ON SAMPLING OF IMAGE-BASED RENDERING DATA

Cha Zhang
June 2004

Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee
Tsuhan Chen, Chair
Takeo Kanade
José Moura
Richard Szeliski (Microsoft Research)

Abstract

Image-based rendering (IBR) generates novel views from images instead of 3D models. It can be considered as a process of sampling the light rays in the space and interpolating the ones in novel views. The sampling of IBR is a high-dimensional sampling problem, and is very challenging. This thesis focuses on answering two questions related to IBR sampling, namely how many images are needed for IBR, and if such number is limited, where should we capture them. There are three major contributions in this dissertation.

First, we give a complete analysis on uniform sampling of IBR data. By introducing the surface plenoptic function, we are able to analyze the Fourier spectrum of non-Lambertian and occluded scenes. Given the spectrum, we also apply the generalized sampling theorem on the IBR data, which results in better rendering quality than rectangular sampling for complex scenes. Such uniform sampling analysis provides general guidelines on how the images in IBR should be taken. For instance, it shows that non-Lambertian and occluded scenes often require higher sampling rate.

Second, we propose a very general sampling framework named freeform sampling. Freeform sampling has three categories: incremental sampling, decremental sampling and rearranged sampling. When the to-be-reconstructed function values are unknown, freeform sampling becomes active sampling. Algorithms of active sampling are developed for image-based rendering and show better results than the traditional uniform sampling approach.

Third, we present a self-reconfigurable camera array that we developed, which features a very efficient algorithm for real-time rendering and the ability of automatically reconfiguring the cameras to improve the rendering quality. Both are based on active sampling. Our camera array is able to render dynamic scenes interactively at high quality. To our best knowledge, it is the first camera array in literature that can reconfigure the camera positions automatically.

Acknowledgements

There are many people who I am deeply indebted to, and without whom this work would never have been finished. First and foremost is my advisor, Prof. Tsuhan Chen. He has served all of the roles of teacher, mentor, guide, encourager, task-master, advocate, and friend. Throughout our weekly group and individual meetings, he teaches me how to select topics, conduct research and communicate effectively. I have learnt from him how to become a successful researcher and I hope to carry on to make contributions to our community. I am very fortunate to become his student and join his group at CMU.

I would also like to thank Prof. Takeo Kanade, Prof. José Moura and Dr. Richard Szeliski for spending their valuable time as my thesis committee members. I would like to thank them for all the fruitful discussions that improve the quality of this work.

My current and previous group mates Fu Jie Huang, Ta-Chien Lin, Deepak Turaga, Trista Chen, Howard Leung, Xiaoming Liu, Claire Fang, Wende Zhang, Edward Lin, Sam Chen, Jessie Hsu, Simon Lucey, Jack Yu, Kate Shim, Avinash Baliga, Michael Kaye, David Liu, Li-Ying Chang and Akira Kubota, broaden my views through the exchange of research ideas. I have developed the sense of belonging and I am proud to be in this AMP (Advanced Multimedia Processing) group. In addition to my group, I also enjoy discussing research ideas with other friends such as Dapeng Wu, Jin Lu, Ying Sun, David Sepiashvili, etc. It was really a great experience to have been in the Electrical and Computer Engineering department at Carnegie Mellon University, where I have the opportunity to meet so many great faculties, staff and students.

I should also thank Dr. Jin Li and Dr. Stephen Lew for supervising me during my internship at Microsoft Research Redmond and NVidia Corp. The experience of internship

greatly broadened my research field and helped me understand better why and how research should be applied to industry.

Most of all, I would like to thank my wonderful wife, Jing Ding. Without her support, encouragement, and confidence in me from the day I came to graduate school until now, this thesis would certainly have gone unwritten. She is my helper and friend, and deserves more credit than I can put into words. I dedicate this thesis to her, to our lovely son, Alex Muyang Zhang, and to my parents.

Table of Contents

Abstract **ii**

Acknowledgements **iii**

Table of Contents **v**

List of Tables **vii**

List of Figures **viii**

1 Introduction **1**
1.1 Objective and Approaches 1
1.2 Contributions 2
1.3 Guide to the Thesis 3

2 A Survey on IBR Techniques **5**
2.1 Introduction 5
2.2 Constraining the Viewing Space 9
2.3 Introducing Source Descriptions 21
2.4 Summary 32

3 Previous Work on IBR Sampling **34**
3.1 Uniform Sampling 35
3.2 Nonuniform Sampling 38
3.3 Summary 39

4 Uniform Sampling: Spectral Analysis of IBR Data **41**
4.1 The Surface Plenoptic Function 42
4.2 Analysis for the Light Field 46
4.3 Analysis of Scenes with Unknown Geometry 56
4.4 Analysis for the Concentric Mosaics 62
4.5 Summary 64

5 Uniform Sampling: Generalized IBR Sampling **66**
5.1 Generalized Sampling for Light Field Data 66
5.2 Experimental Results 70
5.3 Summary 73

6 Freeform Sampling and Active Sampling: A New Sampling Framework **74**
6.1 The Freeform Sampling Problem 75
6.2 Solutions of the Freeform Sampling Problem 77
6.3 Active Sampling 81
6.4 The Proposed Algorithms for Active Sampling 83
6.5 Summary 87

7 Active Sampling: Applications in IBR **88**
7.1 The Local Consistency Score 88
7.2 IBR Active Incremental Sampling 91
7.3 IBR Active Rearranged Sampling 99
7.4 Summary 106

8 The Self-Reconfigurable Camera Array **107**
8.1 System Overview 108
8.2 Camera Calibration 112
8.3 Real-Time Rendering 114
8.4 Self-Reconfiguration of the Cameras 126
8.5 Summary 131

9 Conclusions and Future Work **132**
9.1 Contributions 132
9.2 Future Work 133

Bibliography **136**

List of Tables

2.1 IBR representations with various viewing space constraints. 12
2.2 Various IBR approaches using source descriptions. 21

5.1 Rendering image qualities for different sampling methods. 72

7.1 Performance comparison between uniform sampling and active incremental sampling on light field scene *Earth*. 94

List of Figures

2.1 The 7D plenoptic function. 7
2.2 One parameterization of the light field. 14
2.3 A sample light field image array: fruit plate. 15
2.4 Concentric mosaics capturing. 17
2.5 Parallax observed from concentric mosaics rendered scenes. 18
2.6 A 360° cylindrical panorama. 19
2.7 Synthesize novel view with tensor space. 24
2.8 The layered depth image. 25

3.1 The plenoptic sampling analysis. 36
3.2 The minimum sampling curve. 37

4.1 The 2D surface plenoptic function and general IBR capturing. 43
4.2 The light field parameterization with surface plenoptic function. 47
4.3 Spectrum of Lambertian and non-Lambertian scenes at constant depth. . . 49
4.4 Spectrum of a tilted planar scene with a sinusoid pasted as texture. . . . 51
4.5 Spectrum of an occluded scene. 55
4.6 Best rectangular sampling of a non-occluded and Lambertian scene. . . . 56
4.7 Best rectangular sampling of a non-Lambertian scene. 57
4.8 An example of the minimum sampling rate for non-Lambertian surface. . . 59
4.9 Best rectangular sampling of occluded scenes. 60
4.10 An example of the minimum sampling rate for occluded surface. 61
4.11 The concentric mosaics parameterization with surface plenoptic function. . 61
4.12 Spectrum of a Lambertian scene at constant depth for concentric mosaics. . 64

5.1 Generalized sampling of light field. 67
5.2 Filter design for light field reconstruction. 69
5.3 Test scenes for generalized sampling of light field. 71

6.1 Illustration of Condition 6.1.2 with a 1D example. 76
6.2 The flow of decremental sampling. 78
6.3 The flow of incremental sampling. 79
6.4 The flow of rearranged sampling. 80
6.5 Cliques and their subdivision. 84

6.6 The flow of active incremental sampling. 85

7.1 Interpolation weight calculation using angular difference. 89
7.2 Cases that will cause a low local consistency score. 90
7.3 Active incremental sampling for the light field setup. 92
7.4 Active incremental sampling of a light field *Earth*. 93
7.5 Active incremental sampling for the concentric mosaics setup. 95
7.6 Concentric mosaics scenes: (i) Reflective cone; (ii) Hemisphere bowl; (a) scene snapshot; (b) used scene geometry. 95
7.7 Active incremental sampling results on concentric mosaics scene RC. 97
7.8 Active incremental sampling results on concentric mosaics scene HB. 98
7.9 Comparison of EPIs for different scenes. 99
7.10 The setup of active rearranged sampling for the light field. 101
7.11 Scaling factor for updating the auxiliary weights. 102
7.12 Active rearranged sampling results of a light field *Teapot*. 104
7.13 The setup of active rearranged sampling for the concentric mosaics. 105
7.14 Active rearranged sampling results of concentric mosaics scene *Hemisphere bowl*. 106

8.1 Our self-reconfigurable camera array system with 48 cameras. 108
8.2 The mobile camera unit. 109
8.3 Images captured by our camera array. 110
8.4 Locate the feature corners of the calibration pattern. 112
8.5 The multi-resolution 2D mesh with depth information on its vertices. 114
8.6 The flow chart of the rendering algorithm. 115
8.7 Locate the neighboring images for interpolation and depth reconstruction through plan sweeping. 116
8.8 Determine the ROI of a certain input image. 117
8.9 Subdivision of a mesh triangle. 119
8.10 Synthetic scenes rendered with our proposed algorithm. 122
8.11 Scenes captured and rendered with our camera array. 124
8.12 Real-world scenes rendered with our proposed algorithm. 125
8.13 Scenes rendered by reconfiguring our camera array. 127
8.14 Self-reconfiguration of the cameras. 128

Chapter 1

Introduction

1.1 Objective and Approaches

Image-based rendering (IBR) has attracted much attention recently. By capturing a set of images of a scene, IBR is capable of rendering 3D views with little or no scene geometry. Compared with traditional model-based rendering methods, IBR bypasses the most difficult geometric model reconstruction stage, and is easy to capture and fast to render.

The tradeoff for not using a geometric model for 3D rendering is the tremendous amount of data one needs to capture and store. Given a scene, it is thus essential to answer the following two questions in IBR applications:

> *How many images are needed for IBR? If such number is limited, where shall we capture these images?*

In literature, answers to these questions are rare and incomplete. Many IBR representations resorted to over-sampling to avoid problems caused by insufficient number of samples such as ghosting or aliasing. Solving the sampling problem is thus fundamental, and may bring insights to the design of future IBR algorithms.

In this dissertation, we will first address the uniform sampling of IBR using traditional Fourier transform based sampling methods. Fourier transform based analysis was adopted in some previous papers, but none of them were able to solve it completely, especially for non-Lambertian and occluded scenes. This dissertation proposes a new uniform sampling

analysis approach based on the so-called surface plenoptic function, which is able to provide a more complete solution. The uniform sampling analysis provides general guidelines on how IBR scenes should be sampled, for instance, a non-Lambertian or occluded object should be sampled denser than Lambertian and non-occluded ones.

Based on the guidelines we achieve from uniform sampling analysis, it is natural to consider freeform or non-uniform sampling for IBR, because real world scenes always vary their surface properties. The second part of the dissertation thus focuses on the freeform sampling of IBR data. We not only answer the question how many images are needed, but also tell where these images should be placed. The result of our sampling scheme is a non-uniform arrangement of the capturing cameras, which leads to better worst case rendering quality and less quality jittering during the rendering.

1.2 Contributions

The key contributions of this dissertation are three-fold:

Uniform IBR sampling analysis. We propose a novel method to analyze the Fourier spectrum of IBR scenes, which is able to handle both non-Lambertian surface and occlusions. We show that in both cases the required sampling rate is higher than Lambertian and non-occluded ones. Considering that the IBR sampling problem is a multi-dimensional sampling problem, we also apply the generalized sampling theorem on IBR sampling. We are able to reduce the sampling rate by a factor of 50% in theory, and achieve better rendering quality for complex scenes.

A very general framework on freeform sampling and active sampling. Compared with the traditional uniform sampling theorem, the freeform sampling framework has more practical considerations such as the reconstruction method, the reconstruction set, the sampling noise, etc. General solutions of freeform sampling are described in this dissertation, including decremental sampling, incremental sampling and rearranged sampling. We also present active sampling as a special case of freeform sampling, where the function values

of the sampled signal on the reconstruction set is unknown. We apply it to IBR and show that it is superior to the traditional uniform sampling method.

The self-reconfigurable camera array. We build the world's first self-reconfigurable camera array, where the cameras are mobile, in order to demonstrate our active sampling theory. We develop a very efficient algorithm for the real-time rendering of dynamic scenes. Active sampling is widely used in the algorithm to improve the rendering speed. We also show that by moving the cameras around for active sampling, we can improve the rendering quality, especially at object boundaries.

1.3 Guide to the Thesis

Here we outline the chapters that follow.

Chapter 2 – A Survey on IBR Techniques. We give a survey of the various IBR techniques in the literature. They are classified into two categories based on how the plenoptic function is simplified: constraining the viewing space or introducing source descriptions. This chapter is written in order to give the reader some background knowledge on IBR.

Chapter 3 – Previous Work on IBR Sampling. The literature on IBR sampling is rare and incomplete. We give brief introduction to them and comment on their pros and cons, which serves as a motivation for this dissertation.

Chapter 4 – Uniform Sampling: Spectral Analysis of IBR Data. In this chapter we propose our general framework of IBR uniform sampling analysis based on the surface plenoptic function. Our method is able to derive the IBR spectrum for non-Lambertian and occluded scenes, which is more complete than previous work.

Chapter 5 – Uniform Sampling: Generalized IBR Sampling. Given the Fourier spectrum of the scene, this chapter focuses on how to sample it efficiently. We apply the generalized sampling theory for high-dimensional signal to IBR, and show its advantage

over the traditional rectangular sampling on complex scenes. However, we conclude that rectangular sampling may still be preferable for its simplicity and good performance.

Chapter 6 – Freeform Sampling and Active Sampling: A New Sampling Framework. This chapter tries to give a very general sampling framework named freeform sampling. Samples are no longer limited to regular patterns, thus solutions to freeform sampling also change. Active sampling is a special case of freeform sampling, where the sample values on the reconstruction set are unknown. General algorithms are also presented for active sampling.

Chapter 7 – Active Sampling: Applications in IBR. We apply the active sampling theorem in the last chapter to IBR. Several algorithms specific for IBR are proposed, and show very good performance on synthetic scenes.

Chapter 8 – The Self-Reconfigurable Camera Array. We describe our self-reconfigurable camera array project in this chapter. Our camera array features real-time calibration, real-time geometry reconstruction and rendering, and self-reconfiguration. It shows examples of how active sampling is applied in real-world applications.

Chapter 9 – Conclusions and Future Work. We conclude this work with a review of our contributions along with some discussions of future work for IBR sampling and the camera array.

Readers are suggested to start with Chapter 2 and 3 if not familiar with IBR and its sampling problem. Chapter 4 and 5 are on the uniform sampling of IBR. They are related but can be read separately. Both Chapter 6 and Chapter 7 are for freeform sampling and active sampling, but Chapter 7 refers many conclusions from Chapter 6. Chapter 8 introduces the self-reconfigurable camera array, which loosely depends on Chapter 6 and 7.

Chapter 2

A Survey on IBR Techniques

In this chapter, we give a brief survey on the various techniques developed for IBR. The goal is to provide an overview of research for IBR in a complete and systematic manner. We observe that essentially all the IBR representations are derived from the plenoptic function, which is seven dimensional and difficult to handle. We classify various IBR representations into two categories based on how the plenoptic function is simplified, namely constraining the viewing space and introducing source descriptions. In the former category, we summarize six common assumptions that were often made in various approaches and discuss how the dimension of the plenoptic function can be reduced based on these assumptions. In the latter category, we further categorize the methods based on what kind of source description was introduced, such as scene geometry, texture map or reflection model. While most work in this category used known source descriptions, we also review some of the techniques that tried to infer source descriptions from captured images and perform high quality rendering.

2.1 Introduction

One might remember that in the movie *Matrix*, the scene with Keanu Reeves dodging the bullets might be one of the most spectacular images ever caught on camera. This filming technology is what the movie producers called *Flo-Mo*. *Flo-Mo* lets the filmmakers shoot scenes where the camera moves at a normal speed while the action is frozen or happens in slow motion. Two movie cameras and 120 computer-controlled still cameras were used in

that scene. Similarly, the *Eyevision* system [48] developed by Takeo Kanade [140], which consisted of 33 cameras spaced approximately 6 degrees apart around the rim of the stadium, was used in a live broadcast of Super Bowl game in Jan. 2001. It provided a unique 3D view of selected plays in a 270° stop action image. These novel viewing experiences were brought to us by image-based rendering (IBR), which has been a very active research topic recently. By capturing a set of images or light rays in the space, the goal of IBR is to reproduce the scene correctly at an arbitrary viewpoint, with unknown or limited amount of geometry. Compared with geometric models that dominate the traditional 3D rendering pipelines, images are easier to obtain, simpler to handle and more realistic to render. Moreover, since image processing is one of the most widely studied research topics in the literature, IBR has attracted many researchers from different communities, including graphics, vision and signal processing.

Since the goal of IBR is to capture and render the 3D world, let us first see how the world can be described. One possible solution is to record all the objects in the world and their interactions, which we call a *source description*. The traditional model-based rendering approach adopts such a description: shapes of the objects are represented by certain geometric models; properties of the object surfaces are described by texture maps and reflection models; lighting and shading are the results of interaction between the light sources and objects, etc. The source description is often compact and insightful, because it tells how the world is composed. However, it has the disadvantage that such a description is not always available. From what we can observe with our eyes or cameras, deriving the source description is not trivial, and has been the goal of computer vision for many years.

An alternative way to describe the world is through the *appearance description*. The appearance of the world can be thought of as the dense array of light rays filling the space, which can be observed by posing eyes or cameras in the space. These light rays can be represented through the plenoptic function, proposed by Adelson and Bergen [1]. As shown in Figure 2.1, the plenoptic function is a 7D function that models a 3D dynamic environment by recording the light rays at every space location (V_x, V_y, V_z), towards every

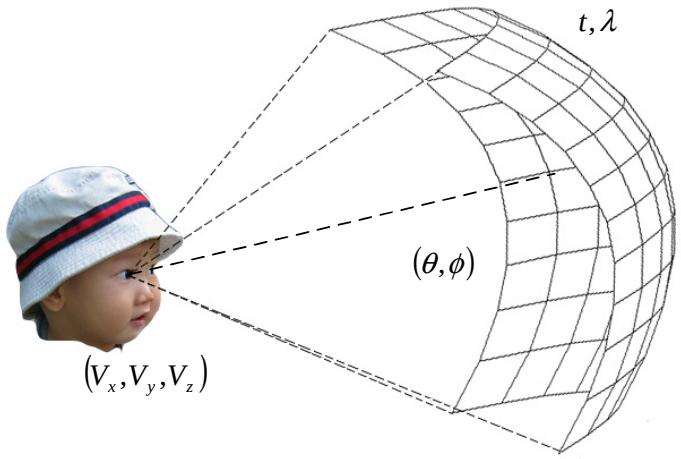


Figure 2.1: The 7D plenoptic function.

possible direction (θ, ϕ), over any range of wavelengths (λ) and at any time (t), i.e.,

$$l^{(7)}(V_x, V_y, V_z, \theta, \phi, \lambda, t) \tag{2.1}$$

As pointed out by Adelson and Bergen [1]:

> The world is made of three-dimensional objects, but these objects do not communicate their properties directly to an observer. Rather, the objects fill the space around them with the pattern of light rays that constitutes the plenoptic function, and the observer takes samples from this function. The plenoptic function serves as the sole communication link between the physical objects and their corresponding retinal images. It is the intermediary between the world and the eye.

When we take an image for a scene with a pinhole camera[1], the light rays passing through the camera's center-of-projection are recorded. They can also be considered as samples of the plenoptic function. As image-based rendering is based on images, it adopts the appearance description. We define IBR under the plenoptic function framework as follows:

[1]Throughout this thesis, we assume that the cameras we use are pinhole cameras. Most of the IBR technologies in the literature made such assumption. There are some exceptions, however, such as the work in [49, 59].

Definition 2.1.1. *Given a continuous plenoptic function that describes a scene,* image-based rendering *is a process of two stages - sampling and rendering*[2]*. 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 from the captured samples.*

The above definition reminds us about what we typically do in signal processing: given a continuous signal, sample it and then reconstruct it. The uniqueness of IBR is that the plenoptic function is 7D - a dimension beyond most of the signals handled before. In fact, the 7D function is so general that, due to the tremendous amount of data required, no one has been able to sample the full function into one representation. Research on IBR is mostly about how to make reasonable assumptions to reduce the sample data size while keeping reasonable rendering quality.

There have been many IBR representations invented in the literature. They basically follow two major strategies in order to reduce the data size. First, one may constrain the viewing space of the viewers. Such constraints will effectively reduce the dimension of the plenoptic function, which makes sampling and rendering manageable. For example, if we limit the viewers' interest to static scenes, the time dimension in the plenoptic function can be simply dropped. Second, one may introduce some source descriptions into IBR, such as the scene geometry. Source description has the benefit that it can be very compact. A hybrid source-appearance description is definitely attractive for reducing the data size. To obtain the source description, manual work may be involved or we may resort to computer vision techniques.

[2]We believe that considering IBR as a generic term for the techniques of both sampling and rendering is more appropriate than the conventional definition in [86]. In the sections that follow, *IBR rendering* will be used to refer to the rendering stage specifically.

2.2 Constraining the Viewing Space

If the viewing space can be constrained, the amount of images required for reproducing the scene can be largely reduced. Take an extreme example: if the viewer has to stay at one certain position and view along one certain direction, an image or a video sequence captured at that position and direction is good enough for representing the scene. Another example is branch movies [75, 117, 89], in which segments of movies corresponding to different spatial navigation paths are concatenated together at selected branch points, and the user is forced to move along the paths but allowed to switch to a different path at the branch points.

2.2.1 Commonly used assumptions to constrain the viewing space

There is a common set of assumptions that people made for constraining the viewing space in IBR. Some of them are preferable, as they do not impact much on the viewers' experiences. Some others are more restrictive and used only when the storage size is a critical concern. We list them below roughly based on their restrictiveness.

[**Assumption 1**] As we are taking images of the scene for IBR, we may simplify the wavelength dimension into three channels, i.e., the red, green and blue channels. Each channel represents the integration of the plenoptic function over a certain wavelength range. This simplification can be carried out throughout the capturing and rendering of IBR without noticeable effects. Almost all the practical representations of IBR make this assumption.

[**Assumption 2**] The air is transparent and the radiances along a light ray through empty space remain constant. Under this assumption, we do not need to record the radiances of a light ray on different positions along its path, as they are all the same. To see how we can make use of this assumption, let us limit our interest to the light rays leaving the convex hull of a bounded scene (if the viewer is constrained in a bounded free-space region, the discussion hereafter still applies). Under Assumption 2, the plenoptic function can be represented by its values along an arbitrary surface surrounding the

scene. This reduces the dimension of the plenoptic function by one. The radiance of any light ray in the space can always be obtained by tracing it back to the selected surface[3]. In other words, Assumption 2 allows us to capture a scene at some places and render it at somewhere else. This assumption is also widely used, such as in [66, 37, 128].

[**Assumption 3**] The scene is static, thus the time dimension can be dropped. Although a dynamic scene includes much more information than a static one, there are practical concerns that restrict the popularity of dynamic IBR. One concern is the sample data size. We all know that if we capture a video for a scene instead of a single image, the amount of data may increase for about 2 or 3 orders of magnitude. It can be expected that dynamic IBR will have the same order of size increase from static IBR. Moreover, IBR often requires a large amount of capturing cameras. If we want to record a dynamic scene, all these cameras must be present and capturing video together. Unfortunately, today's practical systems cannot afford to have that many cameras. The known IBR camera array that has the largest number of cameras may be the Stanford light field video camera [149], which consists of 128 cameras. This is yet not enough for rendering high quality images. Capturing static scenes does not have the above problem, because one can always use the time axis to compensate for the lack of cameras. That is, images captured at different time and positions can be used together to render novel views.

[**Assumption 4**] In stead of moving in the 3D space, the viewer is constrained to be on a surface, e.g., the ground plane. The plenoptic function can then reduce one dimension, as the viewer's space location becomes 2D. Although restricting the viewer on a surface seems unpleasing, Assumption 4 is acceptable for two reasons. First, the eyes of human beings are usually at a certain height-level for walk-through applications. Second,

[3]Strictly speaking, a real camera has finite resolution. A pixel in an image is in fact an average of the light rays from a certain area on the scene surface. If we put two cameras on a line and capture the light ray along it, they may have different results, as their observing area size on the scene surface may be very different. Such resolution sensitivity was pointed out by Buehler et al. in [11].

human beings are less sensitive to vertical parallax and lighting changes because their two eyes are spaced horizontally. Example scenes using concentric mosaics [128] showed that strong effects of 3D motion and lighting change could still be achieved under this assumption.

[Assumption 5] The viewer moves along a certain path. That is, the viewer can move forward or backward along that path, but he/she cannot move off the path. Assumption 5 reduces two dimensions from the full plenoptic function. Branch movies [75, 117, 89] is an example that takes this assumption. This assumption is also reasonable for applications such as virtual touring, where the viewer follows a predefined path to view a large scene [56, 98].

[Assumption 6] The viewer has a fixed position. This is the most restrictive assumption, which reduces the dimension of the plenoptic function by three. No 3D effects can possibly be perceived under this assumption. Nevertheless, under this assumption the representations of IBR can be very compact and bear much similarity to regular images and videos. Capturing such representations is also straightforward. Thanks to these benefits, the QuickTime VR™ technology [18] based on Assumption 6 has become the most popular one among all the IBR approaches in practice.

There is one important thing to notice. That is, the dimension reduced by the above six assumptions may not be addable. In particular, Assumption 2 does not help further save dimension so long as one of the Assumption 4, 5 or 6 is made. This is because when the viewer's position has certain constraints, usually the sampled light ray space intersects each light ray only at a single point, which makes Assumption 2 not useful any more. In the next subsection, we will show a concrete example with concentric mosaics [128].

2.2.2 Various representations and their rendering process

By making the assumptions mentioned above, the 7D plenoptic function can be simplified to lower dimensional functions, from 6D to 2D. A quick summary of some popular

Table 2.1: IBR representations with various viewing space constraints.

Dimension	Example Representations	Assumptions
7D	Plenoptic function	No
6D	Surface plenoptic function	(2)
5D	Plenoptic modeling	(1,3)
	Light field video	(1,2)
4D	Light field/Lumigraph	(1,2,3)
3D	Concentric mosaics	(1,2,3,4)
	Panoramic video	(1,6) or (1,3,5)
	Branch movies	(1,3,5)
	Video	(1,6)
2D	Image mosaicing	(1,3,6)
	Image	(1,3,6)

representations is given in Table 2.1. We will explain these techniques in detail.

6D - The surface plenoptic function

The surface plenoptic function (SPF) was first introduced in [160]. It is simplified from the full 7D plenoptic function using Assumption 2. As we discussed, when radiance along a light ray through empty space remains constant, the plenoptic function can be represented by its values on any surface surrounding the scene. SPF chooses the surface as the scene surface itself. For regular scene surface with dimension two, the SPF is 6D: position on the surface (2D), light ray direction (2D), time (1D) and wavelength (1D). Although it is difficult to apply SPF for capturing real scenes due to unknown scene geometry, SPF was used in [160] for analyzing the Fourier spectrum of IBR representations (see more details in Chapter 4). The *surface light field* [90, 152] could be considered as dimension-reduced version of SPF.

Take anyone among Assumption 1, 3 and 4, we may also obtain a 6D representation of the scene. However, a 6D function is still too complicated for a practical IBR system to capture and render.

5D - Plenoptic modeling and light field video

By ignoring wavelength and time dimensions (Assumption 1 and 3), McMillan and Bishop [86] introduced plenoptic modeling, which is a 5D function:

$$l^{(5)}(V_x, V_y, V_z, \theta, \phi) \tag{2.2}$$

They record a static scene by positioning cameras in the 3D viewing space, each on a tripod capable of continuous panning. At each position, a cylindrical projected image was composed from the captured images during the panning. This forms a 5D IBR representation: 3D for the camera position, 2D for the cylindrical image. To render a novel view from the 5D representation, the close-by cylindrical projected images are warped to the viewing position based on their epipolar relationship and some visibility tests.

The light field video [149, 155] is another 5D representation based on Assumption 1 and 2. It is a straightforward extension of the 4D light field, which will be explained in detail later. Light field video captures dynamic scenes using a multi-camera array. Due to hardware constraints, the number of cameras in the array is very limited at the current stage (128 in [149] and 64 in [155]). Therefore, aliasing or ghosting effects are visible from the rendered videos.

From Table 2.1 it is clear that any IBR representation below 5D will make Assumption 1. The other assumptions are optional and can be chosen to generate new representations. For example, if we constrain the viewer to be on a surface (Assumption 4), we get another 5D representation. Although no work has been reported to take such a representation, it is obviously feasible. What we need to do is to put many cameras on the viewer's surface and capture video sequences. During the rendering, since we did not make Assumption 2, the rendering position is also restricted on that surface.

4D - Light field/Lumigraph

The most well-known 4D IBR representations are the light field [66] and the Lumigraph [37]. They both ignored the wavelength and time dimensions and assumed that radiance

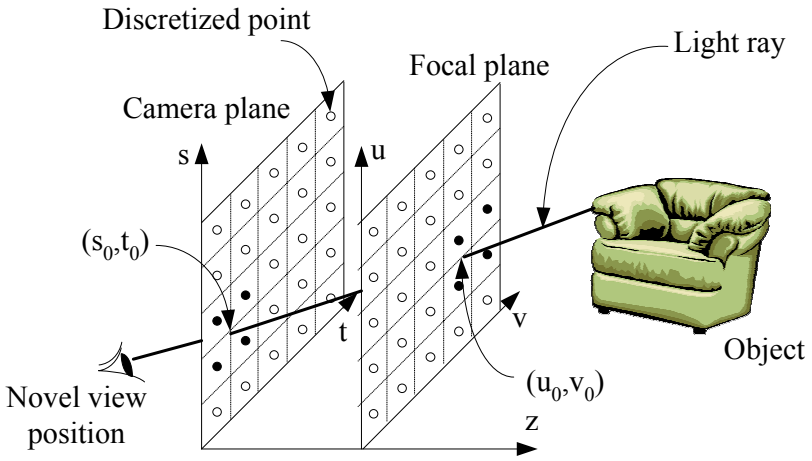


Figure 2.2: One parameterization of the light field.

does not change along a line in free space (Assumption 1, 2 and 3). However, parameterizing the space of oriented lines is still a tricky problem. The solutions they came out happened to be the same: light rays are recorded by their intersections with two planes. One of the planes is indexed with coordinate (u, v) and the other with coordinate (s, t), i.e.:

$$l^{(4)}(s, t, u, v). \tag{2.3}$$

In Figure 2.2, we show an example where the two planes, namely the camera plane and the focal plane, are parallel. This is the most widely used setup. An example light ray is shown and indexed as (s_0, t_0, u_0, v_0). The two planes are then discretized so that a finite number of light rays are recorded. If we connect all the discretized points from the focal plane to one discretized point on the camera plane, we get an image (2D array of light rays). Therefore, the 4D representation is also a 2D image array, as is shown in Figure 2.3. To create a new view of the object, we just split the view into its light rays, which are then calculated by quad-linearly interpolating existing nearby light rays in the image array. For example, the light ray in Figure 2.2 is interpolated from the 16 light rays connecting the solid discrete points on the two planes. The new view is then generated by reassembling the split rays together. Such rendering can be done in real time [65, 131] and is independent of the scene complexity.

