgram. Adaptive techniques for photomosaicking. *IEEE Transactions on computers*, C-26(11):1175–1180, November 1977.

[39] Shree K. Nayar. Catadioptric omnidirectional camera. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997*, pages 482–488, June 1997.

[40] Manuel M. Oliveira. Image-based modeling and rendering techniques: a survey. *RITA - Revista de Informática Teórica e Aplicada*, 9(2):37–66, October 2002.

[41] Shmuel Peleg and Joshua Herman. Panoramic mosaics by manifold projection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997*, pages 338–343, June 1997.

[42] Steven M. Seitz and Charles R. Dyer. View morphing. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pages 21–30, August 1996.

[43] Jonathan Shade, Steven Gortler, Li wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, pages 231–242, August 1998.

[44] Heung-Yeung Shum and Li-Wei He. Rendering with concentric mosaics. In *Proceedings of the 26rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*, pages 299–306, August 1999.

[45] Heung-Yeung Shum, Sing Bing Kang, and Shing-Chow Chan. Survey of image-based representations and compression techniques. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(11):1020–1037, November 2003.

[46] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating system concepts.* Jone Wiley and Sons. Inc, 7th edition, 2005.

[47] Pierre Solle. Morphological image compositing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5):673–683, May 2006.

[48] Johan F. Steffensen. *Interpolation*, chapter 1. Chelsea Publishing Company, 2nd edition, 1950.

[49] Sun. *The VIS instruction set*, June 2002.

[50] Peter Symes. *Digital video compression*, chapter 8. McGraw-Hill, 2004.

[51] Peter Symes. *Digital video compression.* McGraw-Hill, 2004.

[52] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*, pages 251–258, August 1997.

[53] Gwenola Thomas, Gérald Point, and Kadi Bouatouch. A client-server approach to image-based rendering on mobile terminals. Technical Report RR-5447, The French National Institute for Research in Computer Science and Control (INRIA), January 2005.

[54] Tomas Akenine-Möller and Eric Haines. *Real-time rendering*, chapter 6.2. A K Peters, 2nd edition, 2002.

[55] Matthew Uyttendaele, Ashley Eden, and Richard Szeliski. Eliminating ghosting and exposure artifacts in image mosaics. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Volume II*, pages 509–516, December 2001.

[56] Vassilios Vlahakis, Nikolaos Ioannidis, John Karigiannis, Manolis Tsotros, Michael Gounaris, Didier Stricker, Tim Gleue, Patrick Daehne, and Luís Almeida. Archeoguide: an augmented reality guide for archaeological sites. *Computer Graphics and Applications, IEEE*, 22(5):52–60, September/October 2002.

[57] Bennett S. Wilburn, Michal Smulski, Hsiao-Heng K. Lee, and Mark A. Horowitz. Light field video camera. In *Proceedings of SPIE - Volume 4674, Media Processors 2002*, pages 29–36, December 2002.

[58] George Wolberg. *Digital image warping*, chapter 3. IEEE Computer Society Press, 1990.

[59] Jason C. Yang, Matthew Everett, Chris Buehler, and Leonard McMillan. A real-time distributed light field camera. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 77–86, June 2002.

[60] Ilmi Yoon and Ulrich Neumann. Image-assisted visualizations over networks. *Journal of Electronic Imaging*, 12(2):355–363, April 2003.

[61] Cha Zhang and Tsuhan Chen. A self-reconfigurable camera array. In *Proceedings of the 15th Eurographics Workshop on Rendering Techniques*, pages 243–254, June 2004.

[62] Cha Zhang and Tsuhan Chen. A survey on image-based rendering – representation, sampling and compression. *EURASIP Signal Processing: Image Communication*, 19(1):1–28, January 2004.

[63] Zhengyou Zhang. Image-based geometrically-correct photorealistic scene/object modeling (ibphm): A review. In *Proceedings of the 3rd Asian Conference on Computer Vision (ACCV'98), Volume II*, pages 340–349, January 1998.

[64] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, November 2000.

[65] Barbara Zitová and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, October 2003.