We discuss briefly the difference between light field and Lumigraph. Light field assumes

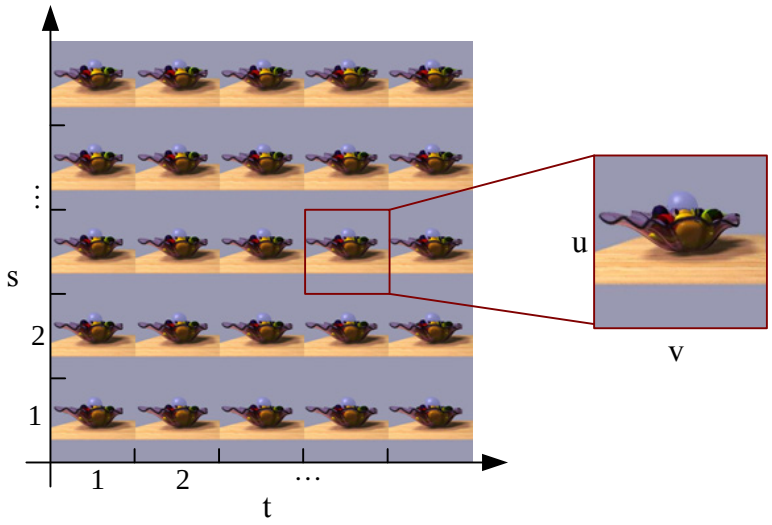


Figure 2.3: A sample light field image array: fruit plate.

no knowledge about the scene geometry. As a result, the number of sample images required in light field for capturing a normal scene is huge [16, 160]. To keep the amount of samples manageable, pre-filtering is applied during the capturing to reduce the light field signal's bandwidth [66]. On the other hand, Lumigraph reconstructs a rough geometry for the scene with an octree algorithm to facilitate the rendering with a small amount of images. Lumigraph also allows irregular sampling with a tracked hand-held camera. A hierarchical algorithm was proposed to resample the irregular samples onto the uniform grid on the camera and focal planes.

As we mentioned before, when Assumption 2 is made, the plenoptic function can be represented by its values on an arbitrary surface surrounding the scene. Often, that surface is where we put our capturing cameras. Light field and Lumigraph both choose this surface to be a box - each face of the box is the camera plane of the two-plane parameterization above. In the spherical light field [47, 15], a spherical surface was chosen for parameterization. Another interesting way to represent all the oriented lines in the space is the sphere-plane light field [15]. In this representation, a light ray is indexed by its direction (2D) and its crossing point (2D) with a plane perpendicular to its direction.

One thing to notice is that all the above representations are *structured* representations.

There were some papers that tried to analyze such line space structures [66, 14, 41] and claimed that one is better than the other [14]. Nevertheless, all the above representations share one common drawback: they do not match with practical image capturing. For instance, in light field although we may place cameras on the camera plane at the exact positions where discrete samples were taken, the pixel coordinates of the captured images cannot coincide with the focal plane samples. A sheared perspective projection was taken to compensate this [66]. In Lumigraph the images were taken with a hand-held camera so a resampling process was required any way. Spherical light field requires all the sample light rays passing through the corners of the subdivided sphere surface, which demands resampling for practical capturing. The sphere-plane light field does not have pencil (a set of rays passing through the same point in space [1]) in the representation so resampling is also needed. It would be attractive to store and render scenes from the captured images directly. In Section 2.3 we will discuss unstructured Lumigraph [11], which does not require the resampling.

Similar to the discussions in 5D, there are other possibilities to generate 4D IBR representations. For example, by making Assumption 1, 3 and 4, we may capture a static scene for a viewer to move smoothly on a surface [4, 18]. If we make Assumption 1 and 5, we may record a dynamic event and allow a viewer to move back and forth along a predefined path.

3D - Concentric mosaics and panoramic video

Other than the assumptions made in light field (Assumption 1, 2 and 3), concentric mosaics [128] further restricts that both the cameras and the viewers are on a plane (Assumption 4), which "reduces" the dimension of the plenoptic function to three. In concentric mosaics, the scene is captured by mounting a camera at the end of a level beam, and shooting images at regular intervals as the beam rotates, as is shown in Figure 2.4. The light rays are then indexed by the camera position or the beam rotation angle α, and the pixel locations (u, v):

$$l^{(3)}(\alpha, u, v). \tag{2.4}$$

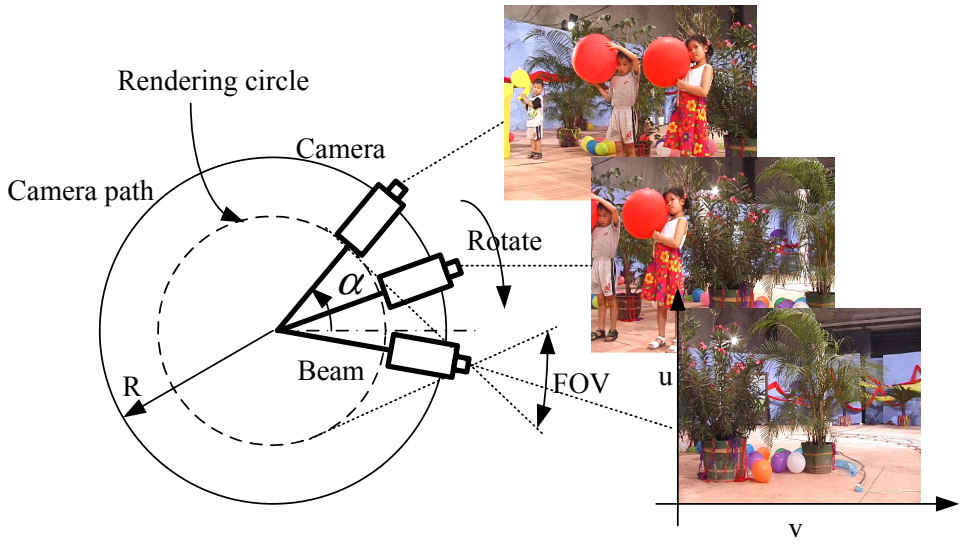


Figure 2.4: Concentric mosaics capturing.

This parameterization is equivalent to having many slit cameras rotating around a common center and taking images along the tangent direction. Each slit camera captures a manifold mosaic, inside which the pixels can be indexed by (α, u), thus the name concentric mosaics. During the rendering, the viewer may move freely inside a rendering circle (Figure 2.4) with radius $R\sin(FOV/2)$, where R is the camera path radius and FOV is the field of view of the cameras. The rendering of concentric mosaics is slit-based. The novel view is split into vertical slits. For each slit, the neighboring slits in the captured images are located and used for interpolation. The rendered view is then reassembled using these interpolated slits.

There is a severe problem with concentric mosaics - the vertical distortion. Unfortunately, no matter how dense we capture the scene on the camera path, vertical distortion cannot be eliminated. In the original concentric mosaics paper [128], depth correction was used to reduce the distortion. That is, we need to have some rough knowledge about the scene geometry. Ignoring how difficult it is to obtain the geometry information, recall in the last subsection that the dimension reduced by Assumption 2 and Assumption 4 are not addable, we realize that the light ray space concentric mosaics is capturing is in fact still 4D. Recording it with 3D data must require extra information for rendering, in this case, the scene geometry.

Figure 2.5: Parallax observed from concentric mosaics rendered scenes.

Despite the severe vertical distortion, concentric mosaics is still a success. Capturing concentric mosaics is very simple. The viewer may experience significant horizontal parallax and lighting changes, as shown in Figure 2.5. A similar work to concentric mosaics is [54], where the camera path is a 1D straight line. Such scheme can be considered as a simplified version of the light field. On the other hand, concentric mosaics can be easily boosted to 4D if we align a vertical array of cameras at the end of the beam [69].

Another popular 3D IBR representation is the panoramic video [18, 35, 98]. It can be used for either dynamic (fixed viewpoint, Assumption 1 and 6) or static scenes (Assumption 1, 3 and 5). Compared with regular video sequences, the field of view in panoramic video is often 360°, which allows the viewer to pan and zoom interactively. If the scene is static, the viewer can also move around [18, 56]. Capturing a panoramic video is an easy task. We simply capture a video sequence by a multi-camera system [35, 132], or an omnidirectional camera [97], or a camera with fisheye lens [154]. Rendering of panoramic video only involves a warping from cylindrical or spherical projected images to planar projected images. Due to the convenience of capturing and rendering, the acceptable perceptual quality and the affordable storage requirement, multi-panorama representations are adopted for several systems to capture large-scale scenes, such as in [55, 139]. Many commercial panoramic video systems are also available, such as iPIX immersive imaging from Internet Pictures Corp. [24], 360 One VR^{TM} from Kaidan [50], TotalView^{TM} from Be Here Technologies

Figure 2.6: A 360° cylindrical panorama of the Confucius Temple, Shandong, China.

[138], Ladybug^{TM} from Point Grey [40], among many others.

2D - Image mosaicing

Image mosaicing composes one single mosaic with multiple input images. The output mosaic is a 2D plenoptic function. Often such mosaic is composed for increasing the field of view of the camera, with early applications in aerial photography [42, 88] and cel animation [153]. Depending on the collection of the light rays recorded in the mosaic, image mosaicing techniques can be classified into two categories: single-center-of-projection mosaic or multiple-center-of-projection mosaic.

In most cases, the light rays recorded in the mosaic share the same center-of-projection (COP), which is called panoramic mosaic or panorama (Figure 2.6). The light rays are indexed by their directions, i.e.:

$$l^{(2)}(\theta, \phi). \tag{2.5}$$

Although a panorama can be easily obtained by hardware intensive systems [87, 97, 154], the focus of research is on how to construct spherical or cylindrical panoramas by stitching multiple input images together [18, 86, 119, 133]. Usually, the input images are taken from the same viewpoint and are related by 2D projective transforms. If the transforms are known in advance, images can be composed together easily [38]. Otherwise, a common technique is to establish at least four corresponding points across each image pair and find such transforms [39]. Other techniques for deriving these transforms without specific point correspondence have also been developed [133, 135]. One practical issue is that the input images may not strictly share the same COP, which causes certain ghosting artifacts in the resultant mosaic. Such artifacts can be partially eliminated through local alignment

algorithms [135, 144].

In the more general scenario, the cameras of the input images can move in free form and the resultant mosaic has multiple COPs. In contrast to the panoramic mosaic where light rays are indexed by their directions, multiple-COP mosaic often indexes the light rays by a certain surface or manifold, thus it is also called *manifold mosaic*. The direction of the light rays is often perpendicular or tangential to the manifold surface. Recall in concentric mosaics [128] the 3D parameterization is equivalent to having many slit cameras rotating around a common center and taking images along the tangent direction. Each slit camera captures a manifold mosaic, which can be indexed by points on a 2D cylindrical surface ((α, u) as in Figure 2.4). All the light rays captured are tangential to that surface. In [105, 106, 168], manifold mosaic is constructed by stitching slit images together, assuming the motion of the camera is slow. Effectively, the surface that is used for light ray parameterization has various forms such as a plane, a cylindrical or other general surfaces. If center slits of the captured images are used for stitching, as was suggested in [106], the indexed light rays will be roughly perpendicular to these manifolds. A particularly interesting mosaic is constructed when the camera has forward or backward motion. Pipe projection was used to construct the mosaic on a pipe surface [107].

The rendering of image mosaicing is very simple. For panoramic mosaic, we often perform a warping from the cylindrical or spherical projected mosaic to planar projected images, as what we have done for panoramic video. Such a warping is often unknown in a general manifold mosaic. Therefore regions of the mosaic may be used directly for rendering, as long as the field of view of the rendered image is small enough [153]. Notice that in both cases, the motion of the viewer is very restricted. In a panoramic mosaic the viewer can only change his/her view direction, while in a manifold mosaic the viewer can only move along a fixed surface/path and towards a fixed direction. It is possible to alleviate the constraints by capturing multiple mosaics. The QuickTime hopping [18] and the manifold hopping [129] are two such extensions for panoramic mosaic and manifold mosaic, respectively.

Other than increasing the field of view of the camera, image mosaicing can also be used to increase the image resolution, namely *super-resolution*. We refer the reader to [8] for a

Table 2.2: A quick summary of various approaches that introduce source descriptions.

Source description		**Representative reference**
Scene geometry	Correspondence	[19, 125, 62, 5], etc.
	Depth map	[127, 17, 113], etc.
	Mesh model etc.	[11, 146, 112], etc.
Texture map (+ scene geometry)		[27, 83], etc.
Reflection model (+ scene geometry)		[13, 152], etc.

[19]View interpolation; [125]View morphing; [62]Reference views; [5]Tensor space;
[127]Layered depth image (LDI); [17]LDI tree; [113]Multiple-center-of-projection images;
[11]Unstructured Lumigraph; [146]Spatial-temporal view interp.; [112]View dependent geometry;
[27]View dependent texture map; [83]Image-based visual hull;
[13]Reflection space IBR; [152]Surface light field;

review of this area.

2.3 Introducing Source Descriptions

Another strategy to make the image-based rendering data manageable is to introduce some source descriptions. Such descriptions can be the scene geometry, the texture map, the surface reflection model, etc. Among them, the scene geometry is the most widely used. Texture map and reflection model are often used as additional descriptions on top of the scene geometry. A quick summary of the different approaches is given in Table 2.2. Source descriptions can tell the correspondence between light rays, thus reduce the overall number of necessary light rays to be captured.

2.3.1 IBR with known scene geometry

Given the scene geometry, light rays from the same surface point can be identified. Since most scene surfaces are close to Lambertian, or at least locally color consistent (light rays from the same surface point share the same color if their reflection directions are similar) [158], geometry can save the number of light rays to be captured for a scene [16]. In fact, as was pointed out by many researchers, there is a geometry-image continuum in the

representations of scenes [52, 64]. The more we know about the scene geometry, the smaller the amount of images we need for good rendering.

The scene geometry can be described in different forms, such as correspondence between images (e.g., optical flow), dense depth map, volumetric or mesh model, etc. Here we classify the various approaches based on different geometry forms they take and present them one by one.

Correspondence between images

Any scene geometry information can be considered as knowledge about the correspondence between images. Here we specifically mean approaches that do not have an explicit geometry representation. Examples of such knowledge are point feature correspondences, disparity map, optical flow, etc. The idea is to find corresponding light rays in the captured image set for those in the novel view. In the photogrammetric community, such approaches are developed under the name of *transfer* methods.

Early work on this track was under the study of image morphing [6] and often involves certain manual help. For example, an animator needs to specify a set of feature correspondences, which form a control mesh. In [150], the novel view is generated by warping the control mesh through spline interpolation. A two-dimensional free-form deformation and Bézier Clipping was used to fulfill the same task in [99]. In [6], Beier and Neely defined a global transform/warping between the two images based on a set of matched line segments. For any view in between, the matched line segments in the novel view are first interpolated, which then determines the transform from one of the reference views to the novel view. A deformable surface model based morphing strategy that does not require the control mesh structure was also discussed in [63]. Recently, a feature-based light field morphing algorithm was proposed in [166].

View interpolation [19], proposed by Chen and Williams, eliminates the need of the human animator. Instead, the optical flow between the two images is assumed as known. To generate an in-between view of the input image pair, the offset vectors in the optical flow

are linearly interpolated and the pixels in the source images are moved by the interpolated vector to their destinations in the novel view. View interpolation performs very well if the two input images are close to each other, so that visibility ambiguity is not an issue. On the other hand, the interpolated views will be physically exact only if the camera motion is perpendicular to the camera viewing axis. In [147] a mathematical formulation was given to show the conditions when linear interpolation is physically correct.

In [125, 124], Seitz and Dyer proposed view morphing. View morphing guarantees that the rendered view is physically valid by introducing a prewarping stage and a postwarping stage. During the prewarping, the two reference images are rectified [43]. After the rectification, the two images share the same image plane and their motion becomes perpendicular to their viewing axis. Linear interpolation is then used to get the intermediate view, followed by postwarping to compensate the rectification effect on that view.

The novel view in view interpolation and view morphing are often in between the two reference images. Laveau and Faugeras [62] first proposed to make use of the epipolar constraints [43], which enabled extrapolation. The novel view is generated from a set of weakly or fully calibrated reference views. The viewpoint and the retinal plane of the novel view are specified by manually selected four corresponding points. A dense disparity map is also assumed to be available. To render the novel view, a ray-tracing like algorithm is implemented, which for each rendered light ray finds the corresponding light rays in the reference views through the epipolar constraint and the disparity map. Notice that when the reference views are weakly calibrated, only projective structure can be recovered [43], thus the resultant novel view may appear warped. Knowing the intrinsic parameters of the cameras (full calibration) will solve such problem.

In plenoptic modeling [86], a similar approach was proposed. The difference is that the reference view positions are known, and the reference views are now cylindrically projected panoramic views. Therefore, cylindrical epipolar constraints and dense angular disparity maps were used for novel view interpolation.

The epipolar constraint is between two images. For three images, there is another constraint represented by the trifocal tensor [43]. Given two views in correspondence and

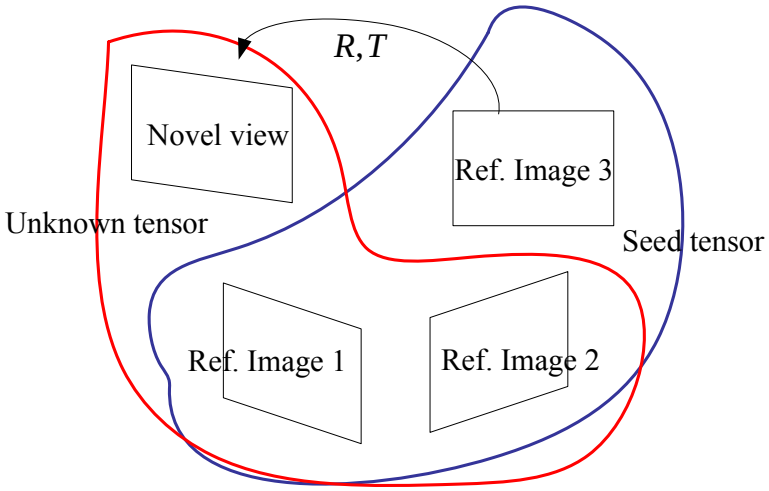


Figure 2.7: Obtain the tensor between a novel view and the two reference views from a seed tensor [5].

a tensor, the corresponding third view can be generated by a warping function. Avidan and Shashua proposed a view synthesis algorithm based on the above principle [5]. The key of their approach is the way to specify the tensor between two reference views and a novel view. As illustrated in Figure 2.7, given a seed tensor between the two reference views and an additional reference view (which could be a duplication of one of the two reference views), the unknown tensor could be obtained by knowing the rotation and translation between the third reference view and the novel view. Therefore specification of a novel view is more direct compared with the epipolar constraint based methods such as [62], where manual selection of matching points is needed. Moreover, Avidan and Shashua argued that trifocal tensor based method is often more stable than the epipolar constraint based ones under certain singular camera configurations (e.g., when the camera centers are collinear).

In a recent paper Lhuillier and Quan [67] presented an interpolation algorithm based on joint view triangulation. Starting from some points of interest selected automatically, they first grow the matching points to their neighborhoods. Planar patches are then fit locally for regularization or removing outliers assuming the matching is piecewise smooth. The two reference views are then triangulated jointly. Novel views are interpolated by warping the matched triangles. A walk-through system based on a similar framework was also developed

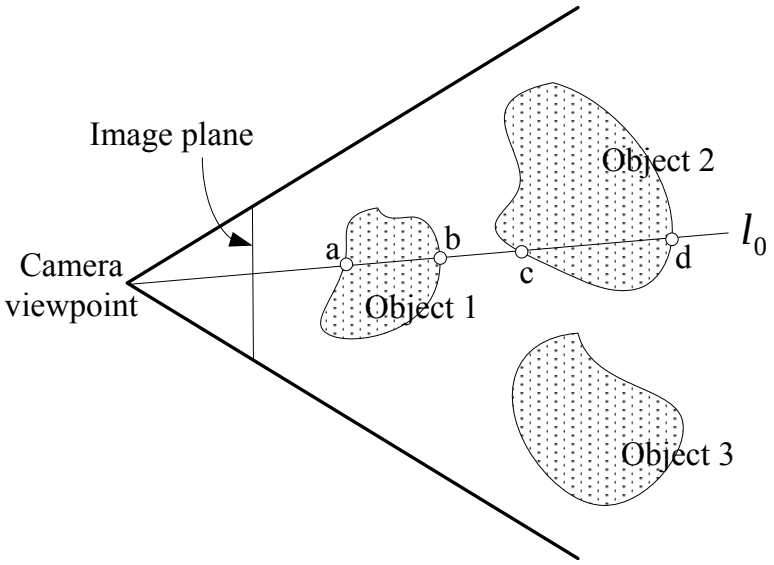


Figure 2.8: The layered depth image.

in [2].

Dense depth map

Another popular scene geometry representation is the dense depth map. It indicates the per-pixel depth values of the reference views. Such a depth map is easily available for synthetic scenes, and can be obtained for real scenes via a range finder.

The simplest IBR representation with a dense depth map is a set of images and their depth maps [85, 122]. An extension to the multiple-center-of-projection (MCOP) mosaic was given in [113], where again a depth value is attached to each pixel in the MCOP image. In [127], Shade et al. proposed the sprite with depth and the layered depth image (LDI). Sprite with depth keeps an out-of-plane displacement component at each pixel in the sprite, which resembles the above-mentioned representations. LDI is a view of the scene from a single input camera view, but with multiple pixels along each line of sight. Correspondingly, the depth map is also multi-valued for each pixel. This is shown in Figure 2.8. On the path of the light ray l_0, the depth and color value of point a, b, c and d are all recorded. Extensions to the LDI include the layered depth cube [76] and the LDI tree

[17].

The rendering algorithms of IBR representations with dense depth map are often similar to each other. In [85], a 3D warping algorithm was proposed to render novel views that are close to a reference view. The pixels of the reference view are first projected back to their 3D locations and then re-projected to the novel view. To speed up the above process, Oliveira and Bishop [101] proposed to factorize the warping process into a simple pre-warping stage followed by a standard texture mapping. The pre-warp handles only the parallax effects resulting from the depth map and the direction of view. The subsequent texture-mapping operation handles the scaling, rotation, and remaining perspective transformation, which can be accelerated by standard graphics hardware. A similar factoring algorithm was performed for the LDI [127], where the depth map is first warped to the output image with visibility check, and colors are pasted afterwards.

One major problem in the above rendering methods is that holes may occur in the rendered view due to undersampling or disocclusion (scene is occluded in the reference view but visible in the novel view). By introducing multiple depth values along a light ray, the disocclusion problem is partially solved in the LDI representation [127]. In [122], because multiple images are available for rendering, holes due to disocclusion are also not serious as long as the number of images is large enough. The undersampling problem can also be solved by taking more images. The LDI tree [17] is a modified LDI approach which combines multiple reference views in to a single hierarchical representation, which maintains the resolution of each reference view in the data structure. On the other hand, even if holes do happen, they may be removed through algorithms such as splatting [80, 127] or meshing [80, 113].

Mesh or volumetric model

Mesh model is the most widely used components in model-based rendering. Despite the difficulty to obtain such a model, if it is available in image-based rendering, we should make use of it to improve the rendering quality.

Buechler et al. proposed the unstructured Lumigraph rendering [11], which addressed the above rendering problem. They first proposed eight goals for IBR rendering: use of geometric proxies; unstructured input; epipole consistency; minimal angular deviation; continuity; resolution sensitivity; equivalent ray consistency and real-time. These goals served as the guidelines of their proposed unstructured Lumigraph rendering approach. Weighted light ray interpolation was used to obtain light rays in the novel view. The weights are largely determined by how good the reference light ray is to the interpolated one according to the goals. A clever weight blending field for the reference views is described to guarantee real-time rendering.

Another popular geometric representation is a volumetric model, which can be reconstructed from various algorithms [37, 146, 126, 32]. Although the volumetric model can be easily converted to a mesh model [78], sometimes it may be preferable to render with the volumetric model directly. The algorithm in [11] can be applied straightforwardly without any change. One concern about the volumetric model is that it has a finite resolution. To remove the granular effects in the rendered image due to finite resolution, in [146] a model smoothing algorithm was applied during the rendering, which greatly improved the resultant image quality.

Rademacher proposed an interesting approach called view dependent geometry [112]. Namely, the geometry used during the rendering may vary when the view position changes. Such approach is attractive for scenes where the geometry reconstruction algorithm can only obtain a model that is locally applicable, such as those obtained through stereo methods [120].

2.3.2 Texture map (+ scene geometry)

Texture map is one of the most widely used source descriptions in model-based rendering. As texture maps are often obtained from real objects, a geometric model with texture mapping can produce very realistic scenes.

In image based rendering, when the scene geometry is available, it is possible to generate

texture maps from the reference views. This has already been demonstrated in the 3D warping algorithm [85] for IBR representations with dense depth map mentioned before. Notice that in IBR we do not apply reflection models of the scene surface as we do in model-based rendering. A scene becomes Lambertian if both the geometry and the texture map are fixed. Such scenes may not be highly interesting. It is therefore natural to introduce texture maps that vary when the viewpoint changes, namely view dependent texture mapping (VDTM) [27].

In [27], Debevec et al. proposed to project the reference views onto the geometric model to form the texture map through a weighting scheme. The weights are determined by the angular deviation from the reference views to the virtual view to be rendered. Later a more efficient implementation of VDTM was proposed in [29], where the per-pixel weight calculation was replaced by a per-polygon search in a pre-computed lookup table. Note that VDTM is in fact a special case of the later proposed unstructured Lumigraph rendering [11].

The image-based visual hull (IBVH) algorithm [83] can be considered as another example of VDTM. In IBVH, the scene geometry was reconstructed through an image space visual hull [61] algorithm. A texture pixel was generated from the reference views by back projection using only the light ray with the smallest angular deviation. Such adaptation is partially due to the fact that only four cameras were used in IBVH.

2.3.3 Reflection models (+ scene geometry)

Other than the texture map, the appearance of an object is also determined by the interaction of the light sources in the environment and the surface reflection model. This becomes more obvious if the texture map is very simple (e.g., uniform color) and the object is highly specular, such as a simple mirror ball.

In image-based rendering, we often do not try to figure out what the scene object's reflection model is. Instead, we capture light rays that are reflected from the scene surface. Recall that such parameterization has been discussed before under the name surface plenoptic function [160]. The advantages of recording only the reflected light rays are numerous:

we do not need to derive the underlying surface reflection model any more; we do not need to model the complex light sources in a real environment; and we do not need to calculate the interaction between the light source and the reflection model. The downside is that the light source and the reflection model are now tightly coupled. Efforts need to be made for relighting the scene under different lighting conditions [151, 26].

In [13], Cabral et al. proposed reflection space image-based rendering. Reflection space IBR records the total reflected radiance for each possible surface direction. Note the difference between such a radiance environment map and the traditional environment map, where the incoming radiance is stored [7, 39]. The proposed radiance environment map is viewpoint dependent, thus a set of such maps are pre-computed before rendering, as the multiple images we often have in normal IBR representations. During the rendering, the radiance environment map is first interpolated/warped to the desired viewpoint and then used for novel view generation. An interesting application of radiance environment map is given in [25], where synthetic objects are rendered into real scenes. A probing mirror ball is used to obtain the radiance environment map at the position the synthetic objects are located. A differential rendering technique allows for good results to be obtained when only an estimate of the local scene reflectance properties is known.

The above method assumes that if two surface points share the same surface direction, they have the same reflection pattern. This might not be true due to multiple reasons such as inter-reflections. A more general approach is to really capture the scene reflections at arbitrary surface points as in the surface plenoptic function [160]. By ignoring the time and the wavelength dimensions, Wood et al. proposed surface light field [152]. They first obtained a base mesh of the scene object through a range scanner. For points on the base mesh, they obtained the reflections along different directions by capturing hundreds of images of the scene. A pointwise fairing algorithm was proposed to resample the irregular sample light rays into a reflection map or lumisphere with a piecewise linear model. Notice that these lumispheres may have missing data as only light rays reflected to the outside of the object can be captured. Rendering surface light field is as straightforward as tracing each rendered light ray onto the geometric model and obtain its radiance. A more compact

representation of surface light field suitable for an accelerated graphics pipeline was recently proposed in [20]. A surface light field created on the surface of a visual hull rather than the true scene geometry is discussed in [84]. As we mentioned earlier in Section 2.2.1, under Assumption 2 (the radiance of a light ray does not change along its path in empty space), using the visual hull surface for recording the light rays is equivalent to using the true scene geometry as long as the viewpoint is outside the visual hull.

As mentioned before, the surface plenoptic function captures the scene only at a fixed lighting condition. Recently there has been some work on the relighting of IBR, such as the human face reflectance field [26], the plenoptic illumination function [151] and the 4D incident light fields [81]. These approaches share similar ideas. Images of the scene under different point light source or directional light source are first captured. These images can then be superimposed to render scenes under a much more complex lighting environment. Such operation can be performed to live-action scenes in real time [28].

2.3.4 Reconstructing source descriptions for IBR

In the above discussions, most IBR approaches assumed known source descriptions such as a known scene geometric model. In practice, acquiring such source descriptions is not a trivial task unless the scene is synthetic or a range finder is on hand. We may resort to computer vision techniques to reconstruct the source descriptions based on the captured images. Such techniques are called image-based modeling (IBM). Due to the close coupling of image-based rendering and image-based modeling, we see a clear convergence of the graphics community and the vision community [64]. Since a survey of the various IBM techniques is out of the scope of this dissertation, we refer the reader to the survey paper by Zhang [167] and by Oliveira [100] for more information.

There is subtle difference in the purpose of source description reconstruction in IBR and IBM. In IBM, the source description is the goal. The effectiveness of an IBM algorithm largely depends on the accuracy of the reconstructed source description. In contrast, the goal of IBR is rendering. While a highly complicated and accurate IBM algorithm can

certainly serve well during the preprocessing stage of IBR, IBR does not necessarily need a high quality source description. As pointed out in [52], the errors in the source description is less visible during local viewpoint perturbation, which is a unique characteristic of IBR.

In certain applications, a complicated IBM algorithm is simply not affordable. For instance, recently there has been increasing interest in capturing and rendering dynamic IBR scenes, as it provides rich new experiences and has wide applications in immersive TV, remote education, teleconferencing, games etc. Dynamic IBR has to be captured by a camera array. Due to cost constraints, existing camera arrays are not dense enough to perform view interpolation without scene geometry [155]. Therefore, one has to reconstruct scene geometry from the captured images. The dynamic property of the scene also forbids any complicated IBM algorithms. In the following text, we review several algorithms designed for dynamic IBR applications, which may not be covered in a normal IBM survey paper.

Depth from stereo (binocular or trinocular) is an attractive candidate for geometry reconstruction for IBR in real-time [45, 94]. Schirmacher et al. [123] built a 6-camera system which was composed of 3 stereo pairs. Each stereo pair is connected to a computer for calculating a dense depth map at 1-2 frames per second (fps)(this speed should be easily boosted with more powerful computer and better implementation of the depth-from-stereo algorithm). Naemura et al. [95] constructed a camera array system consisting of 16 cameras. A single depth map was reconstructed in real-time from 9 of the 16 images using a stereo matching PCI board. The rendering process of the above systems resembles those discussed in Section 2.3.1. We notice that stereo algorithms are not very suitable for large camera arrays if dense depth maps have to be reconstructed for every pair of images. However, view-dependent stereo [130, 156] is a good candidate, as will be discussed later.

Matusik et al. [83] proposed image-based visual hull (IBVH), which rendered dynamic scenes in real-time from 4 cameras. IBVH is a clever algorithm which computes and shades the visual hull of the scene without having an explicit visual hull model. The visual hull is calculated from the virtual viewpoint, thus IBVH has view-dependent geometry. The computational cost is low thanks to an efficient pixel traversing scheme, which can be implemented with software only. Another similar work is the polyhedral visual hull [82],

which computes an exact polyhedral representation for the visual hull directly from the silhouettes. Lok [77] and Li et al. [70] proposed to reconstruct the visual hull on modern graphics hardware with a volumetric representation and an image-based representation, respectively. One common issue of visual hull based rendering algorithms is that they cannot handle concave objects, which makes some close-up views of concave objects unsatisfactory. An improvement was made by the image-based photo hull (IBPH) [130] approach. IBPH utilizes the color information of the images to identify scene geometry, which results in more accurately reconstructed geometry. Visibility was considered in IBPH by intersecting the visual hull geometry with the projected line segments of the considered light ray in the nearby views.

Recently, Yang et al. [156] proposed a real-time consensus-based scene reconstruction method using commodity graphics hardware. Their algorithm utilized the Register Combiner for color consistency verification (CCV) with the sum-of-square-difference (SSD) measure, and obtained a per-pixel depth map in real-time. Both concave and convex objects of generic scenes could be rendered with their algorithm. In Chapter 8 of this thesis, we also present a similar view-dependent geometry reconstruction algorithm, which is fast enough to be implemented in software. The key of our method is to use an adaptive mesh to represent the scene geometry, which greatly reduces the computational cost. Compared with hardware implementations, our algorithm is more flexible in applying different CCV measures, handling lens distortions, etc. Please refer to Chapter 8 for more details.

2.4 Summary

In this chapter we surveyed various techniques of image-based rendering. We found that all the IBR representations are originated from the 7D plenoptic function, which describes the appearance of the world. As the 7D plenoptic function has too much data to handle, various approaches were proposed to reduce the data size while still giving the viewer a good browsing experience. Two major strategies were identified: constraining the viewing space and introducing source descriptions. We have presented many IBR representations

based on such a categorization.

Image-based rendering often involves huge amount of image data. By choosing more restrictive representations one can always reduce such data amount. On the other hand, it is important to know, given certain representations, what is the minimum amount of data needed. In the next section, we review existing methods from this aspect.

Chapter 3

Previous Work on IBR Sampling

In this chapter, we review the literature on IBR sampling. We ask the following question: knowing what representation we will use for the scene, how many images or light rays do we have to capture/store in order to render nice-looking images? Being able to answer such sampling question is essential in providing guidance to various IBR representations.

Unfortunately, IBR sampling is a very difficult problem, as the plenoptic function is such a high dimensional signal. Obviously, the sampling rate will be determined by the scene geometry, the texture on the scene surface, the reflection property of the scene surface, the motion of the scene objects, the specific IBR representation we take, the capturing and the rendering cameras' resolution, etc. Over-sampling was widely adopted in the early stages [66, 128], as no solution to the sampling problem was available. To reduce the huge amount of data recorded due to over-sampling, people have resorted to various compression schemes to save the storage space, which was surveyed in detail in [161]. These compression techniques viewed IBR data as video sequences or image arraies, and utilized existing image compression or video compression methods to reduce their size. In comparison, sampling is more of a fundamental problem to IBR. Solving the IBR sampling problem may provide a unique and deep-grounded view of IBR from signal processing's viewpoint.

Depending on how the capturing cameras are placed, IBR sampling can be classified into two categories: uniform sampling and nonuniform sampling. In uniform sampling, the cameras are positioned evenly on a surface or a line, following a regular pattern. Examples of such IBR representations are the light field [66] and the concentric mosaics [128]. The

main research topic is to find the minimum sampling rate or largest spacing between cameras such that one can achieve perfect reconstruction of the continuous plenoptic function. In nonuniform sampling, the cameras are often allowed to move freely along a surface [121, 158] or a line [159]. The goal of non-uniform sampling analysis is to use the minimum number of cameras and render the highest quality scene by arranging these cameras intelligently.

The literature of IBR sampling is surprisingly poor due to the difficulty of the problem. We next review them in two sections for uniform and nonuniform sampling respectively.

3.1 Uniform Sampling

The uniform sampling analysis of IBR generally follows the classic sampling theorem [102, 31], as IBR is essentially a sampled signal from the 7D plenoptic function. One may first find the Fourier transform of the plenoptic function and then sample it according to its spectrum bandwidth. Nevertheless, although performing the Fourier transform of the 7D plenoptic function is possible in theory, in practice we have to reduce the dimension of the signal. Again we check the assumptions discussed in Section 2.2.1 and see how they may affect our sampling analysis.

Based on Assumption 1, the wavelength dimension can be ignored in most IBR applications. Thus sampling along the wavelength axis can also be ignored. Assumption 2 claims that the radiance of a light ray along its path remains constant in empty space. This means that along the light ray path, one sample is good enough for perfect reconstruction. In reality, although the resolution of real cameras is finite and Assumption 2 may not be strictly valid [11], the sampling along the light ray path is still less interesting because the variation of the radiance is often too slow. Assumption 3 said that if necessary, the time dimension could also be ignored. In practice, even if we are capturing a dynamic scene, sampling on the time axis is often determined by the camera's frame rate and the property of the human eyes' temporal perception [23]. Due to the above reasons, most IBR sampling work in the literature [71, 16, 79] was for the light field and concentric mosaics.

The earliest IBR sampling work was by Lin and Shum [71]. They performed sampling

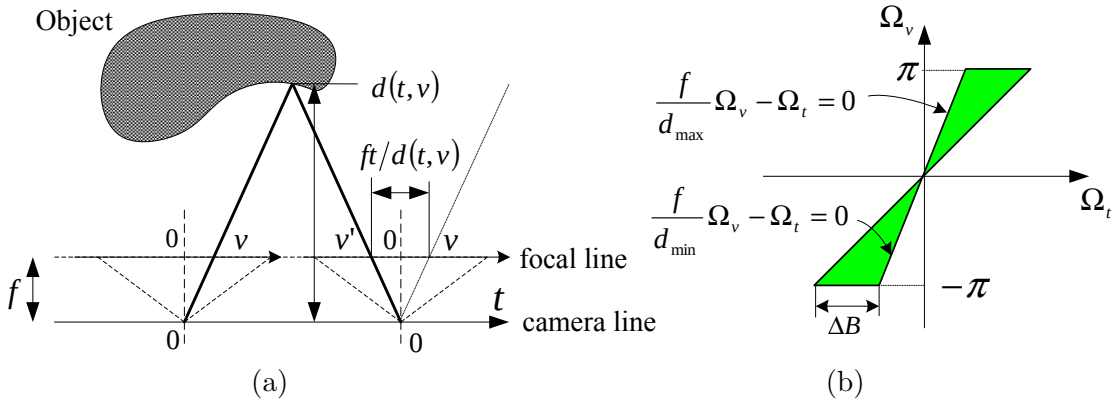


Figure 3.1: The plenoptic sampling analysis in [16]. (a) The light ray correspondence in a 2D light field; (b) the spectrum of light field.

analysis on both light field and concentric mosaics with the scale-space theory. The world is modeled by a single point sitting at a certain distance to the cameras. Assuming using constant depth and bilinear interpolation during the rendering, the bounds are derived from the aspect of geometry and based on the goal that no "spurious detail" should be generated during the rendering (referred as the causality requirement). Although the viewpoint of their analysis is rather interesting, this method is constrained by the simple world model they chose. The texture and the reflection model of the scene surface and occlusions are hard to analyze with such a method.

In [16], Chai et al. first proposed to perform the light field sampling analysis in the classic framework, i.e., applying Fourier transform to the light field signal, and then sampling it based on its spectrum. Assuming Lambertian surface and no occlusions, they found that the light rays represented by the plenoptic function have certain correspondence among themselves. For illustration purpose, we show a simplified 2D light field in Figure 3.1(a). Notice that the camera plane and focal plane in Figure 2.2 degenerate to lines in 2D. A local discretization of the focal line was adopted in their analysis.

We may easily see the following relationship from Figure 3.1(a):

$$l^{(2)}(t,v) = l^{(2)}\left[0, v + \frac{ft}{d(t,v)}\right] \tag{3.1}$$

where f is the focal length and $d(t,v)$ is the scene depth of the light ray (t,v). When the

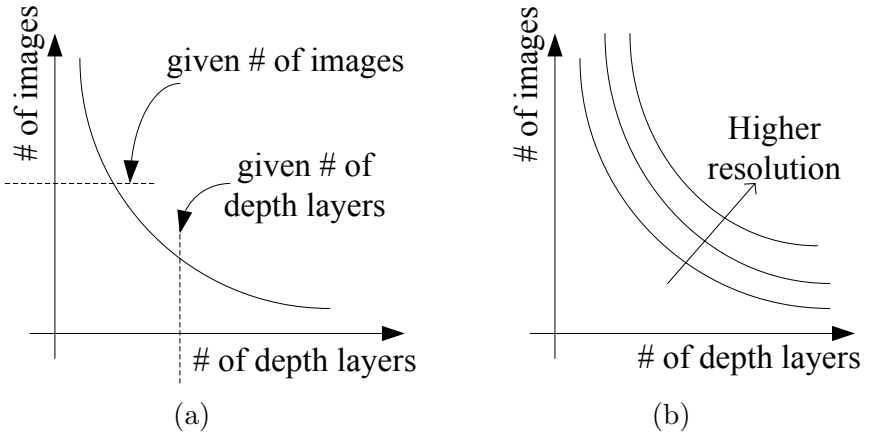


Figure 3.2: The minimum sampling curve in [16]. (a) One sampling curve representing the tradeoff between images and geometry; (b) the minimum sampling curve with respect to the capturing and rendering resolution.

scene is at constant depth $d(t, v) = d_0$, its Fourier transform can be written as:

$$L^{(2)}(\Omega_t, \Omega_v) = L'(\Omega_v)\delta\Big(\frac{f}{d_0}\Omega_v - \Omega_t\Big) \tag{3.2}$$

where $L'(\Omega_v)$ is the Fourier transform of $l^{(2)}(0, v)$ (with some scalar factor) and $\delta(\cdot)$ is the 1D Dirac delta function. Obviously the spectrum has non-zero values only along a line. When the scene depth is varying between a certain range, a "truncating windows" analysis was given in the paper, which concludes that the spectral support of a light field signal is bounded by the minimum and maximum depths of objects in the scene only, no matter how complicated the scene is (Figure 3.1(b)). Such analysis provides a fairly good first-order approximation of the spectrum analysis of IBR. However, the dependency on mapping images captured at arbitrary position to that at the origin prevents it from being applied to more complicated scenes such as non-Lambertian surface, scenes with occlusions and other IBR methods such as concentric mosaics.

Another contribution of [16] is the minimum sampling curve in the joint image and geometry space for light field. Since a Lambertian scene at constant depth corresponds to a tilted line in the frequency domain, if the scene geometry is represented via dense depth map and each depth value has a finite precision (a certain number of bits), we may divide the scene into multiple layers based on the depth values. If occlusions between layers

are ignored, each layer can be sampled and rendered independently. This is equivalent to having many scenes with much smaller depth variation, which reduces the number of images required. An example minimum sampling curve is shown in Figure 3.2(a). Based on this curve, given the number of depth layers, we may tell the minimum number of images needed. On the other hand, given the number of images, we can also tell how many depth layers we need. The minimum sampling curve is also related with the capturing and rendering resolution. The higher the resolution, the more images or depth layers are required, as shown in Figure 3.2(b).

Marchand-Maillet [79] performed Fourier analysis for scenes with functional surfaces. Instead of mapping all the images into one, they fixed the light ray direction and tried to find a one-to-one mapping from points on the scene surface to the camera plane. This one-to-one mapping is valid when no occlusion occurs. They showed that even when there is no occlusion, a band-limited signal pasted on a functional surface will not result in a band-limited light field spectrum.

In Chapter 4 of this thesis, we will provide a new spectral analysis framework based on the surface plenoptic function [160]. Our analysis is very general and can be applied to obtain the spectrum of IBR scenes even when they are non-Lambertian or occluded. Moreover, the same methodology is applicable for concentric mosaics. "Truncating windows" analysis can also be used there when the scene geometry is not available.

In Chapter 5, we propose to apply the generalized sampling theory [31] on IBR sampling after obtaining the spectrum of the IBR representation. Such method can in theory increase the IBR sampling efficiency by 50%.

3.2 Nonuniform Sampling

The uniform sampling analysis provides an indepth view of how a scene should be captured. For instance, in Chapter 4 we will show that a scene with non-Lambertian surface or occlusions generally needs more samples. However, an implicit assumption made by the uniform sampling analysis is that the scene plenoptic function is stationary. In practice, objects in

the scene have varying surface properties, and the plenoptic function is non-stationary. It is therefore natural to consider nonuniform sampling for the plenoptic function. One thing to notice is, in uniform sampling analysis, since the sampling is periodic, we only need to tell how many images/light rays are needed for perfect reconstruction of the scene; in nonuniform sampling, however, we need to answer not only how many images are needed but also where to place these cameras.

Fleishman et al. [34] proposed an automatic camera placement algorithm for IBR. A mesh model of the scene is known. The goal is to place the cameras optimally such that the captured images can form the best texture map for the mesh model. They found that such problem can be regarded as a 3D art gallery problem, which is NP-hard [103]. They then proposed an approximation solution for the problem by testing a large set of camera positions and selecting the ones with higher gain rank. Here the gain was defined based on the portion of the image that can be used for the texture map. A similar approach was proposed in [148], where the set of reference views were selected from a large image pool in order to minimize a certain target function.

Schirmacher et al. [121] proposed an adaptive acquisition scheme for a light field setup. Assuming the scene geometry is known, they added cameras recursively on the camera plane by predicting the potential improvement in rendering quality when adding a certain view. This a-priori error estimator accounts for both visibility problems and illumination effects such as specular highlights to some extent.

Starting from Chapter 6 of this thesis, we propose a general framework for nonuniform sampling, namely freeform sampling and active sampling. We apply the framework to IBR, and show that nonuniform sampling is indeed better than uniform sampling in terms of the worst case rendering quality.

3.3 Summary

The IBR sampling problem is an important one for providing an insight view of IBR. Due to the high dimensionality of the plenoptic function, the IBR sampling problem is highly

complicated and not well studied in the literature. This thesis is about to address some of the issues remained in the literature. For instance, Chapter 4 will present a method for the spectral analysis of non-Lambertian or occluded scenes; Chapter 5 will discuss the most efficient packing of the IBR spectrum; Chapter 6 will provide a general framework for the nonuniform sampling analysis of IBR. Such framework is used in Chapter 7 for the sampling problem of IBR. Chapter 8 describes a mobile camera array which implements nonuniform sampling for real-world scenes.

Chapter 4

Uniform Sampling: Spectral Analysis of IBR Data

Despite the previous work in [16], the spectral analysis problem for IBR has not been completely solved, in particular for non-Lambertian and occluded scenes. In this chapter, we present a new method to parameterize the problem, which is applicable for general-purpose IBR spectral analysis. We notice that any plenoptic function is generated by light rays emitted/reflected/refracted from the object surface. We introduce the surface plenoptic function (SPF), which represents the light rays starting from the object surface. Given that radiance along a light ray does not change unless the light ray is blocked (Assumption 2 in Section 2.2.1), SPF reduces the dimension of the original plenoptic function to 6D. We are then able to map or transform the SPF to IBR representations captured along any camera trajectory. Assuming some properties on the SPF, we can analyze the properties of IBR for generic scenes such as scenes with Lambertian or non-Lambertian surfaces and scenes with or without occlusions, and for different sampling strategies such as lightfield/concentric mosaics. We find that in most cases, even though the SPF may be band-limited, the frequency spectrum of IBR is not band-limited. We show that non-Lambertian reflections, depth variations and occlusions can all broaden the spectrum, with the latter two being more significant.

SPF is defined for scenes with known geometry. When the geometry is unknown, spectral analysis is still possible. We show that with the "truncating windows" analysis and

some conclusions obtained with SPF, the spectrum expansion caused by non-Lambertian reflections and occlusions can be quantatively estimated, even when the scene geometry is not explicitly known.

4.1 The Surface Plenoptic Function

Let us first have a close look at how plenoptic function is generated. Obviously, any light ray in the free space has a source. It can be either emitted from some light source (e.g., the Sun), or reflected by some object surface. If the object is transparent, refraction is also involved. Let the entire surface of all the light sources and objects be S. We can always trace a light ray in the free space back to a point on S. Assume the radiance does not change along a line unless the light ray is blocked, the 7D plenoptic function can be reparameterized to 6D including time (1D), wavelength (1D), point on the surface S (2D), and azimuth and elevation angles (2D) the light ray is emitted. We name this 6D function as the *surface plenoptic function* (SPF).

SPF bears close relationship to the surface light field proposed in [90, 152], but it is more general than surface light field because it may include the time and wavelength information. Wood et al. [152] focused on the representation, rendering and compression of surface light field, while we use SPF for sampling analysis. Wong's plenoptic illumination function [151] and Lin's reflected irradiance field [72] are also similar to SPF in the sense that they tried to model different light ray radiances from same surface points. However, their work studied scenes under different illuminations (fixed viewpoint), while we consider scenes under different viewpoints (fixed illumination). The sampling of reflected irradiance field [72] is independent of the scene geometry, because the viewpoint is fixed and the correspondence between light rays from the same surface point is known (always at the same pixel location). In contrast, for IBR sampling analysis we need some geometry information in order to know the light ray correspondences, such as the constant depth assumption made in [16].

Since light rays start from the object surface and end at capturing cameras, there exists

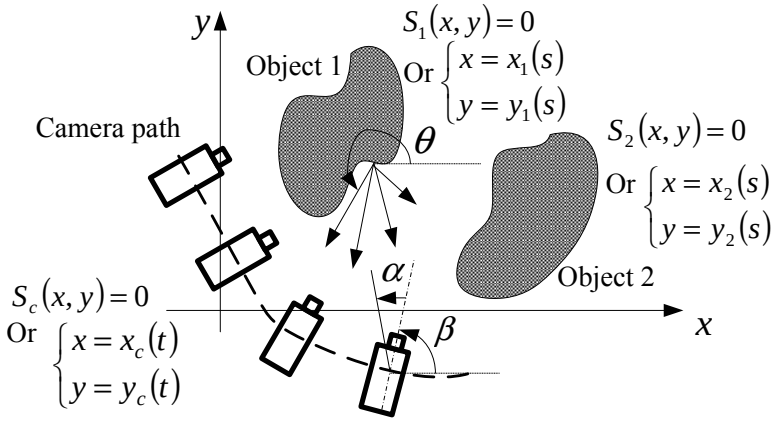


Figure 4.1: The 2D surface plenoptic function and general IBR capturing.

an onto mapping from the SPF to the various IBR representations. Such a mapping depends on both the scene geometry and the camera surface, but not the surface property such as the bidirectional reflection distribution function (BRDF). If we have some knowledge about the scene, in other words, if we know some property about the SPF, related property can be derived for the IBR representation. More importantly, the mapping does not require the scene to have no occlusions or Lambertian surface, which is very attractive for IBR spectral analysis.

Without loss of generality, we use the 2D world as example throughout this chapter for conciseness. The conclusions drawn here are easy to extend to the 3D world. In the 2D world, surface of objects/light sources is described with curves. Ignoring time and wavelength, the SPF is 2D: one dimension for describing a point on a curve, the other for illustrating the direction of the light ray emitted/reflected/refracted. An example scene is shown in Figure 4.1. The surface can be represented by:

$$S_i(x,y) = 0 \text{ or } \begin{cases} x = x_i(s) \\ y = y_i(s) \end{cases} \tag{4.1}$$

where s can be the arc length, i be the index for different objects. For a certain object i, define its SPF as:

$$l_i(s,\theta) \text{ on the curve } \begin{cases} x = x_i(s) \\ y = y_i(s) \end{cases} \tag{4.2}$$

where $0 \leq \theta < 2\pi$ is the direction of the light ray. $l_i(s,\theta)$ is the radiance of the light ray

that can be traced back to the surface point determined by s on object i. Notice that the above function does not appear to be related with what people often use for calculating lightings, such as surface normal, BRDF etc. We intend to do so because it is often too complicated if we try to model how the light rays are generated, in addition to the fact that such a model does not always exist for real scenes. Therefore, we only consider the resultant lighting effects in 4.2, and assume that we have some knowledge about the SPF. If we are able to model the lighting very well, the following analysis can still apply after calculating the lighting effects based on the known model.

Lambertian is the first assumption we could make for the scene, since it has been exclusively used in the IBR sampling literature. In terms of SPF, Lambertian property gives the following relationship:

$$l_i(s, \theta) = l_{is}(s) \tag{4.3}$$

and its Fourier transform is:

$$L_i(\Omega_s, \Omega_\theta) = L_{is}(\Omega_s)\delta(\Omega_\theta) \tag{4.4}$$

where $L_i(\Omega_s, \Omega_\theta)$ is the Fourier transform of $l_i(s, \theta)$, $L_{is}(\Omega_s)$ is the Fourier transform of $l_{is}(s)$.[1]

In the real world, pure Lambertian objects are rare. Highly reflective surface (like a mirror) is infrequent, too. In most cases, light rays from the same point on the object surface tend to be similar and have slow changes with respect to their angles. It is therefore reasonable to assume that $L_i(\Omega_s, \Omega_\theta)$ can be approximated by a band-limited signal. That is:

$$L_i(\Omega_s, \Omega_\theta) \approx L_i(\Omega_s, \Omega_\theta) I_{B_i}(\Omega_\theta) \tag{4.5}$$

where $I_{B_i}(\Omega_\theta)$ is the indicator function over Ω_θ, which is defined as:

$$I_{B_i}(\Omega_\theta) = \begin{cases} 1, & \text{if } |\Omega_\theta| < B_i; \\ 0, & \text{otherwise.} \end{cases} \tag{4.6}$$

[1]Strictly speaking, at a certain point on the surface, $l_i(s, \theta)$ need to be truncated on θ, because we can only observe light rays that come out of the object. Therefore, along Ω_θ $L_i(\Omega_s, \Omega_\theta)$ cannot be a delta function, nor can it be band-limited. However, we assume that this windowing artifact is negligible.

Here B_i defines the bandwidth for object i. Our band-limitness assumption can be connected to the band-limitness of the surface BRDF with the signal-processing framework for inverse rendering recently presented by Ramamoorthi and Hanrahan [114]. For points on a reflective surface, the outgoing light can be described as the convolution of the incoming light and the surface BRDF. If the incoming light is far (thus the incoming light can be considered as a delta function with respect to the angle), as long as the BRDF is band-limited, the outgoing light rays will be band-limited.

In order to capture the plenoptic function or surface plenoptic function, existing IBR approaches align cameras on a path/surface and take images for the scene. For example, cameras are placed on a plane in light field, and on a circle in concentric mosaics. In the 2D world, 2D light field has cameras on a line, while 2D concentric mosaics has cameras on a circle. In general, the cameras can be put along an arbitrary curve, as is shown in Figure 4.1. Let the camera path be:

$$S_c(x, y) = 0 \text{ or } \begin{cases} x = x_c(t) \\ y = y_c(t) \end{cases} \tag{4.7}$$

where t is the arc length. Assume that the camera path curve is differentiable, and the optical axes of our cameras are identical to the normals of the camera path. That is, at arc length t, the optical axis has direction $(-y_c'(t), x_c'(t))$, where $x_c'(t)$ and $y_c'(t)$ are the first order derivatives. Denote the direction of the optical axis with angle β, then:

$$\tan(\beta) = -\frac{x_c'(t)}{y_c'(t)} \tag{4.8}$$

The image pixels can be indexed by the angle between the captured light ray and the optical axis, as is represented by α in Figure 4.1. Denote the radiance of the light ray captured at arc length t, angle α as $l_c(t, \alpha)$. The goal of spectral analysis for this specific IBR representation is to find the Fourier transform of $l_c(t, \alpha)$, denoted as $L_c(\Omega_t, \Omega_\alpha)$, so that we can determine the minimum number of images we have to capture. Given the knowledge we have on the SPF $l_i(s, \theta)$ and its Fourier transform $L_i(\Omega_s, \Omega_\theta)$, the strategy is to associate $l_c(t, \alpha)$ with $l_i(s, \theta)$ and hope that we can represent $L_c(\Omega_t, \Omega_\alpha)$ in terms of $L_i(\Omega_s, \Omega_\theta)$. In Section 4.2 and 4.4, we will show examples where we can apply the above strategy to light field and concentric mosaics, respectively.

Before moving on to the next section, it is necessary to compare our work to the work by Marchand-Maillet in [79]. Being independently developed, they share some similar ideas. For example, both of them try to find a mapping between the captured light field and the light rays from the object surface. However, our approach is more general than that in [79]. We employ the concept of surface plenoptic function, which allows us to perform a better analysis on general IBR representations, as long as we can represent $L_c(\Omega_t, \Omega_\alpha)$ in terms of $L_i(\Omega_s, \Omega_\theta)$. This includes light field and concentric mosaics. In addition, occlusions and non-Lambertian surface effects can be better analyzed within our framework. Another work that might be related is shape from texture (SFT) [36]. Both SFT and our approach are about transforming some spectral property from a certain object surface to the captured images. However, IBR spectral analysis is significantly different from shape-from texture. First, the goal of SFT is to reconstruct the scene geometry from texture information, while we try to find the Fourier spectrum of the IBR representation given the geometry. Second, SFT often recovers geometry from a single image (although it can be extended to multiple views), while we deal with many images. Third, SFT only considers texture spectrum (e.g., isotropy or homogeneity), while we consider both texture and light ray radiance change along different directions, including non-Lambertian surface and occlusions.

4.2 Analysis for the Light Field

As shown in Figure 4.2, 2D light field is parameterized by two parallel lines, indexed by t and v, respectively. The t line is the camera line, while the v line is the focal line. The distance between the two lines is f, which is the focal length of the cameras. It is easy to show that a light ray indexed by pair (t, v) satisfies the following algebraic equation:

$$fx - vy - ft = 0. \tag{4.9}$$

Notice that the focal line is indexed locally with respect to where the camera is.

The relationship between the light field $l_c(t, v)$ and the SPF $l_i(s, \theta)$ is as follows. For the same light ray emitted/reflected/refracted from a surface point, it must be captured at

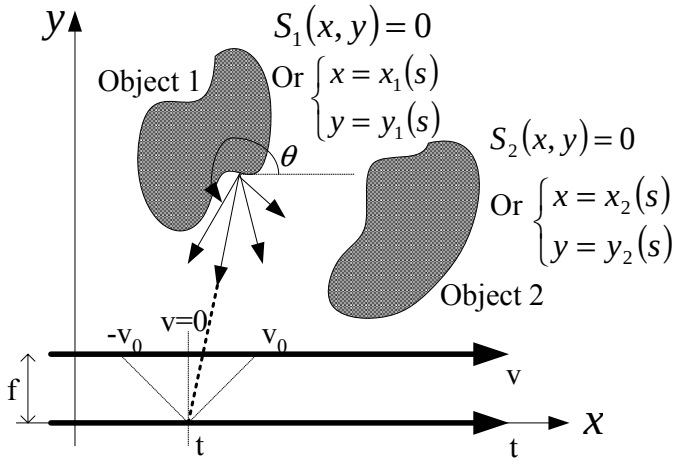


Figure 4.2: The light field parameterization with surface plenoptic function.

the corresponding angle. That is:

$$\tan(\theta) = \frac{f}{v} \text{ or } \theta = \frac{3\pi}{2} - \tan^{-1}\left(\frac{v}{f}\right), \tag{4.10}$$

where $-v_0 \leq v \leq v_0$ and $2\tan^{-1}\left(\frac{v_0}{f}\right)$ is the field of view (FOV). The above equation is actually the mapping between a pixel's angular position θ and its image coordinate v. Such a mapping can be linearized as:

$$\theta \approx \frac{3\pi}{2} - \frac{v}{f}. \tag{4.11}$$

The above linearization will introduce 4.3% maximum error if the FOV of the camera is 40°. In practice, the approximation in Equation 4.11 can be replaced by a simple pixel re-arrangement. In the following discussions, we denote $\phi = \frac{v}{f} \approx \frac{3\pi}{2} - \theta$ for conciseness, where $-\frac{v_0}{f} \leq \phi \leq \frac{v_0}{f}$ due to the limited FOV.

Another constraint is that the light ray (t, v) can be traced back to a cross point on the object surface, whose arc length s can be obtained through solving:

$$\begin{cases} x = x_i(s) \\ y = y_i(s) \\ fx - vy - ft = 0. \end{cases} \tag{4.12}$$

When multiple objects exist in the scene or some objects occlude themselves, Equation 4.12 may have multiple answers. We have to figure out which cross point is the closest

to the cameras. The closest point will occlude all the others. This may make scenes with occlusions hard to analyze. However, for simple scenes this is still doable and we will show examples later in this section.

4.2.1 Scene at a constant depth

The simplest scene we can have for light field is one at a constant depth, as shown in Figure 4.3(a). Denote its SPF as $l_0(s, \theta)$. The surface can be described by:

$$\begin{cases} x = x_0(s) = s \\ y = y_0(s) = d_0. \end{cases} \tag{4.13}$$

We can solve Equation 4.12 without concerning about occlusion:

$$fs - vd_0 - ft = 0 \Longrightarrow s = \frac{vd_0}{f} + t. \tag{4.14}$$

The light field spectrum can be derived as:

$$\begin{aligned} L_c(\Omega_t, \Omega_v) &= \iint l_c(t, v) e^{-j\Omega_t t - j\Omega_v v} dt dv \\ &= \iint l_0 \left[\frac{vd_0}{f} + t, \frac{3\pi}{2} - \tan^{-1}\left(\frac{v}{f}\right) \right] e^{-j\Omega_t t - j\Omega_v v} dt dv \\ &\approx \iint l_0(s, \theta) e^{-j\Omega_t (s - \phi d_0) - j\Omega_v \phi f} f ds d\theta \\ &= f L_0(\Omega_t, d_0\Omega_t - f\Omega_v) e^{j\frac{3\pi}{2}(d_0\Omega_t - f\Omega_v)} \end{aligned} \tag{4.15}$$

We can see that the spectrum of the light field at constant depth is a rotated version of the SPF spectrum, with some constant factor in magnitude and some shift in phase. The rotation angle is determined by the scene depth d_0 and the focal length f.

If the object surface is Lambertian, we have $L_0(\Omega_s, \Omega_\theta) = L_{0s}(\Omega_s)\delta(\Omega_\theta)$ as stated in Equation 4.4. Therefore:

$$\begin{aligned} L_c(\Omega_t, \Omega_v) &= f L_0(\Omega_t, d_0\Omega_t - f\Omega_v) e^{j\frac{3\pi}{2}(d_0\Omega_t - f\Omega_v)} \\ &= f L_{0s}(\Omega_t) \delta(d_0\Omega_t - f\Omega_v) e^{j\frac{3\pi}{2}(d_0\Omega_t - f\Omega_v)}. \end{aligned} \tag{4.16}$$

This is a tilted line in the (Ω_t, Ω_v) space, which is the same conclusion as that in [16].

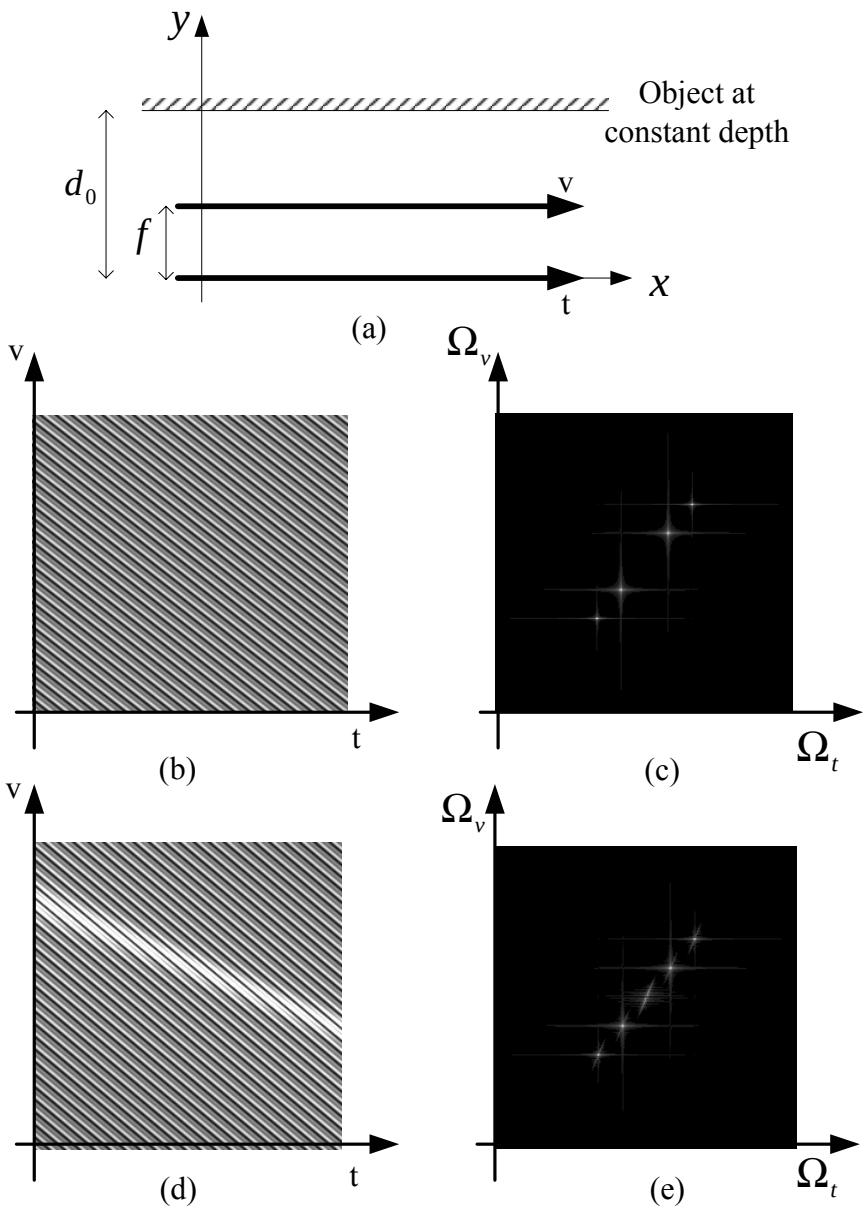


Figure 4.3: Spectrum of Lambertian and non-Lambertian scenes at constant depth. (a) A scene at constant depth; (b) the EPI of the light field when the scene is Lambertian; (c) the Fourier transform of (b); (d) the EPI of the light field when the scene is non-Lambertian; (e) the Fourier transform of (d).

When the object surface is non-Lambertian but the SPF is band-limited, we have $L_0(\Omega_s, \Omega_\theta) \approx L_0(\Omega_s, \Omega_\theta) I_{B_0}(\Omega_\theta)$ as in Equation 4.5. Consequently,

$$\begin{aligned} L_c(\Omega_t, \Omega_v) &= f L_0(\Omega_t, d_0\Omega_t - f\Omega_v) e^{j\frac{3\pi}{2}(d_0\Omega_t - f\Omega_v)} \\ &\approx f L_0(\Omega_t, d_0\Omega_t - f\Omega_v) e^{j\frac{3\pi}{2}(d_0\Omega_t - f\Omega_v)} I_{B_0}(d_0\Omega_t - f\Omega_v). \end{aligned} \tag{4.17}$$

The spectrum is also tilted, but this time it has a finite width $\frac{2B_0}{\sqrt{d_0^2+f^2}}$ perpendicular to the tilted spectrum (or $\frac{2B_0}{d_0}$ horizontally) because of the indicator function.

The above analysis is illustrated in Figure 4.3. A scene at constant depth has two sinusoids (different frequency) pasted on it as texture, as shown in Figure 4.3(a). Figure 4.3(b) is the epipolar image (EPI, which is an image composed of light rays using the camera line t and focal line v as its two axes) when the scene is Lambertian. Figure 4.3(c) is its Fourier transform. The spectrum has several peaks because the texture on the scene object is pure sinusoids. It basically lies on a tilted line with some small horizontal and vertical windowing artifact that is due to the truncation of the range of s and θ. We ignored the windowing artifacts in our analysis for simplicity. Figure 4.3(d) shows the EPI for a non-Lambertian case at the same depth, which is generated by adding a point light source near the scene and assuming a Phong surface reflection model [109]. It can be seen that because of the non-Lambertian property, its Fourier transform in Figure 4.3(e) is expanded. The direction of the expansion is determined by the depth of the point light source, which will also affect the SPF bandwidth.

4.2.2 Scene on a titled line

We next study a scene on a tilted line, as shown in Figure 4.4. We write the surface equation as:

$$\begin{cases} x = x_0(s) = s\cos\varphi + x_0 \\ y = y_0(s) = s\sin\varphi + y_0 \end{cases} \tag{4.18}$$

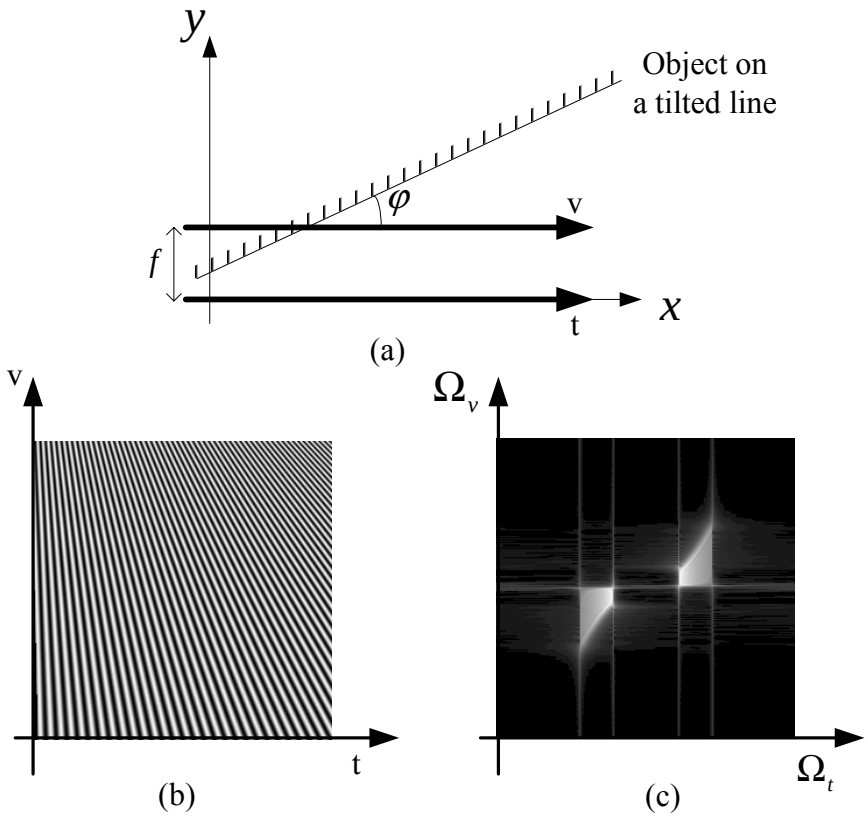


Figure 4.4: Spectrum of a tilted planar scene with a sinusoid pasted as texture. (a) The scene on a tilted line; (b) the EPI of the lightfield; (c) the Fourier transform of (b).

where $0 \leq \varphi < \pi$ is an angle that is known. Assuming no occlusion, we can solve Equation 4.12 as:

$$\begin{aligned} f(s\cos\varphi + x_0) - v(s\sin\varphi + y_0) - ft = 0 \quad &\Longrightarrow \\ s = \frac{ft - fx_0 + vy_0}{f\cos\varphi - v\sin\varphi}. \end{aligned} \tag{4.19}$$

To guarantee that within each captured image the parameter s is continuous, φ cannot be arbitrary. If $-v_0 \leq v \leq v_0$, we can have constraint that $f\cos\varphi - v_0|\sin\varphi| > 0$.

The light field spectrum is:

$$\begin{aligned} L_c(\Omega_t, \Omega_v) &= \iint l_c(t,v) e^{-j\Omega_t t - j\Omega_v v} dt dv \\ &= \iint l_0\left[\frac{ft - fx_0 + vy_0}{f\cos\varphi - v\sin\varphi}, \frac{3\pi}{2} - \tan^{-1}\left(\frac{v}{f}\right)\right] e^{-j\Omega_t t - j\Omega_v v} dt dv \\ &\approx \iint l_0(s,\theta) e^{-j\Omega_t\left[s(\cos\varphi - \phi\sin\varphi) + (x_0 - \phi y_0)\right] - j\Omega_v \phi f} \\ &\quad \cdot(\cos\varphi - \phi\sin\varphi) f ds d\theta. \end{aligned} \tag{4.20}$$

Unfortunately, even for a scene as simple as a line, Equation 4.20 is too complex to solve. Here we consider a case where Equation 4.20 can be further simplified. Let the scene be Lambertian, i.e., as in Equation 4.4. We have:

$$\begin{aligned} L_c(\Omega_t, \Omega_v) &\approx \iint l_0(s,\theta) e^{-j\Omega_t\left[s(\cos\varphi - \phi\sin\varphi) + (x_0 - \phi y_0)\right] - j\Omega_v \phi f} \\ &\quad \cdot(\cos\varphi - \phi\sin\varphi) f ds d\theta. \\ &= \iint l_{0s}(s) e^{-j\Omega_t\left[s(\cos\varphi - \phi\sin\varphi) + (x_0 - \phi y_0)\right] - j\Omega_v \phi f} \\ &\quad \cdot(\cos\varphi - \phi\sin\varphi) f ds d\theta. \\ &= \int L_{0s}\left[\Omega_t(\cos\varphi - \phi\sin\varphi)\right] e^{-j\Omega_t(x_0 - \phi y_0) - j\Omega_v \phi f} \\ &\quad \cdot(\cos\varphi - \phi\sin\varphi) f d\theta. \end{aligned} \tag{4.21}$$

If we have a sinusoid pasted as the texture, e.g., $l_{0s}(s) = \sin(\Omega_0 s)$. Its Fourier transform is:

$$L_{0s}(\Omega_s) = \frac{1}{2j}\left[\delta(\Omega_s - \Omega_0) - \delta(\Omega_s + \Omega_0)\right]. \tag{4.22}$$

Due to symmetry, let us consider the magnitude of $L_c(\Omega_t, \Omega_v)$ when $\Omega_t > 0$ and $\Omega_s > 0$. From Equation 4.21, by changing the integration variable from θ to ϕ we have:

$$\begin{aligned} \left|L_c(\Omega_t, \Omega_v)\right| &\approx \frac{1}{2}\left| \int_{-\frac{v_0}{f}}^{\frac{v_0}{f}} \delta\left[\Omega_t(\cos\varphi - \phi\sin\varphi) - \Omega_0\right] e^{-j\Omega_t(x_0 - \phi y_0) - j\Omega_v \phi f} \right. \\ &\quad \left. \cdot(\cos\varphi - \phi\sin\varphi) f d\phi \right| \\ &= \begin{cases} \frac{\Omega_0 f}{2\Omega_t^2 |\sin\varphi|}, & \text{when } \frac{\Omega_0 f}{f\cos\varphi + v_0|\sin\varphi|} \le \Omega_t \le \frac{\Omega_0 f}{f\cos\varphi - v_0|\sin\varphi|}; \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{4.23}$$

Figure 4.4(a) shows a scene on a line with $\varphi = \frac{\pi}{6}$. A single sinusoid is pasted on the line as texture. Figure 4.4(b) is the EPI of the light field, and Figure 4.4(c) is its Fourier transform. Notice that our analysis matches very well with the example. For example, the spectrum is non-zero only between certain Ω_t thresholds. In the non-zero region, the spectrum decays when $|\Omega_t|$ increases. The spectrum is bounded for Ω_v due to the minimum and maximum depth of the scene, as was described in [16]. One thing we may notice in this example is that compared with the original SPF (which is just a sinusoid), a scene as simple as a tilted line can cause the IBR spectrum to spread a lot. Considering the fact that light rays from a certain surface point tend to change slowly in the real world, we may conclude that the spectrum spreading in real world scenes is mainly caused by its irregular shape, including the occluding effects which we will discuss in Section 4.2.3.

There are two practical considerations. First, the geometry of the scene may be very difficult to obtain. When the geometry is unknown, we may approximate the scene geometry with piecewise constant depths, as discussed in Section 4.3. Following the "truncating windows" analysis there, we may obtain first-order approximations of the spectrum, even if the scene is non-Lambertian. Second, given the scene geometry, sometimes an analytical expression associating the spectrum of SPF and that of IBR representation may not exist. To solve for such scenes, numerical methods may be applied. For example, by replacing $l_0(s, \theta)$ with the inverse Fourier transform of $L_0(\Omega_s, \Omega_\theta)$, Equation 4.20 could be solved numerically.

4.2.3 Occlusions between objects

When occlusion happens, IBR spectral analysis becomes very difficult due to the discontinuity at the occlusion boundary. In fact, any occlusion will simply cause the IBR spectrum to be band-unlimited. In this subsection, we analyze occluded scenes under certain assumptions, and give some insights on the formation of the spectrum of such scenes.

With SPF, solving for occluded scenes means finding the closest cross point among multiple solutions for Equation 4.12. We first assume that objects do not occlude themselves.

Marchand-Maillet [79] once derived the no-occlusion condition for functional surfaces. Similarly, in our notation, no occlusion requires:

$$\max_s \left| \frac{y_i'(s)}{x_i'(s)} \right| < \frac{f}{v_0}. \tag{4.24}$$

where $y_i'(s)$ and $x_i'(s)$ are first order derivatives. Equation 4.24 simply means that the slope of the surface should not be greater than that of any possible light rays captured. This assumption is, however, hard to justify. We treat this as an example where occlusion can be solved under our parameterization. In the meantime, in practice mutual occlusions are often more significant than self-occlusions[2], and self occluded objects can sometimes be decomposed into smaller objects so that the occlusions become mutual.

When objects do not occlude themselves, their spectrums can be obtained through previously mentioned methods, as there are no occlusions. Let the number of objects in the scene be N. Let $l_{c(i)}(t,v), 0 \le i \le N-1$ be the N light fields and $L_{c(i)}(\Omega_t, \Omega_v), 0 \le i \le N-1$ be their Fourier transforms. We also define a silhouette light field for each object i as:

$$\gamma_{c(i)}(t,v) = \begin{cases} 1, & \text{when light ray } (t,v) \text{ can be traced back to object } i,\ 0 \le i \le N-1; \\ 0, & \text{otherwise.} \end{cases} \tag{4.25}$$

Their Fourier transforms are denoted as $\Gamma_{c(i)}(\Omega_t, \Omega_v), 0 \le i \le N-1$. Notice that the silhouette's spectrum can be obtained by setting $l_i(s,\theta) \equiv 1$ on object i's surface.

We are now ready to find the spectrum of the occluded scene. Since Fourier transform is linear, the overall spectrum is simply the sum of individual objects' spectrums. If an object is not occluded, we can just add its spectrum to the overall one. Otherwise, let object i be occluded by K other objects ($K < N-1$). Denote the K occluding objects' silhouette light fields as $\gamma_{c(i_k)}(t,v), 0 \le k \le K-1$. The contribution object i has to the overall light field can be written as:

$$l_{c(i)}^{occ}(t,v) = l_{c(i)}(t,v) \prod_{k=0}^{K-1} \left[1 - \gamma_{c(i_k)}(t,v)\right] \tag{4.26}$$

[2]We claim mutual occlusion is more significant than self-occlusion because the former often causes a sharp boundary/edge in the EPI, while self-occlusion often does not if the surface normal changes slowly.

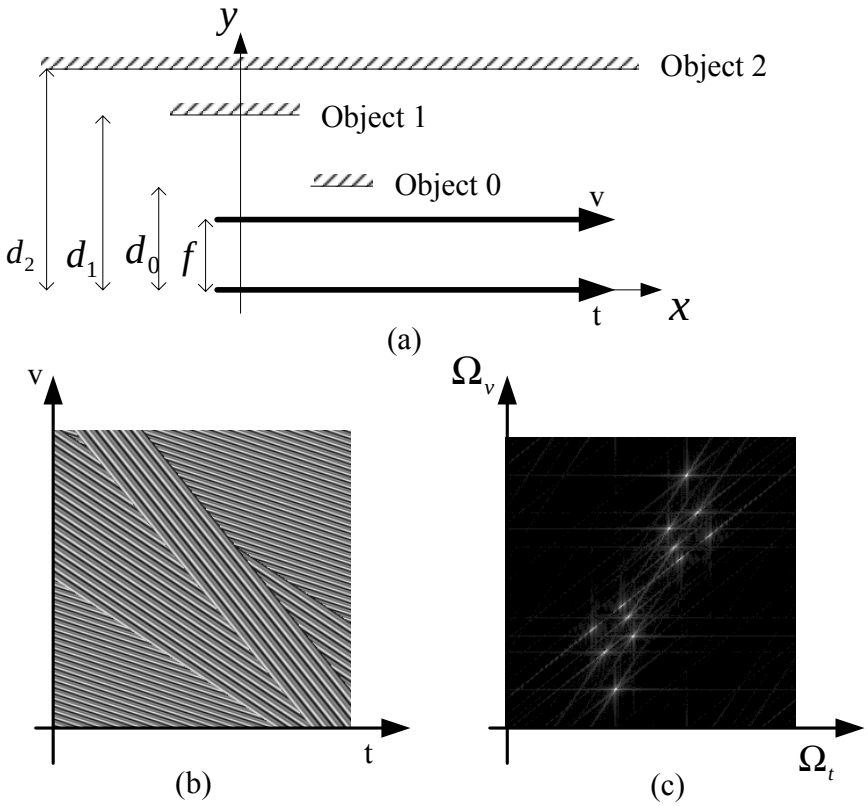


Figure 4.5: Spectrum of an occluded scene. (a) Three objects on different constant depths; (b) the EPI of the light field; (c) the Fourier transform of (b).

where $l^{occ}_{c(i)}(t,v)$ is the occluded light field of object i. Its Fourier transform is:

$$\begin{aligned} L^{occ}_{c(i)}(\Omega_t,\Omega_v) &= L_{c(i)}(\Omega_t,\Omega_v) \otimes \left[\delta(\Omega_t,\Omega_v) - \Gamma_{c(i_0)}(\Omega_t,\Omega_v)\right] \\ &\quad \otimes \cdots \otimes \left[\delta(\Omega_t,\Omega_v) - \Gamma_{c(i_{K-1})}(\Omega_t,\Omega_v)\right] \end{aligned} \tag{4.27}$$

where $\otimes$ stands for convolution, $\delta(\Omega_t,\Omega_v)$ is the Fourier transform of constant 1. From Equation 4.27 we can see that the spectrum of an occluded object is its unoccluded spectrum modulated by all the occluding objects' silhouette spectrum. This modulation will bring some additional components to the overall spectrum.

Figure 4.9(a) shows an example Lambertian scene that has three objects. Each object is at a constant depth and has two sinusoids pasted as texture. Therefore, from what we had in Section 4.2.1, if no occlusion is considered, for $0 \le i \le 2$, $L_{c(i)}(\Omega_t,\Omega_v)$ and $\Gamma_{c(i)}(\Omega_t,\Omega_v)$ both

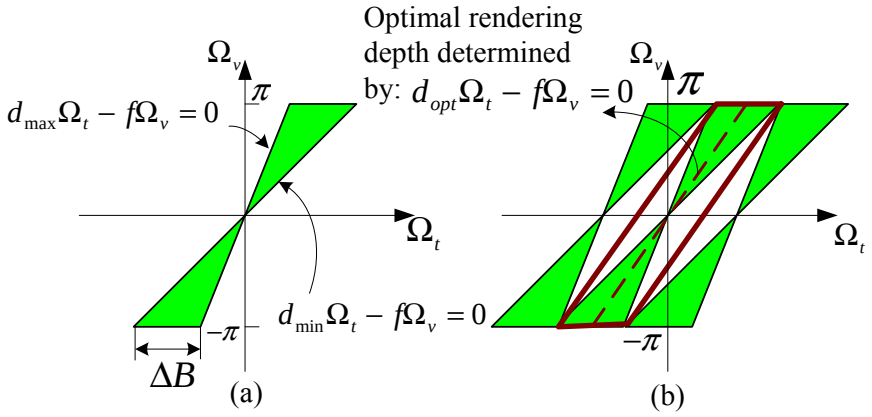


Figure 4.6: Best rectangular sampling of a non-occluded and Lambertian scene, as in [16]. (a) Frequency support of a scene with no occlusion and Lambertian surface; (b) the Optimal rectangular compacting and the corresponding reconstruction filter.

lie on a tilted line whose slope is $\frac{d_i}{f}$, where d_i is the depth of object i. Since objects at larger depth are occluded by closer objects, we will notice additional modulated components along larger slope lines. Moreover, the additional modulated components will be along smaller slopes corresponding to the closer object. This is clearly shown in Figure 4.9(c).

4.3 Analysis of Scenes with Unknown Geometry

The surface plenoptic function is defined on the scene's geometry. It may cause the misunderstanding that the above analysis is not applicable when the scene geometry is unknown. In this section, we show how such scenes can be analyzed with the "truncating windows" analysis [16] and the conclusions obtained above.

We know that it is possible to approximate a complex scene with multiple constant depth layers $d_j, 0 \leq j \leq J-1$. In [16], Chai et al. adopted such assumption and gave a "truncating windows" analysis for non-Lambertian scenes. They showed that if there is no occlusion, the spectrum support of each layer looks like a narrow ellipse. The overall spectrum is simply the sum of the spectrums for all the layers. As a first order approximation, the overall spectrum will be between two slopes—one determined by the minimum scene depth $d_{\min}$ and the other determined by the maximum scene depth $d_{\max}$. This is drawn in Figure 4.6(a).

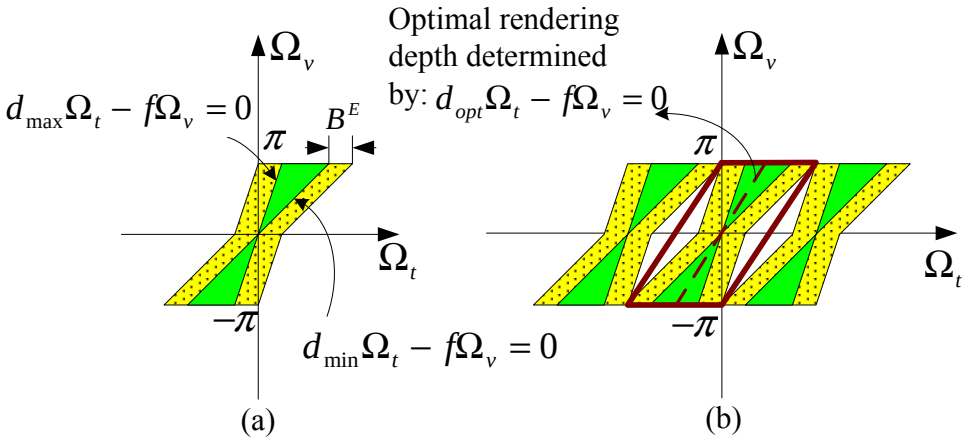


Figure 4.7: Best rectangular sampling of a non-Lambertian scene. (a) The spectrum is expanded from the Lambertian case; (b) the optimal rectangular compacting and the corresponding reconstruction filter.

Without loss of generality we assume in this paper that the spectrum is mainly bounded by the resolution of the camera, and the unit of the v axis is 1, so that $-\pi \leq \Omega_v < \pi$. If rectangular sampling is applied, at minimum sampling rate the spectrum is replicated as in Figure 4.6(b). The corresponding maximum sampling distance is:

$$\Delta t_{\max} = \frac{2}{f\left(\frac{1}{d_{\min}} - \frac{1}{d_{\max}}\right)}, \tag{4.28}$$

where f represents the focal length. The authors of [16] also proposed to use

$$d_{opt} = \frac{2}{\left(\frac{1}{d_{\min}} + \frac{1}{d_{\max}}\right)}, \tag{4.29}$$

as the best depth to render the scene, which can be easily obtained through Figure 4.6(b).

If the scene is non-Lambertian, it is clear that the "truncating windows" analysis is still valid except that for each layer, the spectrum support becomes a "fatter" ellipse. This causes the original fan-like spectrum to be expanded. If the SPF is band-limited along Ω_θ as in Equation 4.5, the light field bandwidth expansion is also limited, as is shown in Figure 4.7(a). The dotted region stands for the extra spectrum support caused by the non-Lambertian surface. A conservative estimation of the amount of expansion is:

$$B^E = \max_j \frac{B_j}{d_j} \tag{4.30}$$

on each side of the spectrum along the horizontal axis Ω_t, based on the discussion after Equation 4.17. Here B_j and d_j represent the bandwidth of SPF and depth for object depth layer j. Therefore, with rectangular sampling, the best way to compact the spectrum is as Figure 4.7(b), and the sampling rate of non-Lambertian scenes has to be increased.

In practice, the effect of non-Lambertian property in expanding the spectrum is often difficult to observe because the light rays from the same surface point change radiance often very slowly. We next show an example where we have to take more images in order to render correctly because of the non-Lambertian property. We render two spheres (Figure 4.8(a) and (b))—one is Lambertian and the other is a purely reflective surface. Meanwhile, we assume that we have a 6-bit depth map of the scene. According to the sampling curve proposed in [16], the number of images required for rendering the scene is reduced if we have an accurate depth map. We found that at a certain sampling rate, we are able to reconstruct the Lambertian scene very well as in Figure 4.8(c). However, if we use the same sampling rate for the reflective surface, the reconstruction is very bad (Figure 4.8(d)). This means that we need to sample more images to reconstruct the latter surface. In Figure 4.8(e) and (f), we show the EPI of both scenes (center row, at a much higher sampling rate along t for illustration purpose). The red dashed lines are interpolation directions determined by the sphere geometry. Obviously, the rendering or interpolation of the reflective surface is along a wrong direction, or using a wrong geometry, which causes its poor quality.

The "truncating windows" analysis can be applied to more complex scenes if occlusions have to be considered. When the scene is heavily occluded, far layers will be blocked by close layers, which causes a spectrum modulation as in Equation 4.27. In worst case, the major occluding objects are at $d_{\min}$, which generates additional modulated components along the slope $\frac{d_{\min}}{f}$, where f is the focal length. The corresponding spectrum is given in Figure 4.9(a). The dotted support is the additional modulated components. If we still apply the rectangular sampling strategy, the spectrum can be compacted as in Figure 4.9(b). Notice that we have to increase the minimum sampling rate by a factor of 2. Or, the maximum

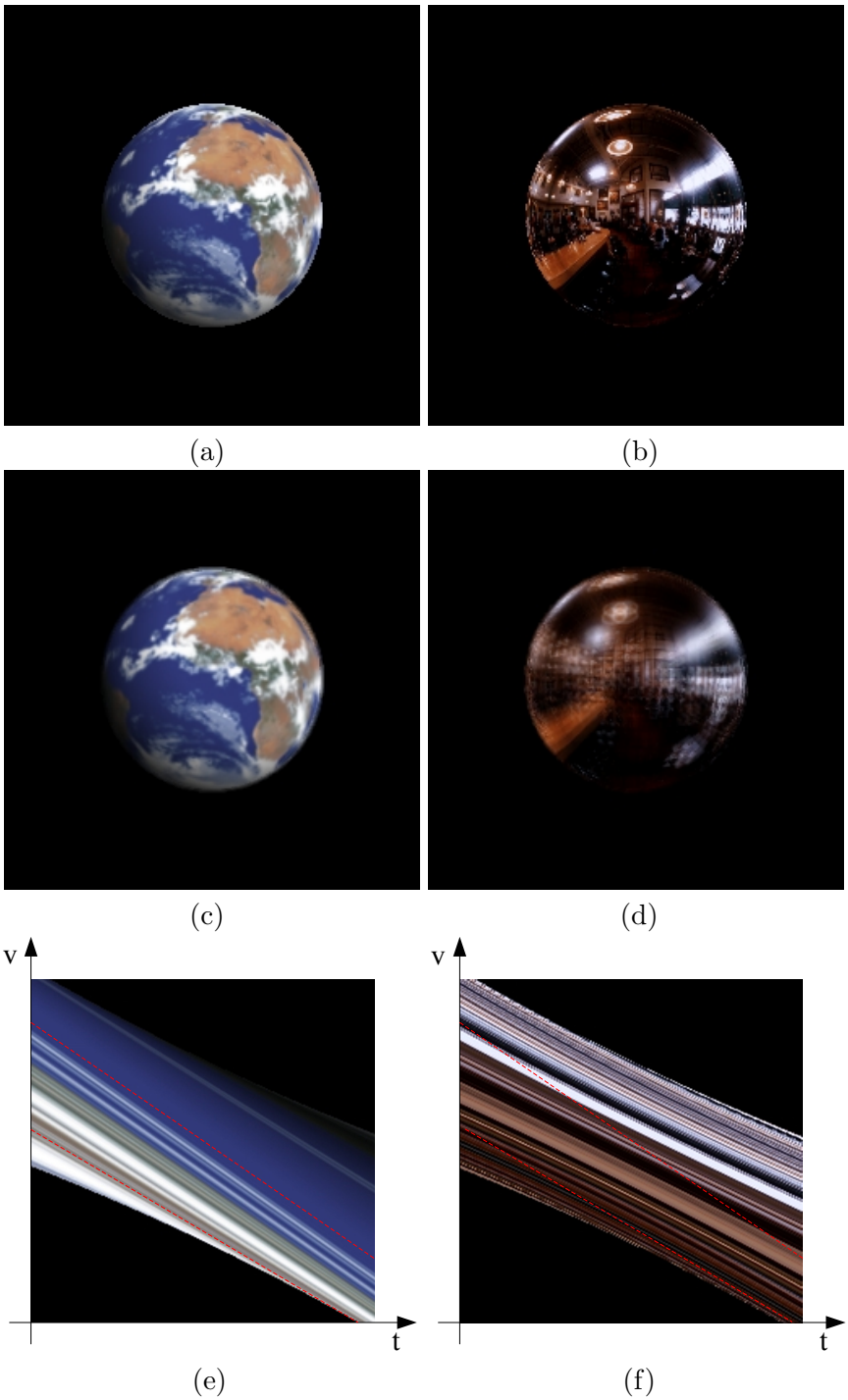


Figure 4.8: An example of the minimum sampling rate for non-Lambertian surface. (a) A Lambertian sphere; (b) a purely reflective sphere; (c) at a certain sampling rate, we are able to reconstruct the scene very well for the Lambertian sphere; (d) at the same sampling rate, the reconstruction is very bad for the reflective surface; (e) EPI of the Lambertian sphere scene (center row); (f) EPI of the reflective surface scene (center row).

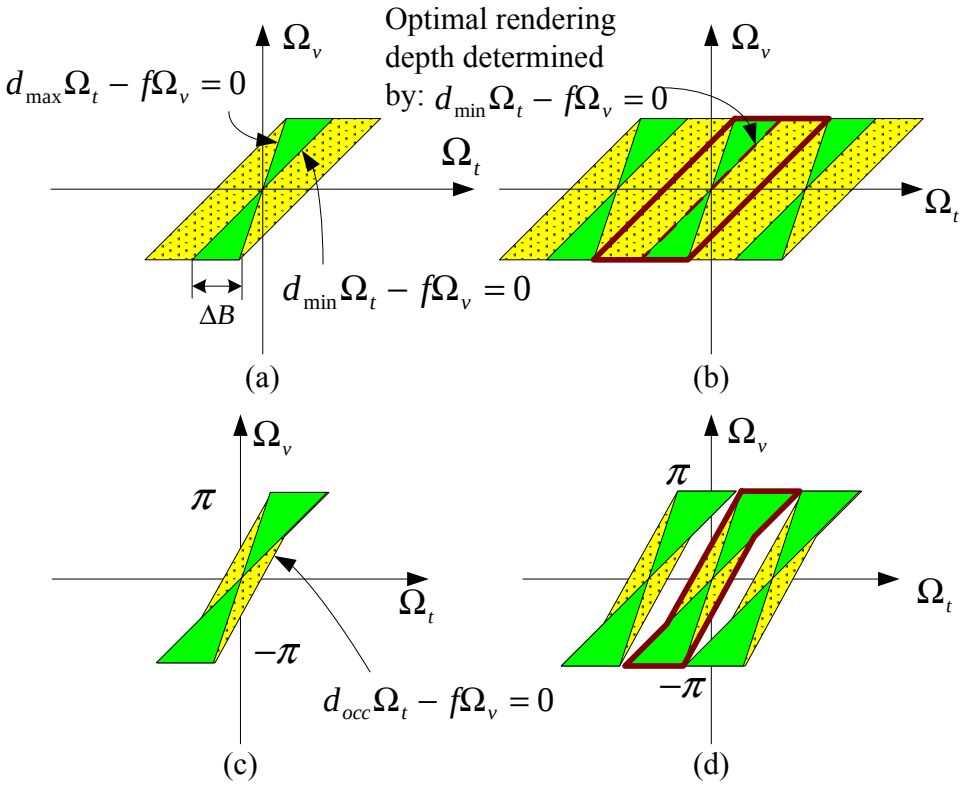


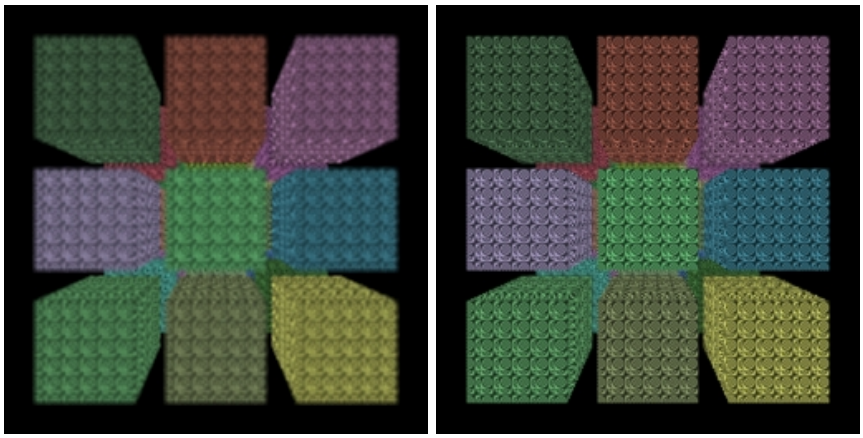
Figure 4.9: Best rectangular sampling of occluded scenes. (a) The most conservative estimation of the spectrum when the object at $d_{\min}$ causes the major occlusions; (b) the optimal rectangular sampling strategy for compacting (a); (c) the spectrum given that the object at d_{occ} is the main source of occlusions; (d) the optimal rectangular sampling strategy for compacting (c).

sampling distance is:

$$\Delta t_{\max} = \frac{1}{f\left(\frac{1}{d_{\min}} - \frac{1}{d_{\max}}\right)}. \tag{4.31}$$

Meanwhile, from Figure 4.9(b) we see that the optimal rendering depth becomes $d_{\min}$ instead of d_{opt} in Equation 4.29. In general, when the major occluding objects are at distance d_{occ}, the spectrum is shown in Figure 4.9(c). After the most compact rectangular sampling, the spectrum is shown in Figure 4.9(d). The minimum sampling rate and the optimal rendering depth both depend on d_{occ}.

We give an example to verify the above analysis. In Figure 4.10 we show an OpenGL scene composed of $3 \times 3 \times 3$ cubes. The cubes stay on a 3D regular grid and the front cubes



(a) (b)

Figure 4.10: An example of the minimum sampling rate for occluded surface. (a) At a certain sampling rate, rendered with d_{opt}; (b) same as (a) but rendered with $d_{\min}$.

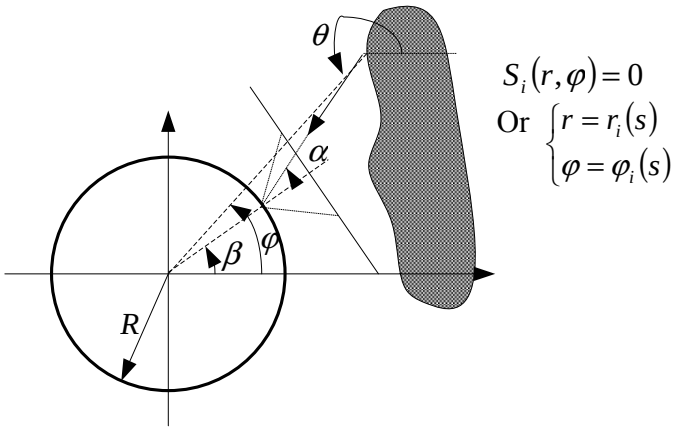


Figure 4.11: The concentric mosaics parameterization with surface plenoptic function.

occlude most regions of the back cubes. The renderings are done through depth-corrected bilinear interpolation assuming a constant depth. Figure 4.10(a) and (b) show the rendered scene with rendering depth d_{opt} and $d_{\min}$, respectively. Obviously, Figure 4.10 (b) is much more pleasing because the foreground cubes are rendered at hight quality. The background cubes are occluded and do not contribute too much to the overall visual quality.

4.4 Analysis for the Concentric Mosaics

The proposed analysis can also be applied to concentric mosaics. Concentric mosaics is captured by rotating a camera along a circle, as is shown in Figure 4.11. Let the circle radius be R. We index concentric mosaics by two angles, α and β. $-\alpha_0 \leq \alpha \leq \alpha_0$ represents the angle between the light ray and the camera optical axis; β denotes where the camera is on the circle. For concentric mosaics, we use polar coordinates instead of rectangular coordinates. The object surfaces are represented as:

$$S_i(r, \varphi) = 0 \text{ or } \begin{cases} r = r_i(s) \\ \varphi = \varphi_i(s) \end{cases} . \tag{4.32}$$

The light ray line indexed by pair (β, α) is:

$$r - \frac{R \sin \alpha}{\sin(\alpha + \beta - \varphi)} = 0. \tag{4.33}$$

Equation 4.33 is fairly complex if we want to solve for α and β from r and φ. Provided that in practice the FOV of the cameras is often limited[3], we let $\sin \alpha \approx \alpha$. Because of the setup of concentric mosaics, the objects in the scene must be outside the capturing circle, we have $|\alpha + \beta - \varphi| < |\alpha|$. Therefore, we can also let $\sin(\alpha + \beta - \varphi) \approx \alpha + \beta - \varphi$. Equation 4.33 is thus simplified as:

$$r(\alpha + \beta - \varphi) - R\alpha \approx 0. \tag{4.34}$$

We also have simple relationship:

$$\alpha + \beta + \pi = \theta. \tag{4.35}$$

Similar to the analysis in light field, light ray (β, α) should be traced back to a cross point on the object surface, whose arc length s can be obtained through solving:

$$\begin{cases} r = r_i(s) \\ \varphi = \varphi_i(s) \\ r(\alpha + \beta - \varphi) - R\alpha \approx 0 \end{cases} \tag{4.36}$$

Equation 4.36 is in great similarity to Equation 4.12. Therefore, discussions on light field can be borrowed directly. Here we only give a simple example by considering a scene at

[3]When the FOV is about 40°, i.e., $\alpha_0 \approx \frac{\pi}{9}$. Making $\sin \alpha \approx \alpha$ has 2% maximum error. Notice that in concentric mosaics we have the same linearization error as in light field (Equation 4.11).

constant depth. A constant depth concentric mosaics scene is described as:

$$\begin{cases} r = r_i(s) = r_0 \\ \varphi = \varphi_i(s) = \frac{s}{r_0} \end{cases} \tag{4.37}$$

where r_0 is the depth of the scene. We can easily solve Equation 4.36:

$$r_0\Big(\alpha + \beta - \frac{s}{r_0}\Big) - R\alpha \approx 0 \Longrightarrow s \approx r_0(\alpha+\beta) - R\alpha. \tag{4.38}$$

The concentric mosaic spectrum can be derived as:

$$\begin{aligned} L_c(\Omega_\beta, \Omega_\alpha) &= \iint l_c(\beta,\alpha) e^{-j\Omega_\beta\beta - j\Omega_\alpha\alpha} d\beta d\alpha \\ &\approx \iint l_0\big[r_0(\alpha+\beta) - R\alpha, \alpha+\beta+\pi\big] e^{-j\Omega_\beta\beta - j\Omega_\alpha\alpha} d\beta d\alpha \\ &= \iint l_0(s,\theta) e^{-j\Omega_\beta \frac{s+(r_0-R)(\pi-\theta)}{R} - j\Omega_\alpha \frac{r_0\theta - s - \pi r_0}{R}} \frac{1}{R} ds d\theta \\ &= \frac{1}{R} L_0\Big[\frac{\Omega_\beta - \Omega_\alpha}{R}, \frac{r_0\Omega_\alpha - (r_0-R)\Omega_\beta}{R}\Big] e^{-j\Omega_\beta \frac{(r_0-R)\pi}{R} + j\Omega_\alpha \frac{\pi r_0}{R}} \end{aligned} \tag{4.39}$$

Again, the spectrum is a rotated version of that of SPF, plus some magnitude change and phase shift. If the scene is Lambertian, or $L_0(\Omega_s, \Omega_\theta) = L_{0s}(\Omega_s)\delta(\Omega_\theta)$, we further have:

$$\begin{aligned} L_c(\Omega_\beta, \Omega_\alpha) &= \frac{1}{R} L_0\Big[\frac{\Omega_\beta - \Omega_\alpha}{R}, \frac{r_0\Omega_\alpha - (r_0-R)\Omega_\beta}{R}\Big] e^{-j\Omega_\beta \frac{(r_0-R)\pi}{R} + j\Omega_\alpha \frac{\pi r_0}{R}} \\ &= \frac{1}{R} L_{0s}\Big(\frac{\Omega_\beta - \Omega_\alpha}{R}\Big) \delta\Big[\frac{r_0\Omega_\alpha - (r_0-R)\Omega_\beta}{R}\Big] e^{-j\Omega_\beta \frac{(r_0-R)\pi}{R} + j\Omega_\alpha \frac{\pi r_0}{R}} \end{aligned} \tag{4.40}$$

The spectrum lies on a line that has slope

$$\frac{\Omega_\alpha}{\Omega_\beta} = \frac{r_0 - R}{r_0}. \tag{4.41}$$

Since $r_0 > R$, the slope is always less than 45°.

Figure 4.12(a) shows an example scene at constant depth with two sinusoids pasted as texture. In Figure 4.12(b) the coordinates α and β are at different scale so the slope does not map to the true value. However, the spectrum in Figure 4.12(c) is obviously along a line. If the scale between α and β is the same, the line will achieve maximum slope when the depth of the scene becomes infinite.

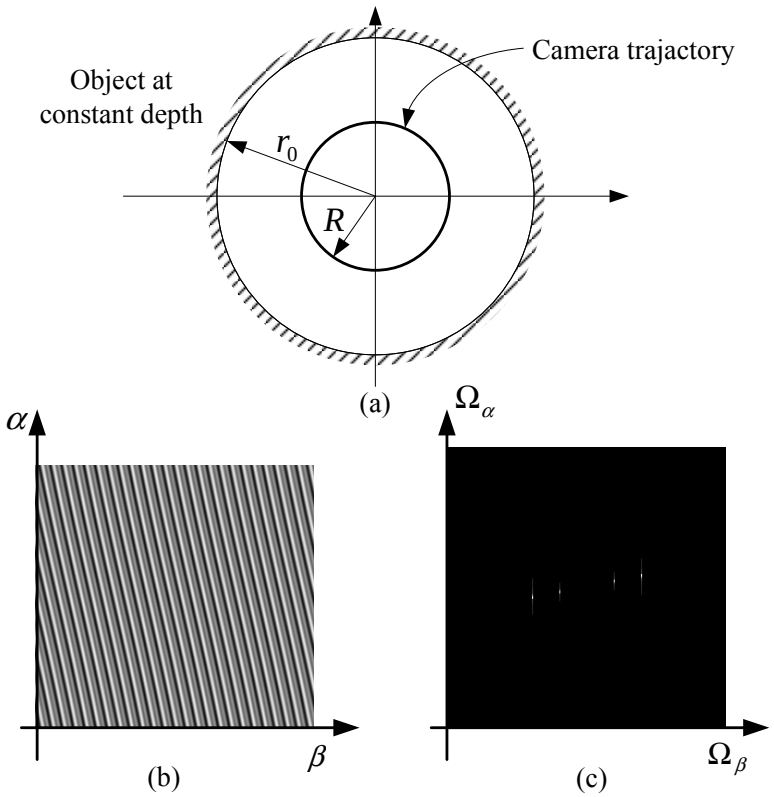


Figure 4.12: Spectrum of a Lambertian scene at constant depth for concentric mosaics. (a) A scene at constant depth with sinusoids pasted as texture; (b) the EPI of the concentric mosaics when the scene is Lambertian. (c) The Fourier transform of (b).

4.5 Summary

In this chapter, we presented a new approach to analyzing the IBR spectrum. We first defined the surface plenoptic function, then showed that any IBR representation can be studied if we can find the light ray mapping from the SPF to the IBR representation. We showed examples where the mapping can be done for light field and concentric mosaics. Non-Lambertian property, scene depth variations and occlusions fit naturally into our approach and we were able to analyze the effects of them as long as certain assumptions are satisfied. We believe that this approach greatly improves the understanding of IBR spectral analysis.

Spectrum analysis is the first but not the last step of IBR uniform sampling analysis. Knowing the spectrum of the scene, it is also important to know how should we sample the

signal. Rectangular sampling was used in the light field application as it is straightforward. In the next chapter, we will apply generalized sampling theory to such a task and compare its performance with rectangular sampling.

Chapter 5

Uniform Sampling: Generalized IBR Sampling

In this chapter, we apply generalized sampling (GS) theory [31] to image-based rendering (IBR) data, more specifically, the light field. Based on the analysis in the last chapter, concentric mosaics share similar spectrum with light field, thus the following analysis will also apply. We show that in theory the lowest sampling rate of light field when we use generalized sampling can be as low as half of that when we use rectangular sampling (RS). However, in practice rectangular sampling has several advantages over generalized sampling. We analyze the pros and cons for each sampling approach, and explain why in practice rectangular sampling is still more preferable.

5.1 Generalized Sampling for Light Field Data

In the spectrum analysis of IBR data in Chapter 4, we used rectangular sampling to demonstrate that more samples are needed when the scene is non-Lambertian or occluded. While the conclusions drawn there are still valid, rectangular sampling is not the best sampling strategy in high-dimensional space, which is a well-known fact in multi-dimensional signal processing theory [31]. Hexagonal sampling, for instance, is known as the best sampling strategy for a circularly band-limited 2D signal, which outperforms rectangular sampling for about 13.4% in sampling efficiency.

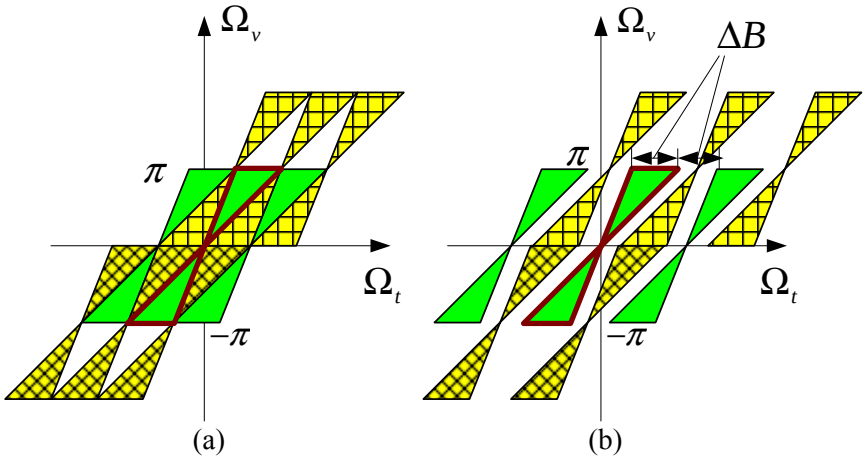


Figure 5.1: Generalized sampling of light field. (a) The most compact way to pack the light field spectrum; (b) reduce the sampling rate such that GS has the same efficiency as RS in Figure 4.6(b).

Figure 5.1(a) shows how we can compact the light field frequency support and the replicas better with generalized sampling theory. We ignored non-Lambertian and occlusion effects in this example. With this sampling strategy, the sampling efficiency can be improved by a factor of 2 compared with Figure 4.6(b), which means we only need 50% of the samples. The reconstruction filter is marked in bold contour in Figure 5.1(a), which is a tilted fan-like filter. For detailed information about the generalized sampling theory, we refer the reader to [31].

There are several problems to be concerned for the GS approach. First, the corresponding sampling lattice in the spatial domain for Figure 5.1(a) may not be consistent with how we take images for the scene. We propose to sample the scene with RS at a higher sampling rate and then down-sample it to the required lattice. Interpolation may be required during the down-sampling process. Of course, interpolation will introduce extra errors in practice. This is what GS has to pay to achieve a lower sampling rate.

Second, during the rendering, there can be two approaches to reconstruct the continuous light field signal from the GS sampled data. One is to introduce a preprocessing stage before rendering. At this stage we up-sample the data to a rectangular grid by a discrete fan-like reconstruction filter. During the rendering we can simply apply depth-driven bilinear

interpolation as before. This approach requires huge amount of memory to store the up-sampled data, but has a fast rendering speed. Another approach is to use a continuous fan-like reconstruction filter directly for rendering. Since it is difficult to find the best continuous filter given its finite support (which is required in rendering due to speed consideration), we design a discrete optimal reconstruction filter first and interpolate it to get the continuous filter. To speed up the rendering, look-up tables can be used to store the filter values. We adopt the second approach in the follow sections because it does not need much memory to store the up-sampled data.

Third, in theory we may be able to design an ideal fan-like reconstruction filter to get back the original light-field signal without losing any information; in practice, however, we cannot design a filter without any transition band. We choose to reduce the sampling density for the ease of filter design. To give a fair comparison between GS and RS, we let them have the same sampling density, as is shown in Figure 4.6(b) and Figure 5.1(b). We will focus on the optimal discrete reconstruction filter design for these two cases.

Assume we start with a RS sampled data at a sampling rate that is much higher than the minimum requirement. Its *discrete* Fourier transform is shown in Figure 5.2(a), where the two slopes $k_{\min}$ and $k_{\max}$ are determined by the minimum and maximum depths of the scene. If we down-sample the data with RS to its minimum sampling rate, we get a new spectrum as in Figure 5.2(b). If we down-sample it to the same rate with GS, we have Figure 5.2(c)[1]. Figure 5.2(d) and (e) are up-sampling filter specifications for (b) and (c), respectively. S, T and P stand for stop-band, transition-band and pass-band, respectively. We can observe that the transition-band of the up-sampling filter for RS is very narrow for high frequency components, but very wide for low frequency components. On the other hand, the transition-band of the filter for GS has constant width along all the frequencies. This observation implies that RS is better for scenes that has less high-frequency components, while GS is better otherwise.

We use the eigenfilter approach [145] to design the filters in Figure 5.2(d) and (e). Let

[1]Strictly speaking, Figure 5.2(c) is a transformed version of the original discrete Fourier transform. We perform this transform so that we can compare the filter specifications of the two sampling strategies. Please refer to [31] for details about such transforms.

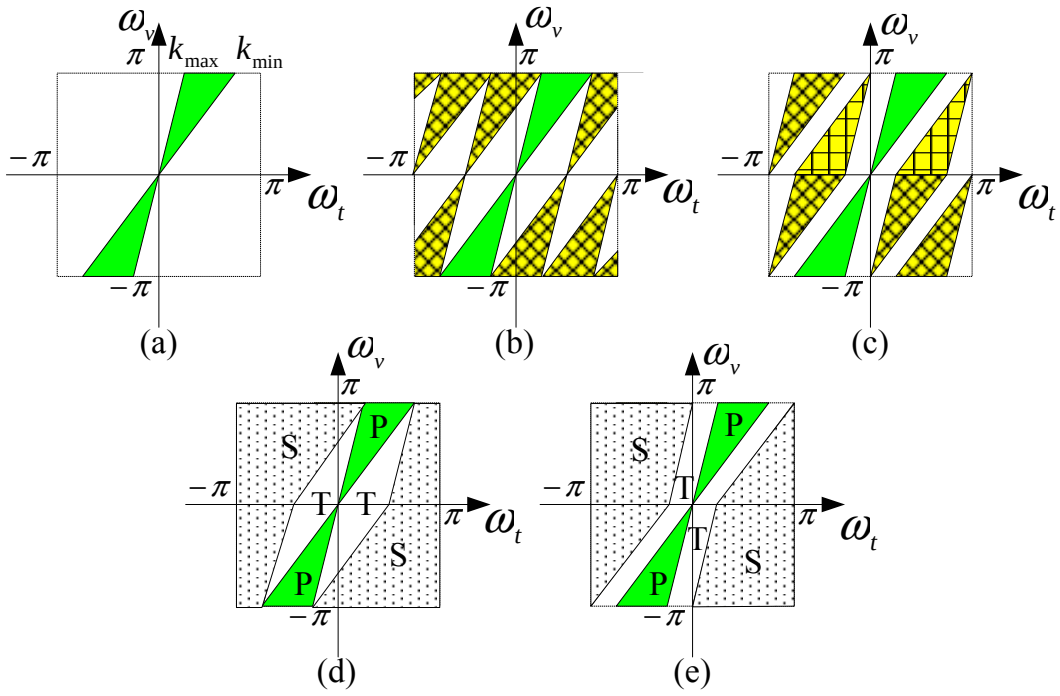


Figure 5.2: Filter design for light field reconstruction. (a) The Fourier transform of the original discrete signal; (b) the Fourier transform of the signal with minimum sampling rate for RS; (c) the Fourier transform of the signal with the same sampling rate for GS; (d) up-sampling filter specification for RS; (e) up-sampling filter specification for GS.

$D(\omega_t, \omega_v)$ be the desired frequency response in the pass-band, $H(\omega_t, \omega_v)$ be the filter we try to design. The Eigenfilter approach finds:

$$\begin{aligned} \arg\min_{H(\omega_t,\omega_v)} E &= \arg\min_{H(\omega_t,\omega_v)} \left\{ \alpha E_P + \beta E_S \right\} \\ &= \arg\min_{H(\omega_t,\omega_v)} \left\{ \alpha \iint_P \left| D(\omega_t, \omega_v) - H(\omega_t, \omega_v) \right|^2 d\omega_t d\omega_v \right. \\ &\quad \left. + \beta \iint_S \left| H(\omega_t, \omega_v) \right|^2 d\omega_t d\omega_v \right\} \end{aligned} \tag{5.1}$$

where E is the overall square error measured by the weighted sum of the pass-band error E_P and the stop-band error E_S; α and β are weighting constants which control the accuracies of the approximation. In a normal setup, we often have $D(\omega_t, \omega_v) = 1$ within the pass-band.

Eigenfilter is optimal for filter design itself, but it is not necessarily optimal in terms of

the reconstruction error for a certain signal. Let the down-sampled signal spectrum (such as Figure 5.2(b) and (c)) be $X(\omega_t, \omega_v)$. We try to find the optimal reconstruction filter through:

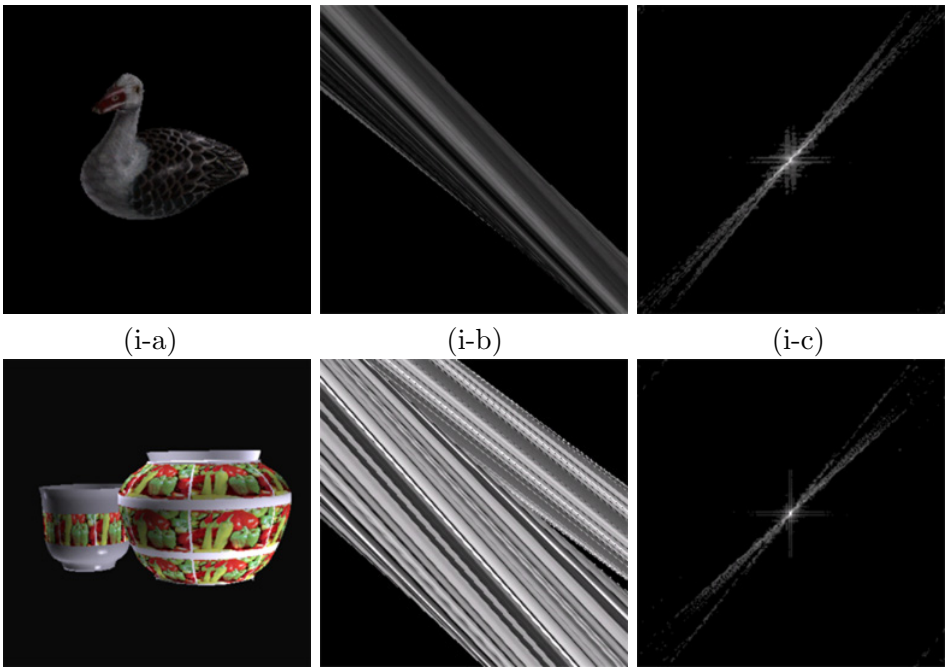
$$\begin{aligned}
\arg\min_{H(\omega_t,\omega_v)} E^r &= \arg\min_{H(\omega_t,\omega_v)} \left\{ E_P^r + E_S^r \right\} \\
&= \arg\min_{H(\omega_t,\omega_v)} \left\{ \iint_P |X(\omega_t,\omega_v)|^2 \left| D(\omega_t,\omega_v) - H(\omega_t,\omega_v) \right|^2 d\omega_t d\omega_v \right. \\
&\quad \left. + \iint_S \left| X(\omega_t,\omega_v) \right|^2 \left| H(\omega_t,\omega_v) \right|^2 d\omega_t d\omega_v \right\}, \qquad (5.2)
\end{aligned}$$

where E^r is the overall reconstruction error, E_P^r and E_S^r are the reconstruction errors in pass-band and stop-band, respectively. Equation 5.2 can still be solved through the eigenfilter approach, as it differs from Equation 5.1 only by a weighting function. Notice that the optimal filter is related with the signal $X(\omega_t, \omega_v)$. In our experiments in the next section, we show that by using the Fourier transform of a first order auto-regressive (AR-1) signal to model $X(\omega_t, \omega_v)$, we get better reconstruction than the regular eigenfilter approach.

5.2 Experimental Results

We next show some experimental results on the two different sampling approaches. In order to have full control on the scenes and the cameras, we choose two scenes rendered from 3D models with texture. These scenes are shown in Figure 5.3, where scene (i) and (ii) are named *Duck* and *Containers*, respectively. We take the center horizontal lines to construct the epipolar images (EPIs). Figure 5.3(i-a) and (ii-a) are snapshots of the scenes; (i-b) and (ii-b) are their EPIs; (i-c) and (ii-c) are the Fourier transform of the EPIs. Although occlusions can be observed in the scenes, we ignore them in our analysis since they are not significant in these two examples.

From the Fourier transform of the scenes in Figure 5.3, we can find the corresponding sampling rate and compact the spectrum to Figure 4.6(b) and Figure 5.1(b). One thousand random images are then rendered for each scene with different reconstruction filters. These images are also synthesized through a 3D model rendering engine. The difference between



(i-a) (i-b) (i-c)

(ii-a) (ii-b) (ii-c)

Figure 5.3: Test scenes for generalized sampling of light field. (i) The scene *Duck*; (ii) the scene *Containers*; (a) the scene snapshot; (b) the EPI constructed from the center horizontal slits; (c) Fourier transform of (b).

the synthesized images and the rendered images is used to measure the quality of the sampling process. In our experiments, PSNR is used to measure such differences.

The experimental results are shown in Table 5.1. We test the regular eigenfilter (REF) and the optimal reconstruction filter (ORF) presented in the last section for both RS and GS. To design the ORF, we model each replica of the Fourier spectrum as the Fourier transform of an AR-1 signal with $\rho = 0.9$ along ω_v, and constant along ω_t. For example, the replica centered at $\omega_v = \omega_t = 0$ is represented as:

$$\left|X(\omega_t, \omega_v)\right|_P = \left|\frac{1}{1 - \rho e^{j\omega_v}}\right| \tag{5.3}$$

where the subscript P simply means that it is valid only in the pass-band. We also list depth-driven bilinear interpolation for RS in Table 5.1 for comparison. Several conclusions can be drawn from the above table. RS REF performs much worse than GS REF. This is because the specification for RS REF design has a zero transition-band at high frequency

Table 5.1: Rendering image qualities for different sampling methods and reconstruction filters (PSNR: dB).

Approach	***Duck***	***Container***
RS, regular eigenfilter	33.99	20.44
RS, optimal reconstruction filter	35.89	21.81
GS, regular eigenfilter	35.68	22.04
GS, optimal reconstruction filter	36.06	22.21
RS, bilinear interpolation	36.10	21.67

components, which is hard to design. In all cases, ORF is significantly better than designing a general-purpose filter. This is because extra knowledge was employed during the ORF design process. With ORF, the difference between RS and GS becomes very small, which again shows the power of ORF. Interestingly, the simple approach that uses RS with bilinear interpolation gives comparable performance as GS with optimal filter for reconstruction. This is unexpected but was verified in some other scenes we tested. Comparing the two test scenes, there is more improvement by using GS for the scene *Containers*, because *Containers* has more high frequency components in its spectrum. This is consistent with our analysis in the last section.

Real world scenes often has weak high frequency components and strong low frequency components. Thus for a typical light field, RS is usually more suitable than GS. Even for the *Containers* scene, the improvement by using GS is minor, which cannot justify the increased complexity in GS. Since GS is inconsistent with how the images are taken, the required re-sampling may introduce extra error. The rendering speed is another concern. Unless we build a huge look-up table for the reconstruction filter and always do simple rounding when searching for a filter value, bilinear interpolation is required to get the filter values at arbitrary viewpoints. Even if the designed reconstruction filter has the same size of support as bilinear interpolation, the filter interpolation will slow down the rendering by a factor of 4. Therefore, we conclude that in practice rectangular sampling is preferable to non-rectangular sampling for light field.

5.3 Summary

In this chapter, we presented the idea of applying generalized sampling theory to light filed. When there is no occlusion and the scene is Lambertian, we showed that in theory the sampling density in the spatial domain could be half of that when we use rectangular sampling. We found that using such theory we may achieve higher reconstruction quality at the same sampling rate when the scene has much high frequency components. However, for typical scenes, such improvement cannot justify the complexity increase. We conclude that rectangular sampling and bilinear interpolation is still the most preferable approach for light field.

Chapter 6

Freeform Sampling and Active Sampling: A New Sampling Framework

In Chapter 4 and 5, we have studied the uniform sampling of image-based rendering data using the traditional sampling analysis method. Such analysis may provide a general guidance on how a scene should be sampled. For instance, if a scene has non-Lambertian surface or occluded regions, more samples are needed. However, in applications such as IBR, uniform sampling can be a big constraint on the sampling efficiency one can achieve. In general, a real world scene is composed of Lambertian and non-Lambertian surfaces. The Lambertian surface may need a low sampling rate, while the non-Lambertian surface may need a high sampling rate. Uniformly sampling the scene without concerning about the regional surface property may easily cause over-sampling of the Lambertian surface or under-sampling of the non-Lambertian surface.

In this chapter, we propose a new framework for sampling, namely the *freeform sampling* framework. In contrast to the traditional uniform sampling approach, where samples have to be taken according to a fixed sampling pattern, freeform sampling allows the sample locations to be irregular or nonuniform. Such freedom may greatly improve the sampling efficiency, which allows a signal to be reconstructed from much fewer samples.

6.1 The Freeform Sampling Problem

Consider a function in the form of $y = f(\mathbf{x}), \mathbf{x} \in \mathbf{X}$, where $\mathbf{X}$ is the region of support (ROS). Here $\mathbf{x}$ can be a variable, a vector, a matrix, or anything else that is meanful. For example, in the scenario of IBR sampling, y is the light rays captured at $\mathbf{x}$, where $\mathbf{x}$ may include a 3D vector representing the camera position as well as a 2D vector representing the directions of the light rays.

To find out what the unknown function $f(\mathbf{x})$ is, one may take a set of samples in the ROS $\mathbf{X}$. Let the *sample set* be $\mathbf{X}^s \subset \mathbf{X}$. In freeform sampling, this sample set can be irregularly distributed. The *sample set values*, denoted as $\tilde{f}(\mathbf{X}^s)$, may subject to some noise $n(\mathbf{X}^s)$:

$$\tilde{f}(\mathbf{X}^s) = f(\mathbf{X}^s) + n(\mathbf{X}^s). \tag{6.1}$$

Given such a sample set and its values, the original signal can be reconstructed. For any given $\mathbf{x} \in \mathbf{X}$, we denote the reconstructed signal value as:

$$\hat{f}\Big(\mathbf{x}\Big|\big(\mathbf{X}^s, \tilde{f}(\mathbf{X}^s)\big), \mathcal{R}\Big), \tag{6.2}$$

where $\mathcal{R}$ is the reconstruction method. Notice that when we choose different reconstruction methods, the above equation may present different results. In many cases, the reconstructed signal may be different from the original function value, thus an error function $e(\mathbf{x}), \mathbf{x} \in \mathbf{X}$ can be defined. The concrete form of the error function is user or application dependent. For example, one possible error function is the squared error:

$$e(\mathbf{x}) = \left\| f(\mathbf{x}) - \hat{f}\Big(\mathbf{x}\Big|\big(\mathbf{X}^s, \tilde{f}(\mathbf{X}^s)\big), \mathcal{R}\Big) \right\|^2. \tag{6.3}$$

The general freeform sampling problem can thus be defined as:

Definition 6.1.1. *Given a function $y = f(\mathbf{x}), \mathbf{x} \in \mathbf{X}$, which is either known or unknown, find a set of sample locations $\mathbf{X}^s \subset \mathbf{X}$, such that the reconstruction of y from the sample set $\mathbf{X}^s$ using method $\mathcal{R}$ meets certain error requirement $\mathcal{Q}\big(e(\mathbf{x})\big)$ on a* reconstruction set $\mathbf{X}^r \subseteq \mathbf{X}$.

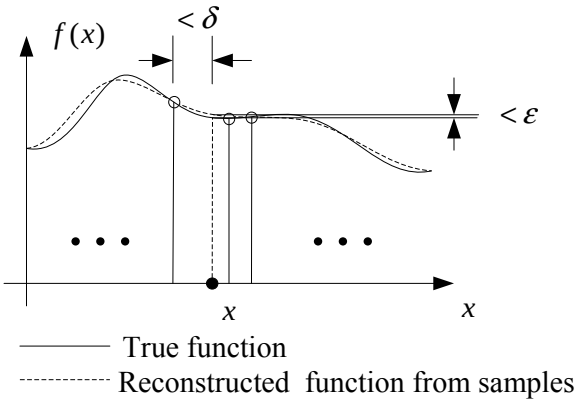


Figure 6.1: Illustration of Condition 6.1.2 with a 1D example.

$\mathbf{X}^r$ can be the whole ROS $\mathbf{X}$ or a subset of it. This error requirement is again user or application dependent. For instance, we may let $\mathcal{Q}\big(e(\mathbf{x})\big)$ be:

$$e\Big(\mathbf{x}\Big|\big(\mathbf{X}^s, \tilde{f}(\mathbf{X}^s)\big), \mathcal{R}\Big) < \varepsilon, \forall \mathbf{x} \in \mathbf{X}^r, \tag{6.4}$$

in which case we know that the reconstruction error in the reconstruction sample set should be bounded by ε.

The definition of freeform sampling above differs from the traditional uniform sampling problem in many ways. First and most distinguishably, the sample set $\mathbf{X}^s$ is in freeform instead of being uniformly distributed. Second, we consider the reconstruction method as an important factor which will affect the sampling process, because different reconstruction methods may produce different reconstructions. Third, when we take the sample, there is a noise term. In practice, observations always have noises, and these noises cannot be easily eliminated. Lastly, we introduce the reconstruction sample set $\mathbf{X}^r$. If $\mathbf{X}^r \neq \mathbf{X}$, the best sampling strategy can also be set $\mathbf{X}^r$ dependent. This inspires our view-dependent freeform sampling of IBR in Section 7.3.1.

Another important thing to notice, is that given the reconstruction method, since infinitely many functions can have the same value on a sample set $\mathbf{X}^s$ and meet the error requirement, the above sampling problem is meaningful only by adding constraints on $f(\mathbf{x})$. In the traditional uniform sampling theory, a common constraint is that $f(\mathbf{x})$ has band-limited Fourier spectrum [143, 3]. That is, the Fourier transform of $f(\mathbf{x})$ is non-zero only

on a finite support. As Fourier transform does not maintain any local information in the spacial domain, such a constraint is not suitable for freeform sampling. Instead, we propose to use local constraints to limit the solution space. More specifically, we constrain that:

Condition 6.1.2. *$\forall \varepsilon, \mathbf{x} \in \mathbf{X}^r$, $\exists \delta$, such that as long as $d\big(\mathbf{x}, \Phi_N(\mathbf{x})\big) < \delta$, $\mathcal{Q}\big(e(\mathbf{x})\big)$ is satisfied.*

Here $\Phi_N(\mathbf{x}) \subset \mathbf{X}^s$ is the nearest N neighbors of $\mathbf{x}$ in $\mathbf{X}^s$. $d\big(\mathbf{x}, \Phi_N(\mathbf{x})\big)$ is the maximum distance from $\mathbf{x}$ to these neighbors:

$$d\big(\mathbf{x}, \Phi_N(\mathbf{x})\big) = \max_{\mathbf{x}_i^s \in \Phi_N(\mathbf{x})} \|\mathbf{x} - \mathbf{x}_i^s\| \tag{6.5}$$

The neighborhood size N is often application dependent. The above condition is illustrated with a 1D example in Figure 6.1. In this example $N = 3$. Condition 6.1.2 states that as long as the maximum distance from a reconstruction sample x to its 3 closest neighboring samples is less than a threshold δ, the reconstructed function value is guaranteed to have an error less than ε.

The above condition implies that the local variation of function $f(\mathbf{x})$ is limited. If we sample the signal at a sampling density that is high enough, we are guaranteed to satisfy the sampling error requirement. On the other hand, it is permissible to take samples at a lower density, as long as the error requirement is fulfilled. This gives us the freedom of performing nonuniform sampling of the signal.

6.2 Solutions of the Freeform Sampling Problem

Depending on the application scenario, solutions to the above freeform sampling problem can in general be classified into three categories: decremental sampling, incremental sampling and rearranged sampling.

6.2.1 Decremental sampling

In decremental sampling, we assume there is already a dense set of samples available that fulfills the error requirement. This sample set might be too large, thus decremental sampling

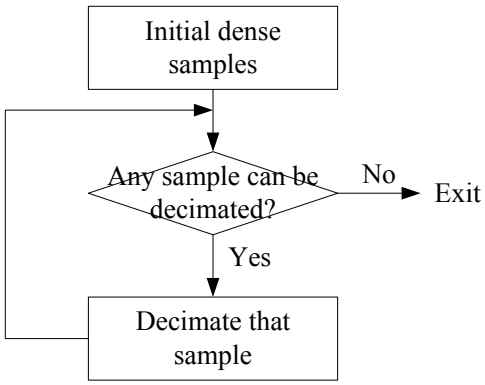


Figure 6.2: The flow of decremental sampling.

can be used to reduce the size, yet still keep the error within the requirement. Figure 6.2 shows the general flow of decremental sampling. Starting with the dense set of samples, decremental sampling first identifies which sample can be removed without having problem with the error requirement. If such sample exists, it decimates this sample, and continue with the next one. Otherwise, it exits.

One example of decremental sampling is the JPEG image compression standard [108]. In JPEG, blocks of images are first transformed by DCT, followed by quantization and entropy coding. Only the DCT coefficients that are non-zero after quantization will be coded into the bitstream, and all the other coefficients will be decimated. In this example, the reconstruction set is all the DCT coefficients, and the sample set is the remained coefficients after quantization (quantization introduces sample noise here). The error requirement is that the difference between the reconstructed DCT coefficients and the original ones are less than the quantization step size. Therefore this is a decremental sampling scheme in the DCT domain. Similar ideas have been widely used in the compression of video [92], mesh model [46, 68], Lumigraph [163], etc.

In more general sense, feature selection is another example. In pattern classification, one often need to reduce the dimension of the feature vector in order to avoid the curse of dimensionality [30] or to integrate prior knowledge. For instance, in face recognition, the well-known eigenface approach [142] obtained a low dimensional feature vector from face

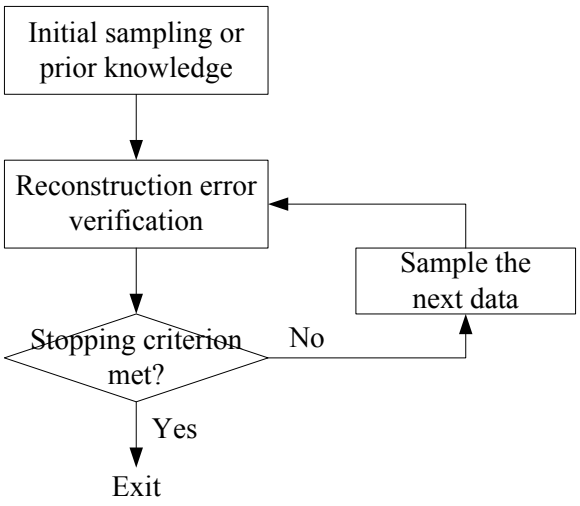


Figure 6.3: The flow of incremental sampling.

images which may have thousands of pixels through principle component analysis (PCA). The selection of eigenfaces is a process of decremental sampling in the sense that after PCA it only keeps several dominate eigenvectors.

6.2.2 Incremental sampling

In incremental sampling, the samples are taken one by one incrementally. The stopping criterion is either a limit on the overall number of samples one can take, or an error requirement that the sampling process must achieve.

As shown in Figure 6.3, the general flow of incremental sampling starts with an initial sampling, which might be uniform or random following a certain prior probability distribution. This initial sampling may also be skipped if other prior knowledge is available. Given the set of initial samples or prior knowledge, it then verifies the reconstruction error on the reconstruction sample set. If the error requirement is not met on some of the reconstruction samples, more samples will be taken in the next step to improve the reconstruction of these samples. This process loops until the error requirement is met or the maximum allowable samples have been taken.

In the recent wavelet based image compression algorithms such as the SPIHT algorithm

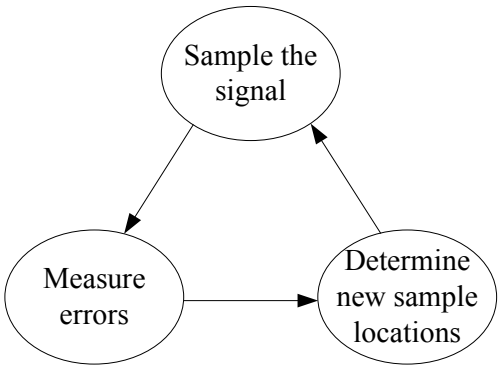


Figure 6.4: The flow of rearranged sampling.

[118] and the new JPEG 2000 standard [137], the wavelet coefficients are encoded bitplane by bitplane. Therefore, coefficients with greater magnitudes will be sampled first, and those with smaller magnitudes will be encoded later. This can be considered as an incremental sampling scheme which samples important coefficients first and tries to minimize the energy difference between the original image and the encoded one. A nice property of incremental sampling is that one can stop at any time during the sampling process, thus here the compressed bitstream is embedded or scalable.

6.2.3 Rearranged sampling

In rearranged sampling, the total number of samples is limited. The goal of rearranged sampling is to position these samples at their best locations, such that certain accumulated reconstruction error is minimized on the reconstruction sample set, or the reconstruction error satisfies a given condition.

The flow of rearranged sampling is shown in Figure 6.4. Again we start with sampling the signal. Given the sampled data, the error of the reconstruction sample set is analyzed. After such analysis, rearranged sampling identifies a new set of sample locations which may possibly reduce the accumulated reconstruction error or cause the error to fit better to the given distribution. It then samples the data at these new locations, which completes a loop. For a static signal, the above loop repeats until the sample locations do not change any more, which results in the most critical set of samples that can be used to represent the

whole signal. For a dynamic signal, the above loop repeats until the end of the capturing, which ought to provide better reconstruction quality than a passive sampling approach. Obviously, for dynamic signals, the relocation of the samples should be faster than the signal changes, such that it is reasonable to use the error analysis performed for the last sampling to determine the new sample locations.

Vector quantization (VQ) can be considered as a very good example of rearranged sampling. The goal of VQ is to represent a large set of data points with a small set of codewords. Therefore the codewords form the sample set, the large data set is the reconstruction set. The reconstruction method is to use the nearest codewords to represent the points in the reconstruction set. The error requirement is that such reconstruction has minimum accumulated error. Furthermore, the popular LBG algorithm [73], which is an iterative algorithm to solve VQ, matches very well with the general flow of rearranged sampling in Figure 6.4. One thing to mention is, depending on the initialization, the LBG algorithm may be trapped in local minimum during the iteration, so do all the other VQ algorithms. Therefore, rearranged sampling in general may not always achieve the optimal result due to such inherent problems, but it still has a fairly large chance to beat a fixed sampling pattern strategy such as uniform sampling, if the initialization is reasonable.

6.3 Active Sampling

In the examples we presented in the last section, we know the function values on the reconstruction set. For example, in image compression, the original image (or its transformed version) is the reconstruction set and we know it before sampling. This allows us to easily calculate the reconstruction error given any sample set, and verify if the reconstruction error fulfills the error requirement. Unfortunately, in some other applications (see later in this section for examples), the function values on the reconstruction set is unknown. Therefore to perform freeform sampling, one must estimate what is the reconstruction error and guess if the error requirement has been fulfilled. Such prediction can only be performed based on samples that have been taken and certain prior knowledge one might have. We term this

kind of freeform sampling strategy as *active sampling*. The general solutions of incremental sampling and rearranged sampling is still valid in active sampling. Decremental sampling, however, is not applicable any more because we do not have the starting dense sample set.

Despite the difficulty in estimating the reconstruction errors, active sampling have been used in several applications in the literature. There are some shared characteristics of these applications. First, the to-be-sampled function is often unpredictable, thus no fixed sampling pattern can be used to guarantee that the error requirement will be fulfilled, unless a very dense sampling is used. Second, taking a sample in these applications is often very expensive or time consuming. To save the cost or to speed up the sampling process while still fulfilling the error requirement, active sampling becomes the best choice. Below we list some of these application that employs active sampling.

Next best view In automated surface acquisition, using range sensors to obtain the object surface is a labor intensive and time-consuming task. Moreover, no fixed sampling pattern can be used because the captured objects often have irregular shape. The main purpose of the next best view (NBV) algorithm is to ensure that all scannable surfaces of an object will be scanned, and determine when to stop scanning [110, 116]. Incremental sampling was used in NBV, where one often determines the next sampling position based on the 3D geometry reconstructed and merged from the previous samples. The error function predicted is usually the holes exhibited in the current geometric model.

Active learning For many types of machine learning algorithms, one can find the statistically "optimal" way to select the training data. The pursuing of the "optimal" way by the machine itself was referred to as active learning [21, 58, 44]. For example, a classification machine may determine the next training data as the one that is most difficult for it to classify. Notice in this application, the sample set is the training data, the reconstruction set is the objects in the whole database (or maybe some cross-validation set), the value of each sample is the label of classification. The error function is predicted as the certainty of the classification. Recently active learning has been applied to the annotation of database

for retrieval [157], which can save a lot of human laboring or annotation.

Motion estimation Motion estimation is an important component in modern video compression algorithms [92]. A full search of the motion vector can provide the best motion vectors, but it may be too slow. Various other search algorithms have been proposed, among them the most famous ones are such as the three-step search (TSS) [57] and the four-step search (FSS) [60] algorithms. These searching algorithms first measure the matching error for a subset of possible motion vectors. Only the ones that are promising to be the true motion vector will be refined (more motion vectors will be tested around them and the best one among the tested will be chosen as the final result). This is a good example of sampling—analyzing—sampling loop in incremental sampling.

6.4 The Proposed Algorithms for Active Sampling

When the values of the reconstruction set are known, one can easily measure the reconstruction error, thus the performance of freeform sampling can be largely guaranteed. However, in active sampling such reconstruction error must be estimated. The key to the success of active sampling thus lies in how good we can perform such error prediction.

Since the form of reconstruction error varies from application to application, it is generally not possible to present an algorithm that is suitable to all applications. In this section, we make certain assumptions about the applications and present some algorithms that are widely applicable, particularly for the case of image-based rendering.

6.4.1 Active Incremental Sampling

We first add some structure to the incremental sampling process. Assume neighboring samples in the sample set can be organized into cliques, as shown in Figure 6.5. Here we give three representative cliques: rectangle, triangle and sample pair. As samples in the same clique are close to each other, we assume their sample values should share certain local property. If, for some reason, we believe the local property is not fulfilled, we shall

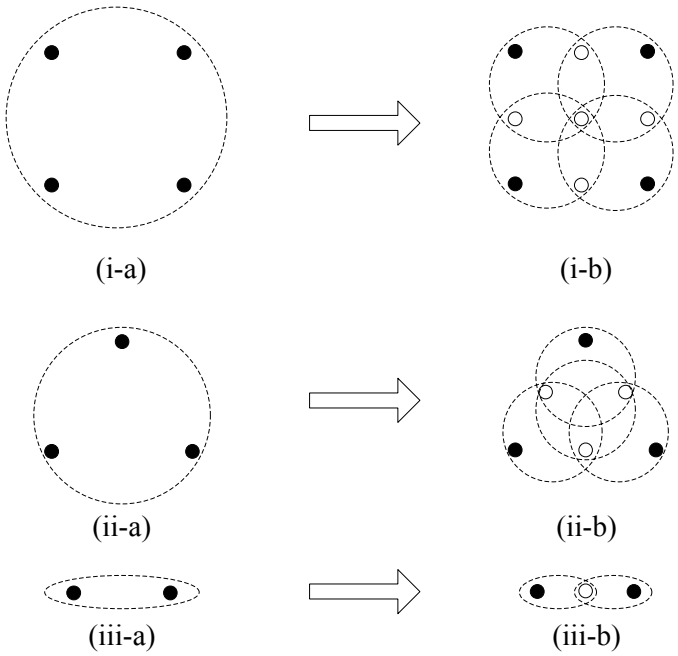


Figure 6.5: Cliques and their subdivision. (i) rectangle; (ii) triangle; (iii) sample pair; (a) original clique; (b) subdivided clique.

increase the sampling rate by subdividing the clique to smaller ones, as shown in Figure 6.5(b). This incremental sampling structure naturally leads to nonuniform sampling of the sampled signal. Notice in the above incremental sampling procedure, the local property of the cliques is the key in determining the final sample set. As in active sampling, the final goal is to fulfill the error requirement on the reconstruction set, we shall associate the local property of cliques with the reconstruction method and the reconstruction set.

In many practical applications, it is preferable to have a reconstruction method that is simple, so that the reconstruction can be performed in real time. Here simplicity is two-fold. First, if the function value at a certain location is reconstructed, only a few samples nearby will be involved. Second, the algorithm used for reconstruction is often as simple as some weighted interpolation of the nearby samples. The assumption behind such a reconstruction method, is that albeit a signal can vary violently in large scale, within a small neighborhood it should change slowly.

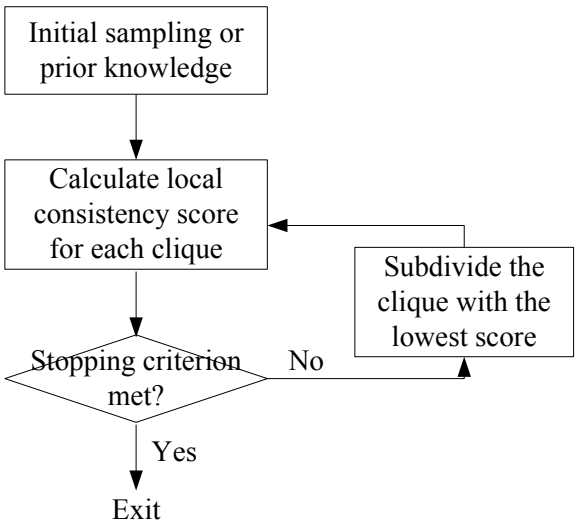


Figure 6.6: The flow of active incremental sampling.

Under the same assumption above, we hereby claim the following *local consistency principle* for active sampling:

Principle 6.4.1. *Given a to-be-reconstructed sample location, the consistency of the sample values of the nearby samples may serve as a good indicator of its reconstruction error.*

In particular, the consistency of the sample values in a clique may be used to determine whether it should be subdivided or not. Several things should be noticed in the above principle. First, the concrete form of local consistency between samples may still vary from application to application. Second, the above principle applies only when some to-be-reconstructed sample location is given. If a clique of samples is never used in reconstructing any samples in the reconstruction set, it shall never be subdivided no matter how inconsistent they are. Third, although the local consistency is a good indicator, it cannot be 100% accurate due to the spacing between samples and the sample noise introduced in Equation 6.1. Therefore, the sample noise and the size of the clique should both be considered while measuring the local consistency. For instance, The accuracy often improves when the samples in the clique get closer.

We therefore detail the flow chart of incremental sampling in Figure 6.3 with the clique

sampling structure and the local consistency principle above in Figure 6.6. After the initial sampling, we calculate for each clique its local consistency score. Afterwards, we test if any stopping criterion has been met, e.g., if we have reached the specified maximum number of samples, or if all the cliques have score higher than a certain threshold. If such stopping criterion is not met, we subdivide the clique with the lowest score. This at least helps in reducing the worst case error if the local consistency reflects the error well enough.

6.4.2 Active Rearranged Sampling

In rearranged sampling, the number of samples in the sample set is fixed. One must move these samples so that certain error requirement is fulfilled on the reconstruction set. For instance, one may require that the accumulated reconstruction error be minimized, or that the worst reconstruction error be minimized. In both cases, rearranged sampling can be solved using a technique similar to vector quantization.

Assume at a certain instance, we have captured a set of samples in the sample set. The reconstruction errors of the samples in the reconstruction set can thus be estimated by the local consistency principle. Samples with lower consistency scores may have larger reconstruction error, thus some sample set samples should move toward them to form a denser sampling. In contrast, if some reconstruction set samples have high consistency scores, the sample set samples could be relocated away from them to form a sparser sampling. The final result of such rearrangement is to have the reconstruction error close to uniform everywhere.

Depending on the distribution of the local consistency scores of the reconstruction set samples, we discuss two algorithms for active rearranged sampling: local rearrangement and global rearrangement.

In local rearrangement, we assume that the local consistency scores of the reconstruction set samples have been close to uniform. Only small local adjustments need to be performed on the sample set. The movement of the sample set samples resembles the movement of the centroids in the LBG vector quantization algorithm [73]. That is, for each sample in the

sample set, we consider all the samples in the reconstruction set that treat it as one of their neighboring samples. The local consistency scores of them will determine the new location of that sample set sample.

Sometimes when we give an initial sampling in rearranged sampling, the local consistency scores may vary a lot. It may take the above local rearrangement algorithm quite a while to converge to some result that is reasonable. To avoid such situation, we may choose to perform a global rearrangement. Essentially, a global rearrangement is a recast of the samples in the sample set by considering both their current position and the local consistency score distribution. Unlike the local rearrangement algorithm above, where samples in the sample set often move with small size steps, a global rearrangement algorithm may move some samples dramatically just to increase the sampling density at certain regions that are highly inconsistent.

6.5 Summary

In this chapter, we have presented a framework for freeform sampling and active sampling. Freeform sampling, as the name implies, allows the sample locations to be distributed in freeform instead of a regular uniform pattern. This gives us the possibility to efficiently sample signals that are highly complex or non-stationary, which may save many samples from the traditional uniform sampling. The freeform sampling approaches are often based on measuring the reconstruction errors of samples in a given reconstruction set. If such error measuring is not available, active sampling takes the place, which performs freeform sampling using estimated errors. We described several active sampling algorithm, which often aim at reducing the worst case reconstruction errors. In the next chapter, we will show several examples in applying active sampling to the application of image-based rendering.

Chapter 7

Active Sampling: Applications in IBR

In this chapter we will apply the active sampling framework in Chapter 6 for image-based rendering. We first discuss the local consistency score of IBR applications in Section 7.1. We then discuss both active incremental sampling and active rearranged sampling in various IBR applications, such as the light field and the concentric mosaics. We show that active sampling outperforms uniform sampling in all the scenes we tested.

7.1 The Local Consistency Score

The key to active sampling, is how to define the local consistency score for estimating the reconstruction errors. We mentioned that such consistency score should be related to the reconstruction method, the reconstruction set, the sample noise, and the confidence of applying the consistency measure as the reconstruction error, etc.

The most widely used reconstruction method in IBR is through weighted interpolation of nearby captured light rays [66, 128, 11]. In [11], eight goals were proposed for IBR, which led to a weighted interpolation algorithm that considers angular difference, resolution sensitivity and field of view (FOV). As shown in Figure 7.1, let OP be a light ray to be rendered. Cameras $C_1, C_2, \cdots, C_K$ are the K nearest cameras to that light ray in terms of their angular difference to OP, which are denoted as $\alpha_1, \alpha_2, \cdots, \alpha_K$. Typically $K = 4$

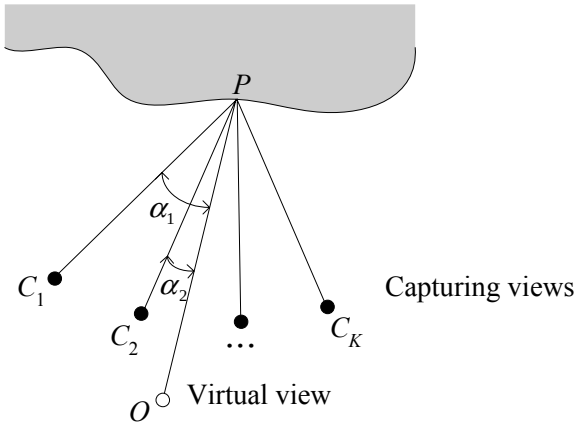


Figure 7.1: Interpolation weight calculation using angular difference.

is good enough. Light rays $C_kP, k = 1, 2, \cdots, K$ will then be used to interpolate OP. We define the weight for angular difference as:

$$w_k^{ang} = \frac{1}{\epsilon + \alpha_k}, k = 1, 2, \cdots, K \tag{7.1}$$

where ϵ is a small number (e.g., 10^{-10}) to avoid division by zero. If the 3D point P projects to image captured by C_k as (x_k, y_k), the weight for FOV $w_k^{fov}, k = 1, 2, \cdots, K$ is defined as 1 if (x_k, y_k) is inside the FOV, and 0 if it's outside. A narrow region is defined along the FOV boundary to create a smooth transition from 1 to 0. In our current implementations, since the capturing cameras basically lie on a circle as in the concentric mosaics or a plane as in the light field, the weight for resolution can be ignored and fixed as $w_k^{res} \equiv 1, k = 1, 2, \cdots, K$. The overall weight for the light ray C_kP is the multiplication of the three factors:

$$w_k = w_k^{ang} \times w_k^{fov} \times w_k^{res}, k = 1, 2, \cdots, K. \tag{7.2}$$

These weights are then normalized to make sure they sum up to 1 for a given light ray OP. Although our weight definition is different from that in [11], which used a penalty-based weighting scheme, our experiments show that the rendering quality is not sensitive to how the weights are specified, as long as it satisfies the other more basic requirements such as continuity and epipole consistency [11].

The local consistency score of the light rays $C_kP, k = 1, 2, \cdots, K$ can be defined in various ways [126, 10]. Let their color intensity be $l_k, k = 1, 2, \cdots, K$. The simplest yet

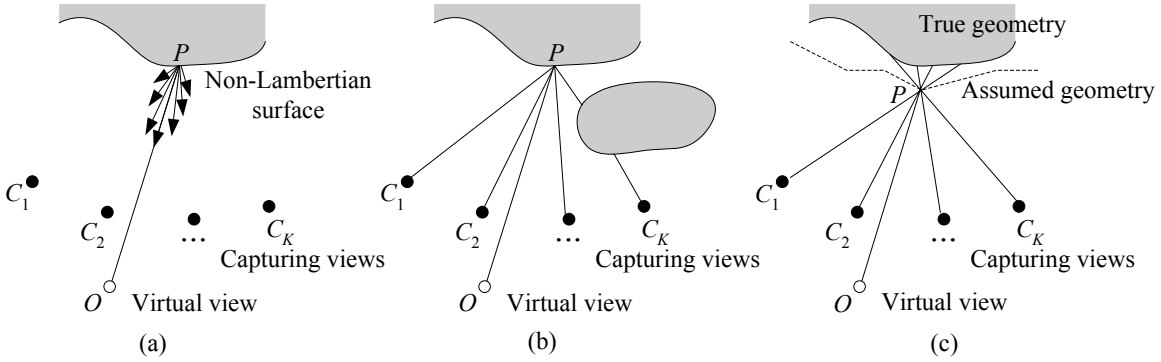


Figure 7.2: Cases that will cause a low local consistency score. (a) Non-Lambertian surface; (b) occlusions; (c) inaccurate geometry.

widely used technique is to make use of their intensity variance σ:

$$\mathcal{C} = \frac{1}{\sigma}, \text{ where } \sigma = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(l_k - \bar{l})^2}, \tag{7.3}$$

here $\bar{l} = \frac{1}{K}\sum_{k=1}^{K} l_k$ is the average intensity of l_k's. The above measure essentially assumes the surface is Lambertian and not occluded, and reflects the likelihood of the light ray intensities being independent and identical Gaussian distribution as a stochastic variable. On the other hand, under several situations one will obtain a low consistency score: non-Lambertian surface, occluded objects or inaccurate geometry, as shown in Figure 7.2. According to the uniform sampling theory in Chapter 4, all these cases in fact requires a sampling rate that is higher than normal. In active sampling, regions with low consistency score implies higher reconstruction error and will be sampled denser, thus it is consistent with the previous conclusions.

If one particular light ray's reconstruction error is concerned, such as the light ray OP in Figure 7.1, we may apply a slightly modified variation assuming the sensor noise is negligible:

$$\mathcal{C} = \frac{1}{\sigma'}, \text{ where } \sigma' = \sqrt{\sum_{k=1}^{K} w_k(l_k - \bar{l}')^2}, \tag{7.4}$$

here $\bar{l}' = \sum_{k=1}^{K} w_k l_k$ is the weighted average intensity. Notice the w_k's should have been normalized to sum to 1. Compared with the standard variance definition in Equation 7.3,

the above weighted variation takes the reconstruction method into consideration, and may better reflect the actual interpolation quality of the given light ray for non-Lambertian or occluded objects.

The confidence of applying the consistency measure as the reconstruction error is the most difficult thing to integrate into active sampling. It requires certain prior knowledge about the sampled signal. Our solution is to adjust the consistency score obtained in Equation 7.3 and 7.4 with a penalty multiplication factor $f(\alpha)$, where α can be the average angular difference $\alpha = \frac{1}{K}\sum_{k=1}^{K}\alpha_k$. The larger the α, the smaller the multiplication factor $f(\alpha)$, the lower the resultant consistency score. $f(\alpha)$ can be considered as a tuning factor which prevents the active sampling from being too aggressive at some local region, thus increases the robustness of active sampling. It can also be thought of as determining the tradeoff between uniform sampling and active sampling, because if $f(\alpha)$ gives too much penalty for large α, the final sampling result will tend to have the same α everywhere, which becomes uniform sampling. In the following discussions, we will not worry about the above problem and show only the raw performance of active sampling.

7.2 IBR Active Incremental Sampling

In this section, we apply active incremental sampling on IBR. We will use the light field and the concentric mosaics as examples.

7.2.1 Active incremental sampling for the light field

In the original light field setup [66], the cameras are positioned on a regular grid of the camera plane. For active sampling, we still assume that the cameras are on the camera plane, but they can be arranged non-uniformly. As shown in Figure 7.3, assume that the capturing cameras are within a rectangular range determined by $(0, 0)$ and (s_{max}, t_{max}). We initialize the active capturing process by a reasonably dense uniform sampling. We use the rectangular clique in Figure 6.5(i) for the subdivision process. That is, each time when we capture some new images, we subdivide one of the cliques into four. In the example shown in

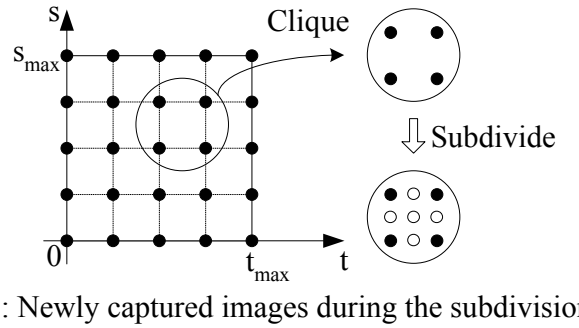


Figure 7.3: Active incremental sampling for the light field setup.

Figure 7.3, five new images are taken during the subdivision. It is always the clique that has the lowest accumulated consistency score of all the corresponding light rays (Equation 7.3) is subdivided, thus in this example we improve the worst case rendering quality through active sampling. The sampling process recursively performs the above subdivision until certain limit on the number of images is reached or the local consistency scores of all the cliques have been smaller than a given threshold.

The above active incremental sampling strategy is tested on a synthetic scene *Earth*, as shown in Figure 7.4(a). *Earth* is a near-Lambertian scene, whose geometry is known and represented as a 96×96×64 volumetric model. We initialize our active incremental sampling algorithm by 7×7 uniform sampling, and the overall number of images is limited to be less than or equal to 169. The result is compared to a 13×13 uniform sampling approach. Figure 7.4(b) shows the final camera arrangements on the camera plane using active incremental sampling. For comparison, (c) shows that of uniform sampling. Each dot represents a camera being there and taking one image. It can be observed that active incremental sampling puts more cameras at the top-right portion of the camera plane. Figure 7.4(d) is an example view captured in active incremental sampling (red circled in (b)). Uniform sampling did not sample that view. Figure 7.4 (e) is what can be rendered from the sampled images in uniform sampling. As a comparison, Figure 7.4 (f) is a view captured in uniform sampling (red circled in (c)). It is not captured in active incremental sampling but can be rendered as (g). Obviously we would prefer to sample (d) instead of

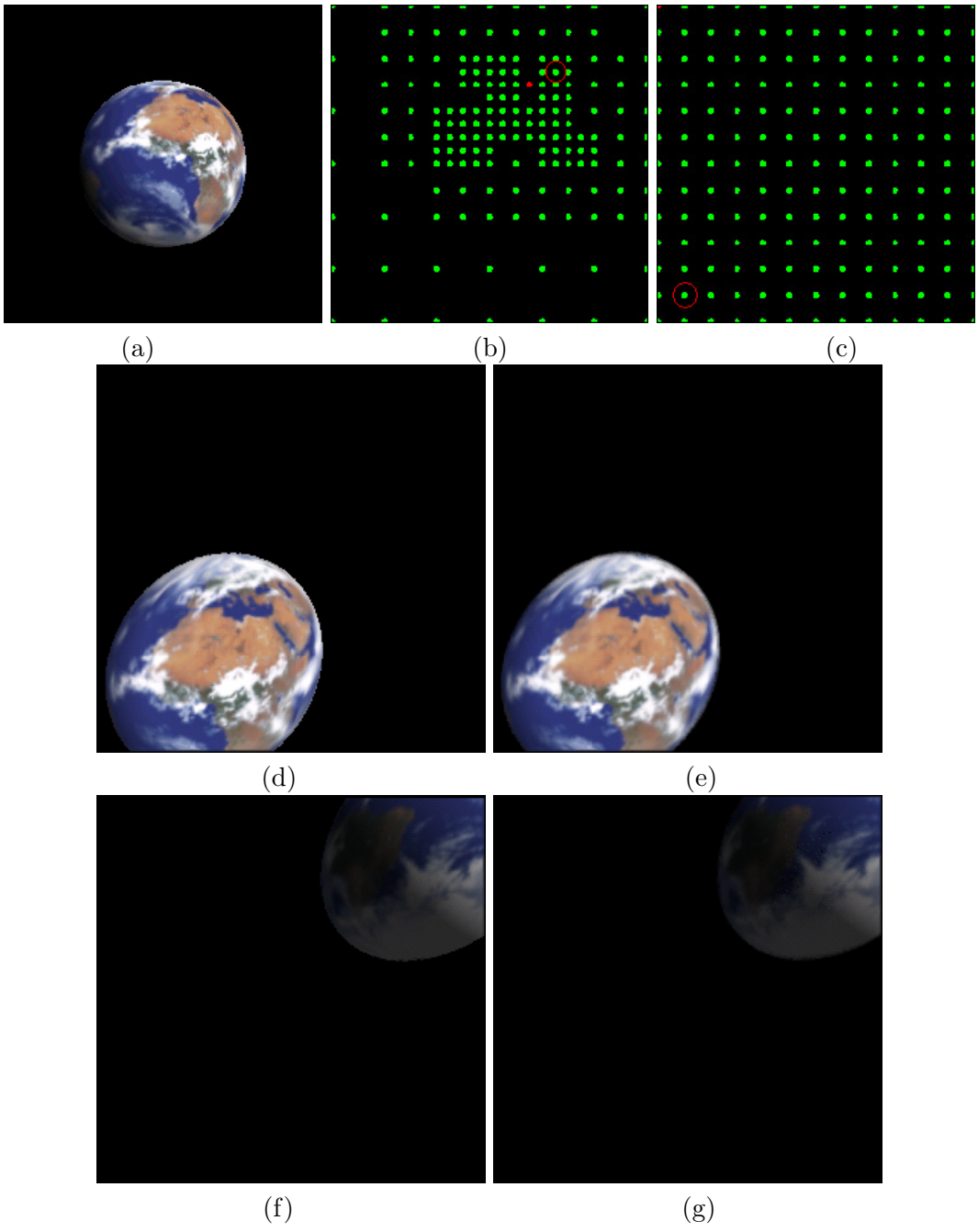


Figure 7.4: Active incremental sampling of a light field *Earth*. (a) A snapshot of the *Earth* scene; (b) camera map using active incremental sampling; (c) camera map using uniform sampling; (d) an image captured by active incremental sampling, but not by uniform sampling; (e) rendered image of the same image in (d) from uniform sampling; (f) an image captured by uniform sampling but not by active incremental sampling; (g) rendered image of the same image in (f) from active incremental sampling.

Table 7.1: Performance comparison between uniform sampling and active incremental sampling on light field scene *Earth*.

	Uniform sampling	Active incremental sampling
Avg. PSNR of 30 worst center views	32.8dB	33.2dB
PSNR Var. of 1000 rendered images	3.14dB	2.61dB

(f) because the quality degradation from (d) to (e) is more obvious than that from (f) to (g). Therefore active incremental sampling has made the right decision.

To measure the improvement of active incremental sampling over the uniformly sampled light field, we employ two objective measures. The first is the worst-case quality. From the cliques formed by both approaches, we render the virtual views at their centers. As the center views are the farthest from the sampled images, most likely they will have the worst quality. Our first measure is the average peak signal-to-noise ratio (PSNR) of the worst 30 center views. Notice that we are able to measure the PSNRs because we are using synthetic scenes and we have the real rendered images as our ground-truth. The second measure is the PSNR variance of rendered images. We randomly render 1000 images on the camera plane and measure the variance of the PSNRs. The results are shown in Table 7.1. It can be observed that active IBR has a better worst-case quality and a smaller variance, which is what we expected by performing active sampling.

We may also perform active incremental sampling with triangle cliques, as was shown in Figure 6.5(ii). The independent work by Schirmacher et al. [121] was such an example despite some difference in detailed implementation. The same technique may also be applied for spherical light field [47] if the cliques are defined on the spherical surface.

7.2.2 Active incremental sampling for the concentric mosaics

In concentric mosaics, the cameras are arranged on a circle. Here we present the active incremental sampling for a inward-looking concentric mosaics[1], as shown in Figure 7.5. The sample pair subdivision structure of Figure 6.5(iii) is adopted, as shown in Figure 7.5(b).

[1]Sometimes this is also referred as *orbital motion* in computer vision.

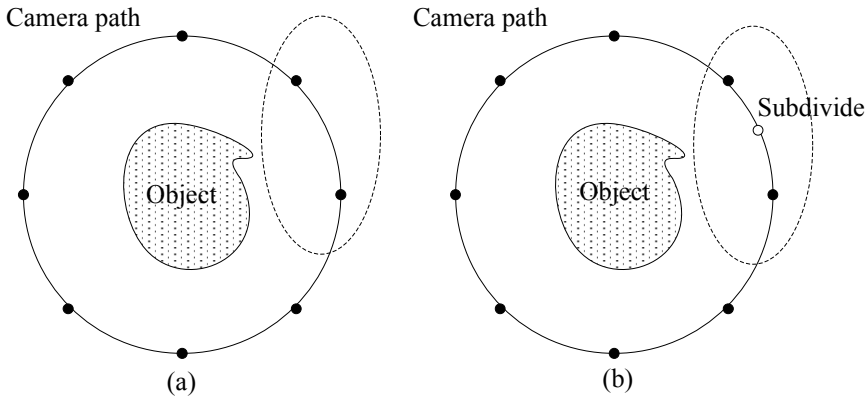


Figure 7.5: Active incremental sampling for the concentric mosaics setup.

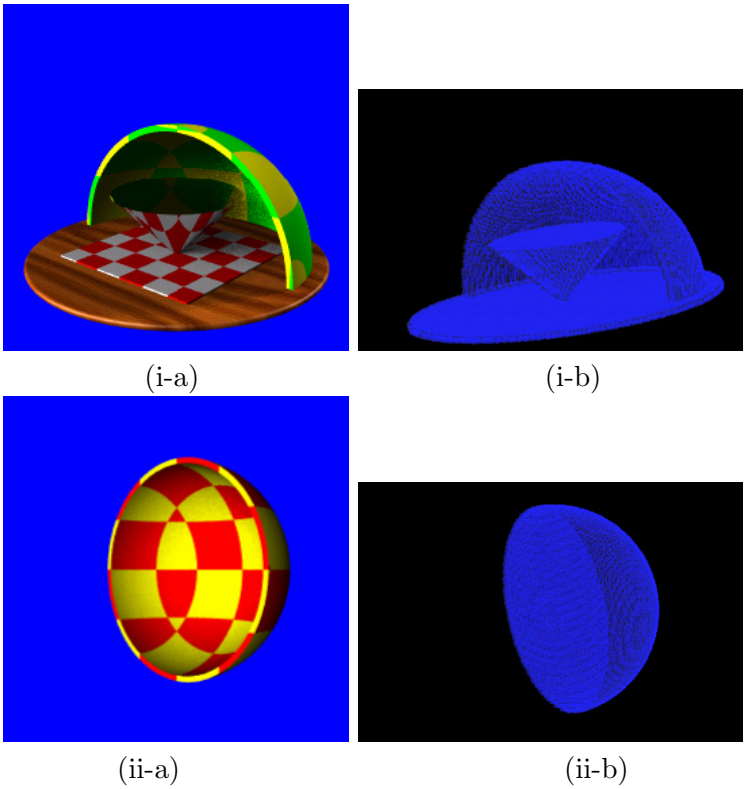


Figure 7.6: Concentric mosaics scenes: (i) Reflective cone; (ii) Hemisphere bowl; (a) scene snapshot; (b) used scene geometry.

The above algorithm is applied on two synthetic scenes: the *Reflective cone* (RC) and the *Hemisphere bowl* (HB), whose snapshots are shown in Figure 7.6. In the RC scene, we assume that the geometry model of the scene is known, as in Figure 7.6(i-b). A reflective cone is positioned at the center of the scene, which requires more samples than usual objects as it is highly non-Lambertian (Chapter 4). The HB scene is a hollow hemisphere similar to a bowl. We assume that we do not have its true geometric model, but only its visual hull [61], as in Figure 7.6(ii-b). This example is to show that more samples should be placed at the side where the geometry is wrong, as will be done by active sampling.

We first apply both active incremental sampling and uniform sampling on the scene RC. The number of overall images is constrained to be 96. In active incremental sampling, 24 images are used as the initial set that are uniformly sampled. The experimental results are shown in Figure 7.7. Figure 7.7(a) and (b) show the camera distribution of active incremental sampling and uniform sampling. Each small yellow sphere represents the camera position of a certain captured image. The front side towards the reader corresponds to view positions that can see the reflective cone. It is obvious that active incremental sampling puts a lot more effort on this side. Figure 7.7(c) and (d) are the variance of all the corresponding light rays between neighboring images (σ in Equation 7.3). The horizontal axis is the captured camera's view direction; the vertical axis is the variance. Figure 7.7(c) corresponds to active incremental sampling; (d) corresponds to uniform sampling. Despite the fact that the subdivision of the angles is non-uniform in Figure 7.7(c), active incremental sampling has more uniform variance, which can be translated to more uniform rendering image quality. Figure 7.7(e) and (f) show the rendering results of a certain view position where active incremental sampling to sample a lot more than uniform sampling. Notice the double edge in (f) does not show up in (e). Figure 7.7(g) and (h) show the rendering results of another view position where active incremental sampling captured much less images than uniform. No noticeable difference can be observed from these two images.

The scene HB is also captured with 96 images. Active incremental sampling uses 24 images as initialization. The results are shown in Figure 7.8. Active incremental sampling in this example decided to capture more images on the side where the geometry information

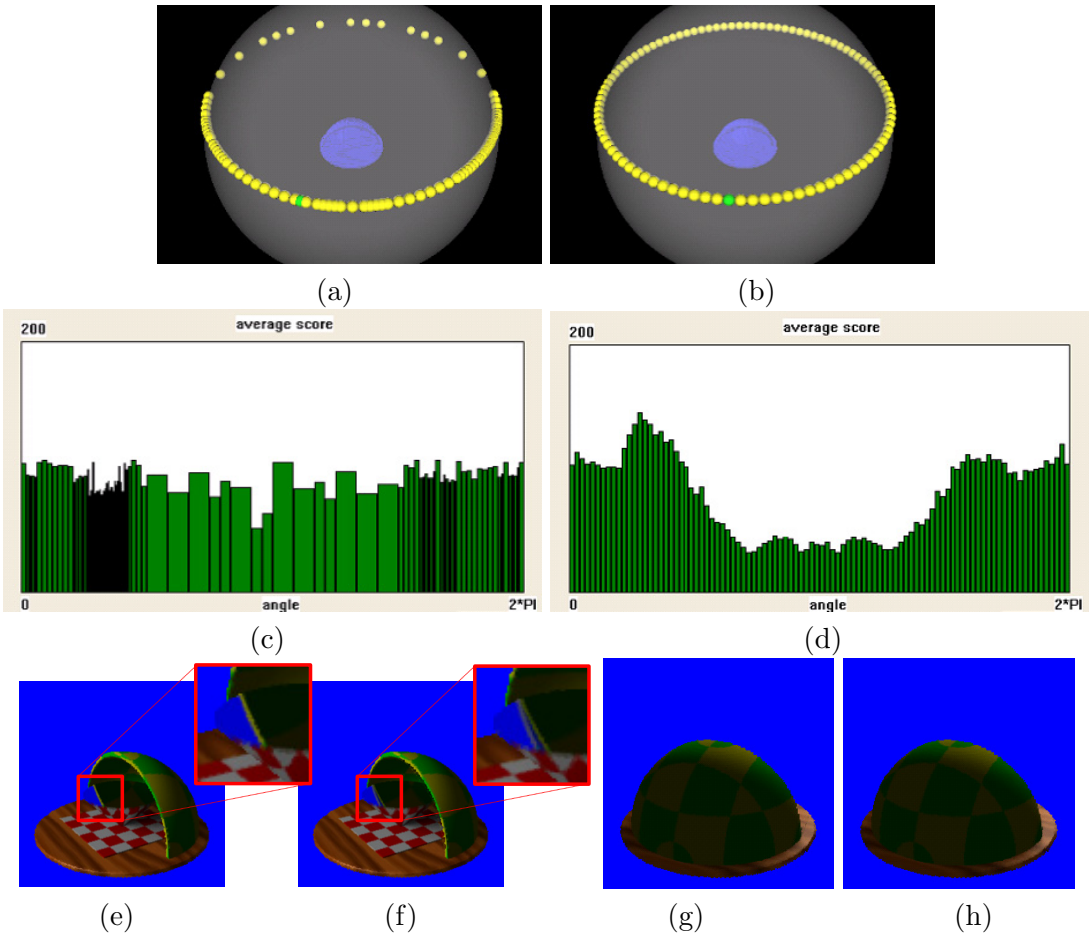


Figure 7.7: Active incremental sampling results on concentric mosaics scene RC. (a) Camera distribution for active incremental sampling; (b) camera distribution for uniform sampling; (c) the local inconsistency scores (variance) after active incremental sampling; (d) the local inconsistency scores after uniform sampling; (e)(g) rendering result for active incremental sampling. (f)(h) Rendering result for uniform sampling.

was wrong. Notice the final variance distribution of uniform sampling in Figure 7.8(d) vary much more than that of (c). From Figure 7.8(e) and (f), we see the ghosting effect in the uniform sampling rendered image is greatly reduced in active incremental sampling. On the other hand, less samples on the other side does not cause any quality degradation, as shown in Figure 7.8(g) and (h).

To give more insights on why the sample images should be nonuniformly arranged for the above two scenes, we show some of their EPIs in Figure 7.9. Figure 7.9 (a) and (b)

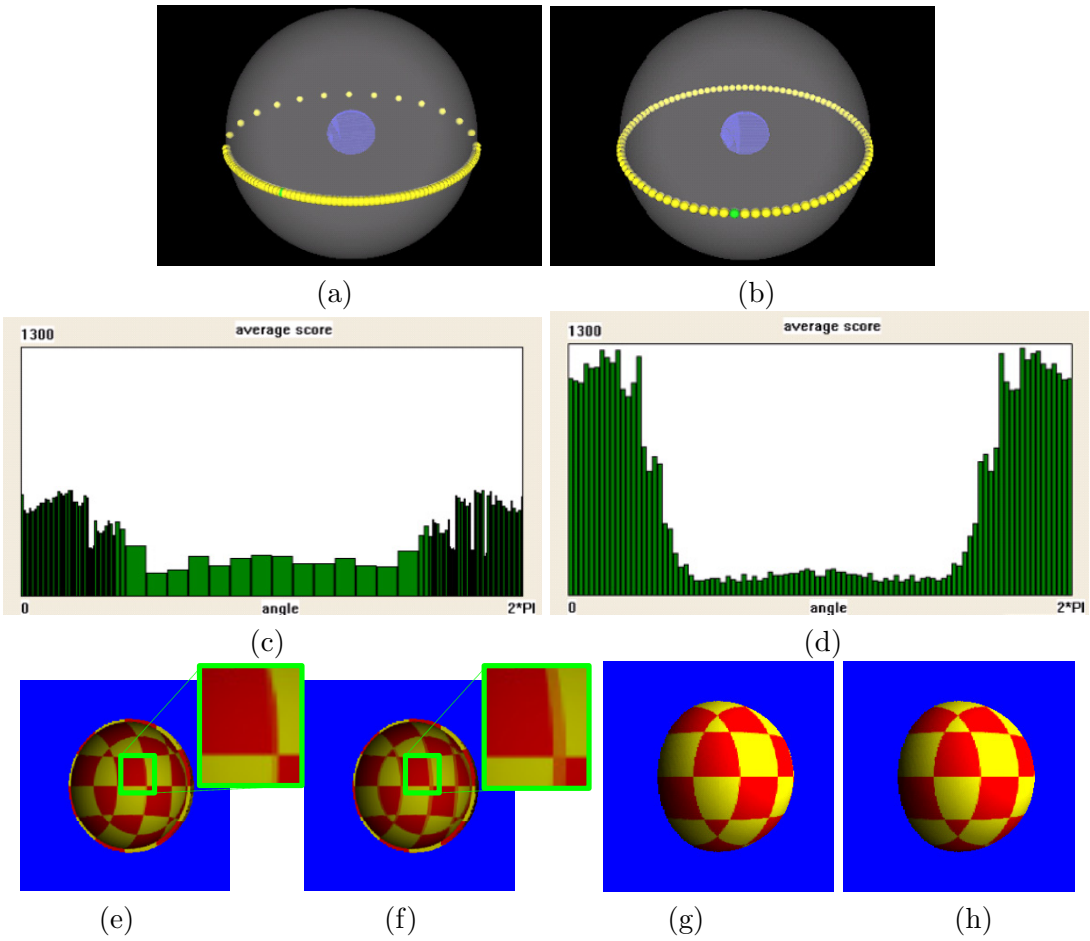


Figure 7.8: Active incremental sampling results on concentric mosaics scene HB. (a) Camera distribution for active incremental sampling; (b) camera distribution for uniform sampling; (c) the local inconsistency scores (variance) after active incremental sampling; (d) the local inconsistency scores after uniform sampling; (e)(g) rendering result for active incremental sampling. (f)(h) Rendering result for uniform sampling.

show the RC scene and one of its EPIs that cross the reflective cone. Figure 7.9 (c) and (d) are almost the same scene but the cone is non-reflective. As we are performing geometry-assisted rendering, the edge directions in (d) can tell what are the interpolation directions during the rendering. However, when the cone is reflective, the edge directions in (b) is very different from (d), which results in a bad reconstruction. In the back side of the scene, however, the edge directions are the same, so not much samples are needed. Figure 7.9 (e) and (f) are about the HB scene. Figure 7.9 (g) and (h) are a hemisphere scene,

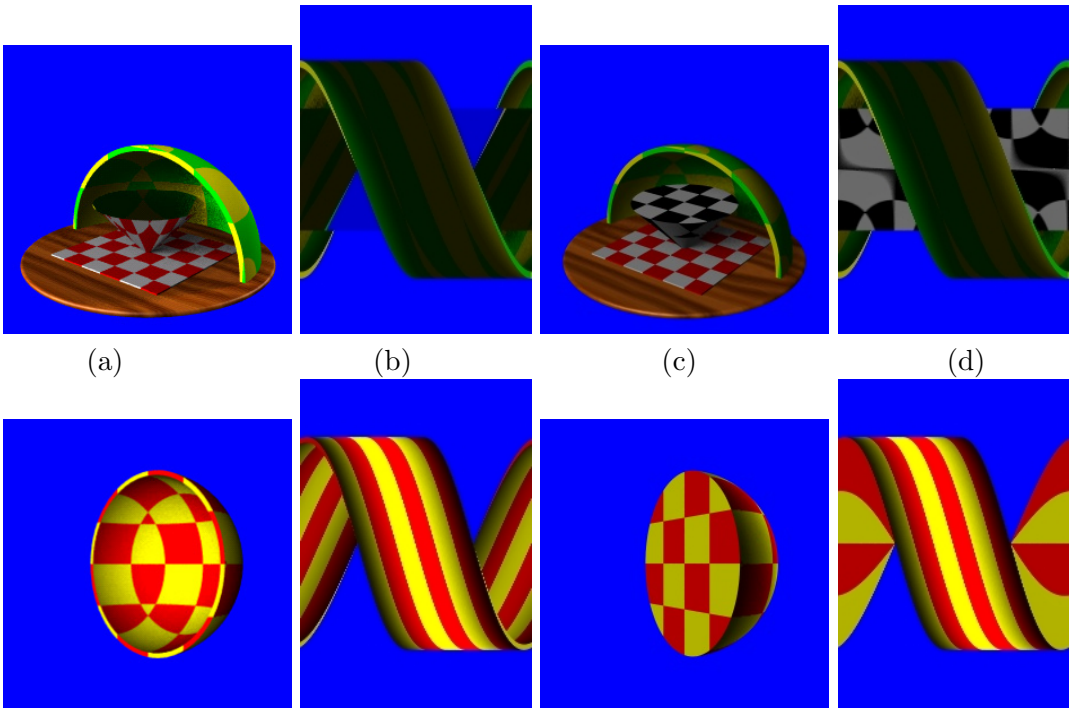


Figure 7.9: Comparison of EPIs for different scenes. (a) The RC scene; (b) EPI of (a) (row 140), which includes the cone; (c) same scene as (a) but the cone is not reflective; (d) EPI of (c) (row 140); (e) the HB scene; (f) EPI of (a) (row 130, which is about the center row); (g) a hemisphere scene; (h) EPI of (g) (row 130).

thus the geometry in Figure 7.6 (ii-b) is a perfect match with the scene content. The edge directions of Figure 7.9 (h) thus imply the interpolation directions when using a hemisphere as the rendering geometry. Obviously the edge directions in Figure 7.9 (f) is very different from (h), particularly at the left-most and right-most of the EPI, which suggests a higher sampling rate there.

7.3 IBR Active Rearranged Sampling

Active rearranged sampling is applicable when the overall number of images one can keep is limited. We again apply it to both the light field and the concentric mosaics. Recall in

Section 6.4.2 we mentioned that there are two algorithms for active rearranged sampling—local rearrangement and global rearrangement. As local rearrangement is generally slow, we focus on global rearrangement in this section. A local rearrangement algorithm will be presented in Section 8 for our mobile camera array.

7.3.1 Active rearranged sampling for the light field

Similar to Section 7.2.1, in this study we assume the capturing cameras are arranged on the camera plane, and they can freely move around. We assume that there are altogether N cameras on the camera plane. The goal of active rearranged sampling is thus to arrange these N cameras such that the reconstructed virtual views have higher worst-case quality and smaller quality jittering, as was done through active incremental sampling in Section 7.2.1. This problem is not difficult to solve using the general strategy presented in Section 6.4.2.

To show the full capability of freeform sampling and active sampling, here we make some further assumptions and demonstrate the case where the reconstruction set becomes a subset of the whole plenoptic function. We assume during the camera rearrangement there are P viewers who are watching the scene. The goal changes to arranging these N cameras such that the P views can be rendered at their best worst-case quality. Notice that with this assumption, the reconstruction set effectively changes to the light rays to be synthesized in the P virtual views.

Figure 7.10 shows the setup of our example with a single virtual view ($P = 1$). The capturing cameras are on the camera plane. The virtual view is at O. During the rendering, we assume a certain rendering geometry, and interpolate each light ray (such as OP in Figure 7.10) through its nearby captured views, as described in Section 7.1. Following the discussion around Equation 7.4, we know the local consistency score of each light ray can easily be calculated. In most cases these consistency scores will vary a lot if the initial camera distribution is uniform or random. We next describe an algorithm to perform global rearrangement of these capturing cameras.

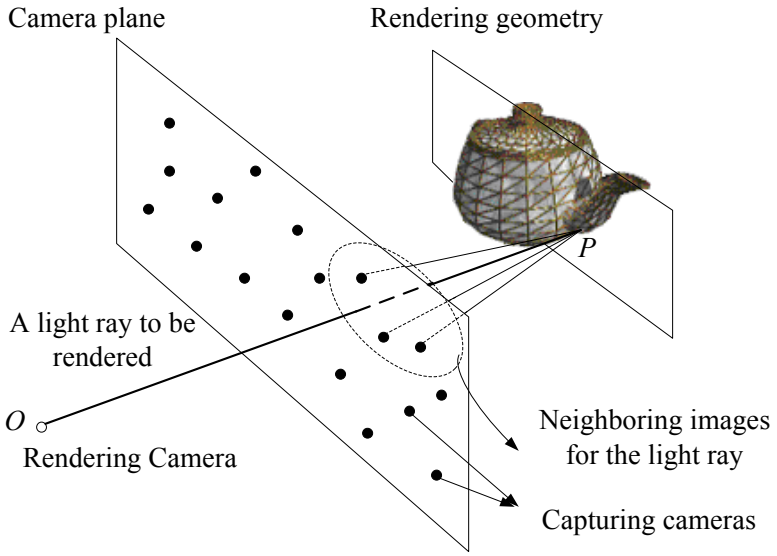


Figure 7.10: The setup of active rearranged sampling for the light field.

At time instance k, let the cameras' positions on the camera plane be $\mathbf{c}_j^k, j = 1, 2, \cdots, N$. For the P views being rendered, we may split them into totally L light rays. Denote them as $l_i, i = 1, 2, \cdots, L$. The intersection of the light rays and the camera plane are denoted as $\mathbf{x}_i, i = 1, 2, \cdots, L$. Notice the distance from a camera position $\mathbf{c}_j$ to a light ray intersection $\mathbf{x}_i$ is an indication of how close the camera is to the light ray. For each light ray, a local consistency score can be calculated as Equation 7.4, denoted as $\mathcal{C}_i^k, i = 1, 2, \cdots, L$. The problem is, given $\mathcal{C}_i^k$ and $\mathbf{c}_j^k$, if $\mathbf{x}_i$ does not change, how we position the cameras at the next time instance $k+1$, i.e., how to obtain $\mathbf{c}_j^{k+1}, j = 1, 2, \cdots, N$, so that the virtual views' worst case quality can be improved.

To solve the above problem, we introduce auxiliary weights on the light rays $w_i^k, i = 1, 2, \cdots, L$, which are normalized as $\sum_{i=1}^{L} w_i^k = 1$. These weights represent how close the corresponding light rays require their closest cameras to be. As a result, we determine the camera positions at the next time instance $k+1$ as[2]:

$$\left\{\mathbf{c}_j^{k+1}, j = 1, \cdots, N\right\} = \underset{\{\mathbf{c}_j, j=1,\cdots,N\}}{\arg\min} \sum_{i=1}^{L} w_i^k \min_{j=1,\cdots,N} \|\mathbf{x}_i - \mathbf{c}_j\| \tag{7.5}$$

which is in the form of a weighted vector quantization (VQ) problem. Equation 7.5 tries

[2]This formulation was developed step-by-step in [162].

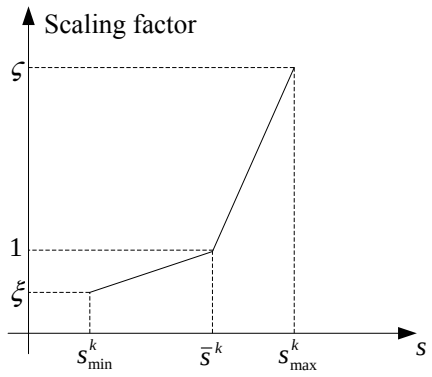


Figure 7.11: Scaling factor for updating the auxiliary weights.

to find the set of camera positions at time instance $k+1$, such that the sum of all the (weighted) distances from the light ray intersections to their nearest capturing cameras are minimized. The problem of updating the camera locations becomes the updating of these auxiliary weights. Notice in the above minimization problem, the current camera positions $\mathbf{c}_j^k, j = 1, \cdots, N$ is not explicitly involved. These positions and the captured images at instance k are only used when calculating the color consistency of the light rays and updating the auxiliary weights $w_i^k, i = 1, 2, \cdots, L$. If a certain light ray has a bad reconstruction quality at instance k, its weight should be increased, so that after the weighted VQ the cameras will move closer to that light ray, which will subsequently reduce the distances from the light ray to the nearby cameras and increase the rendering quality of that light ray; otherwise, its weight should be decreased. We thus associate the weight updating with the local consistency scores. Let:

$$s_i^k = -\log \mathcal{C}_i^k = \log {\sigma'}_i^k, \tag{7.6}$$

where ${\sigma'}_i^k$ was defined in Equation 7.4. Let $s^k_{\min}$ and $s^k_{\max}$ be the minimum and maximum value of $s_i^k, i = 1, \cdots, L$. $\overline{s}^k$ be the average value of s_i^k. The weight w_i^k at time instance k is updated from those w_i^{k-1} at time instance $k-1$ as:

$$w_i^k = \begin{cases} \left[1 + (\xi - 1)\frac{\overline{s}^k - s_i^k}{\overline{s}^k - s^k_{\min}}\right] w_i^{k-1}, & s_i^k \le \overline{s}^k; \\ \left[1 + (\zeta - 1)\frac{s_i^k - \overline{s}^k}{s^k_{\max} - \overline{s}^k}\right] w_i^{k-1}, & s_i^k > \overline{s}^k. \end{cases} \tag{7.7}$$

where ξ and ζ are the minimum and maximum weight scaling factor. They are set as 0.5 and

4 respectively in the current implementation. The relationship between the scaling factor and s_i^k is plotted in Figure 7.11. It basically says that if the variance of the projections to the neighboring images for a light ray is greater than the average (thus the local color consistency is bad), its weight will be increased. In fact any scaling factor that satisfies this condition can be employed here. During the weighted VQ, the camera positions will then move closer to that light ray. Otherwise, the camera positions will move away. Notice that after the weight update with Equation 7.7, one should normalize the new weights such that $\sum_{i=1}^{L} w_i^k = 1$. The initial values of the weights $w_i^0, i = 1, 2, \cdots, L$ can be rather arbitrary, as it will not affect too much after the second round of updating. For example, we may simply set:

$$w_i^0 = \frac{1}{L}, i = 1, 2, \cdots, L. \tag{7.8}$$

After the weights have been updated, we solve the weighted VQ problem in Equation 7.5 using a slightly modified LBG algorithm [73], which is based on the following two criteria:

1. Nearest neighbor condition:

$$\mathbf{x}_i \in \mathcal{R}_j, \text{ if } \|\mathbf{x}_i - \mathbf{c}_j\| \leq \|\mathbf{x}_i - \mathbf{c}_{j'}\|, \forall j' = 1, \cdots, N \tag{7.9}$$

where $\mathcal{R}_j$ is the neighborhood region of centroid $\mathbf{c}_j$.

2. Centroid condition:

$$\mathbf{c}_j = \frac{\sum_{\mathbf{x}_i \in \mathcal{R}_j} w_i \mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathcal{R}_j} w_i}, j = 1, \cdots, N \tag{7.10}$$

The above weighted VQ algorithm iteratively applies Equation 7.9 and Equation 7.10 to find the solution of Equation 7.5. The initial centroid $\mathbf{c}_j$ of the above algorithm can either be the current camera positions or a random set. In fact to avoid local minimum solution, multiple initial positions can be tried and the best result can be selected. Notice that during the weighted VQ, the capturing cameras do not need to move until we obtain the converged solution of the centroids.

We verify the effectiveness of the proposed active rearranged sampling algorithm with a static synthetic scene, as shown in Figure 7.12. The scene is named *Teapot* and is captured by 64 cameras. A single virtual view is used whose position is off the camera plane. In

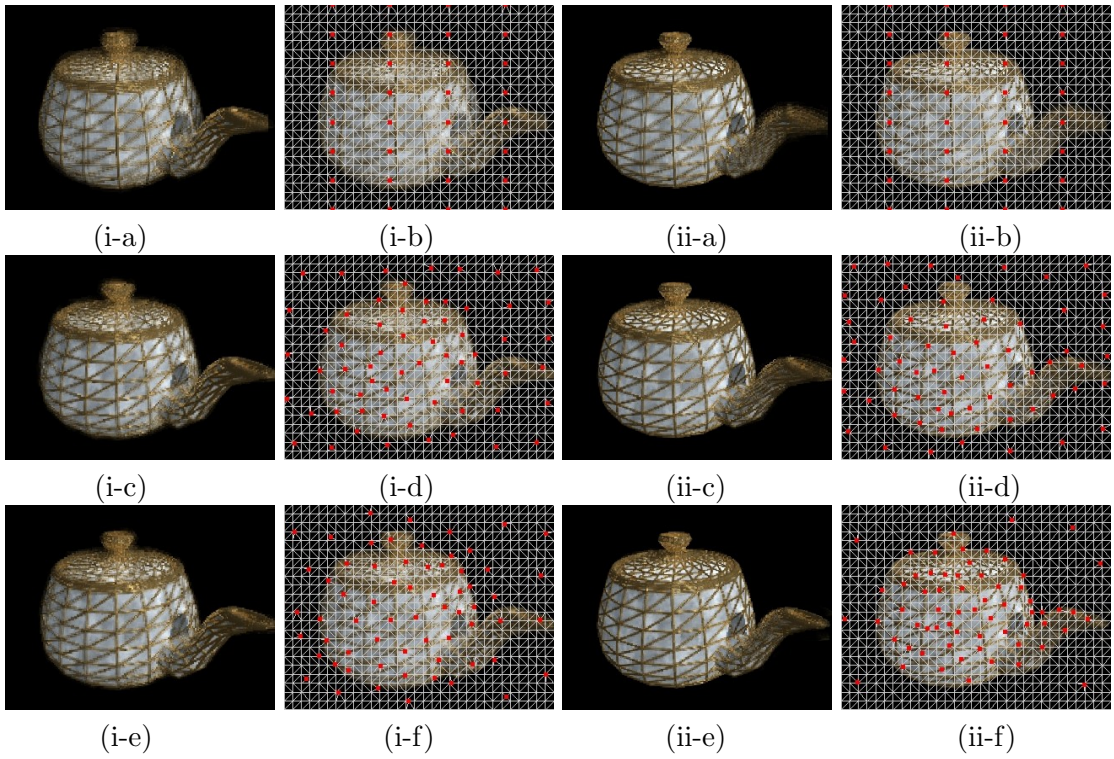


Figure 7.12: Active rearranged sampling results of a light field *Teapot*. (i) View rendered with the depth plane at its mouth; (ii) view rendered with the depth plane around the lid. (a)(b) Uniform sampling and its camera distribution. (c)(d) After one iteration of weighted VQ and the corresponding camera distribution. (e)(f) After 3 iterations.

Figure 7.12(i-a), we use a constant depth plane at the teapot's mouth as the rendering geometry. The body and lid of the teapot are thus blurred due to the inaccurate geometry. Figure 7.12(i-b) shows the projections of the cameras' positions to the rendered view (red dots). The white triangles are used during the rendering for texture mapping, however the corner of the triangles have the same depth since we do not have the scene geometry. Figure 7.12(i-c) and (i-d) are the active rearranged sampling results after one iteration of weighted VQ. The rendering quality improvement is very obvious. Notice that the camera positions are moving towards the body and lid of the teapot, where the rendering quality was bad. Figure 7.12(i-e) and (i-f) are the results after 3 iterations. More samples are around the lid because that is the place that has the wrongest geometry. Figure 7.12(ii-a) to (ii-f) is another set of results when the rendering depth is around the lid. Notice that the active

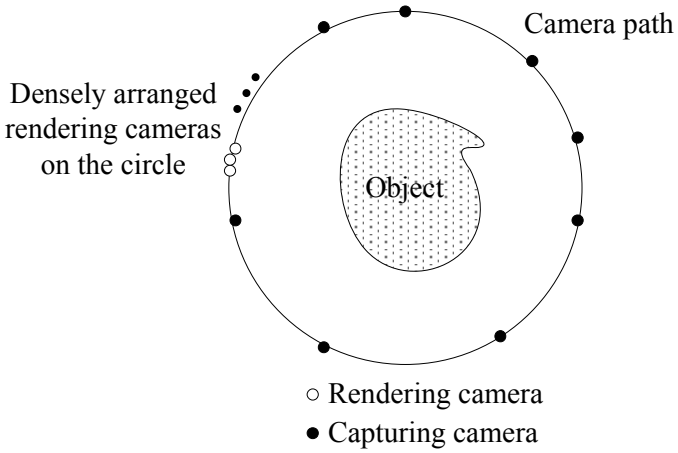


Figure 7.13: The setup of active rearranged sampling for the concentric mosaics.

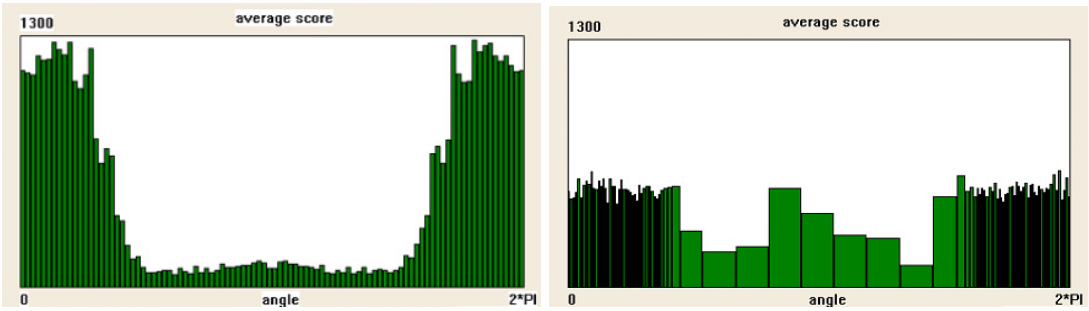
rearranged sampling scheme automatically moves the cameras to the regions of body and mouth.

7.3.2 Active rearranged sampling for the concentric mosaics

Active rearranged sampling can be done for the concentric mosaics setup in a way very similar to the algorithm we presented in Section 7.3.1. As shown in Figure 7.13, we assume N capturing cameras are placed on a circle. We also assume there are P virtual views to be rendered. These P views are uniformly distributed on the same circle. Here P can be very large, so that the P views can be considered as a good approximation of the whole region of support of the plenoptic function in concentric mosaics.

Active rearranged sampling for concentric mosaics is a one-dimensional problem. At time instance k, let the camera positions be $c_j^k, j = 1, 2, \cdots, N$, and the P virtual views be at $x_i, i = 1, 2, \cdots, P$, where c_j^k and x_i can both be angle in the unit of degree on the circle. For each virtual view, an accumulated local consistency score can be calculated, similar to that in Section 7.2.2. Denote them as $\mathcal{C}_i^k, i = 1, 2, \cdots, P$. The goal of active rearranged sampling in this case is to arrange the N capturing cameras such that the P virtual views have the best worse case quality.

This time we introduce a weight for each view, i.e., $w_i^k, i = 1, 2, \cdots, P$. The next



(a) (b)

Figure 7.14: Active rearranged sampling results of concentric mosaics scene *Hemisphere bowl*. (a) Uniform sampling; (b) Active rearranged sampling after two iterations of weight update.

positions of the cameras are obtained through:

$$c_j^{k+1} = \arg\min_{c_j} \sum_{i=1}^{P} w_i^k \min_{j=1,\cdots,N} |x_i - c_j| \tag{7.11}$$

which is again a weighted vector quantization problem. Let:

$$s_i^k = -\log \mathcal{C}_i^k, \tag{7.12}$$

and we may again use Equation 7.7 to update the weights.

We tested the algorithm on the scene *Hemisphere bowl* in Figure 7.6(ii). The number of capturing cameras is $N = 96$. Figure 7.14(b) shows the camera and error distribution after two iterations of weight update in active rearranged sampling. Compared with the uniform sampling results in Figure 7.14(a), the improvement is significant.

7.4 Summary

In this chapter, we demonstrated the applications of active sampling in image-based rendering. In either a light field setup or a concentric mosaics setup, active sampling outperforms the traditional uniform sampling approach in terms of the worst case rendering quality. The scenes we tested in this chapter are all synthetic scenes. In the next chapter, we present a mobile camera array for capturing real-world scenes, and we will show more examples of active sampling.

Chapter 8

The Self-Reconfigurable Camera Array

In this chapter, we presents a self-reconfigurable camera array system, which captures video sequences from an array of mobile cameras, renders novel views on the fly and reconfigures the camera positions to achieve better rendering quality. Our system is composed of 48 cameras mounted on mobile platforms. Compared with the previous work reviewed in Section 2.3.4, the contribution of our work is twofold. First, we propose an efficient algorithm that is capable of rendering high-quality novel views from the captured images in real-time. The algorithm involves the reconstruction of the scene geometry as an adaptive 2D mesh, which is yet another example of active incremental sampling. Second, we present an active rearranged sampling algorithm for the system, which moves the cameras in order to show better rendering results.

Compared with a single moving camera which follows the viewer's instruction and move around, a camera array has the clear benefit that its captured images can be used by thousands of viewers to view the scene simultaneously and from arbitrary view positions. In addition, the viewers are free to move their virtual viewpoint around without worrying about the mechanical speed of the single moving camera. Our work also added self-reconfigurability to the camera array, which can give a better rendering quality than a normal static camera array, as shown later in this chapter.

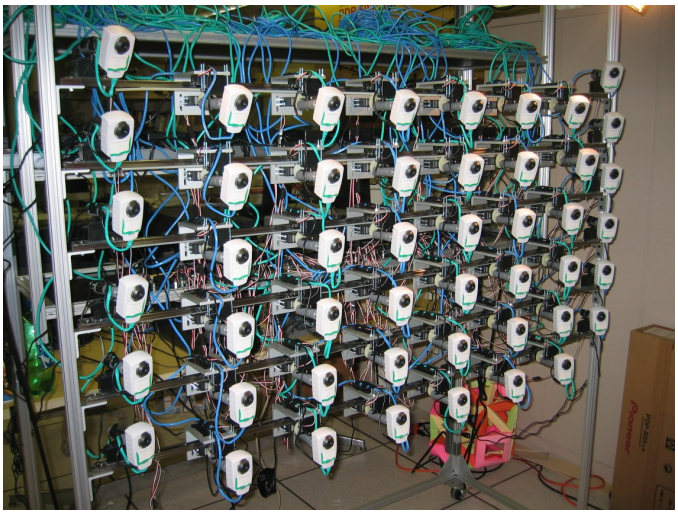


Figure 8.1: Our self-reconfigurable camera array system with 48 cameras.

8.1 System Overview

8.1.1 Hardware

As shown in Figure 8.1, our self-reconfigurable camera array system is composed of inexpensive off-the-shelf components. There are 48 (8×6) Axis 205 network cameras placed on 6 linear guides. The linear guides are 1600 mm in length, thus the average distance between cameras is about 200 mm. Vertically the cameras are 150 mm apart. They can capture up to 640×480 pixel^2 images at maximally 30 fps. The cameras have built-in HTTP servers, which respond to HTTP requests and send out motion JPEG sequences. The JPEG image quality is controllable. The cameras are connected to a central computer through 100Mbps Ethernet cables.

The cameras are mounted on a mobile platform, as shown in Figure 8.2. Each camera is attached to a pan servo, which is a standard servo capable of rotating for about 90 degrees. They are mounted on a platform, which is equipped with another sidestep servo. The sidestep servo is a hacked one, and can rotate continuously. A gear wheel is attached to the sidestep servo, which allows the platform to move horizontally with respect to the linear guide. The gear rack is added to avoid slippery during the motion. The two servos

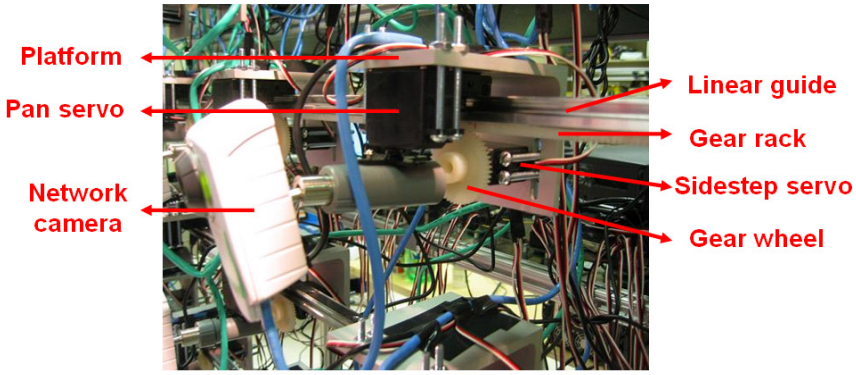


Figure 8.2: The mobile camera unit.

on each camera unit allow the camera to have two degrees of freedom—pan and sidestep. However, the 12 cameras at the leftmost and rightmost columns have fixed positions and can only pan.

The servos are controlled by the Mini SSC II servo controller [91]. Each controller is in charge of no more than 8 servos (either standard servos or hacked ones). Multiple controllers can be chained, thus up to 255 servos can be controlled simultaneously through a single serial connection to a computer. In the current system, we use altogether 11 Mini SSC II controllers to control 84 servos (48 pan servos, 36 sidestep servos).

Unlike any of the existing camera array systems mentioned in Section 2.3.4, our whole system uses only one single computer. The computer is an Intel Xeon 2.4 GHz dual processor machine with 1GB of memory and a 32 MB NVIDIA Quadro2 EX graphics card. As will be detailed in Section 8.3, our rendering algorithm is so efficient that the ROI identification, JPEG image decompression and camera lens distortion correction, which were usually performed with dedicated computers in previous systems, can all be conducted during the rendering process for a camera array at our scale. On the other hand, it is not difficult to modify our system and attribute ROI identification and image decoding to dedicated computers, as was done in the MIT distributed light field camera [155].

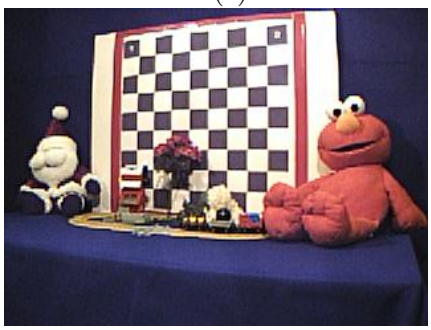
Figure 8.3(a) shows a set of images about a static scene captured by our camera array. The images are acquired at 320×240 pixel^2. The JPEG compression quality is set to be

(a)

(b)

(c)

(d)



(e)

Figure 8.3: Images captured by our camera array. (a) All the images; (b–e) sample images from selected cameras.

30 (0 being the best quality and 100 being the worst quality). Each compressed image is about 12-18 Kbytes. In a 100 Mbps Ethernet connection, 48 cameras can send such JPEG image sequences to the computer simultaneously at 15-20 fps, which is satisfactory. Several problems can be spotted from these images. First, the cameras have severe lens distortions, which have to be corrected during the rendering. Second, the colors of the captured images have large variations. The Axis 205 camera does not have flexible lighting control settings. We use the "fixed indoor" white balance and "automatic" exposure control in our system. Third, the disparity between cameras is large. As will be shown later, using constant depth assumption to render the scene will generate images with severe ghosting artifacts. Finally, the captured images are noisy (Figure 8.3 (b)–(e)). Such noises are from both the CCD sensors of the cameras and the JPEG image compression. These noises bring extra challenges to the scene geometry reconstruction.

The Axis 205 cameras cannot be easily synchronized. We make sure that the rendering process will always use the most recently arrived images at the computer for synthesis. Currently we ignore the synchronization problem during the geometry reconstruction and rendering, though it does cause problems while rendering fast moving objects, as might have been observed in the submitted companion video files.

8.1.2 Software Architecture

The system software runs as two processes, one for capturing and the other for rendering. The capturing process is responsible for sending requests to and receiving data from the cameras. The received images (in JPEG compressed format) are directly copied to some shared memory that both processes can access. The capturing process is often lightly loaded, consuming about 20% of one of the processors in the computer. When the cameras start to move, their external calibration parameters need to be calculated in real-time. Camera calibration is also performed by the capturing process. As will be described in the next section, calibration of the external parameters generally runs fast (150–180 fps).

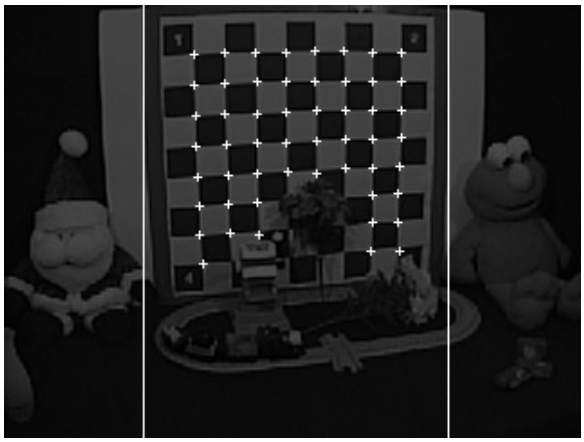


Figure 8.4: Locate the feature corners of the calibration pattern.

The rendering process runs on the other processor. It is responsible for ROI identification, JPEG decoding, lens distortion correction, scene geometry reconstruction and novel view synthesis. Details about the rendering process will be described in Section 8.3.

8.2 Camera Calibration

Since our cameras are designed to be self-reconfigurable, calibration must be performed in real-time. Fortunately, the internal parameters of the cameras do not change during their motion, and can be calibrated offline. We use a large planar calibration pattern for the calibration process (Figure 8.3). Bouguet's calibration toolbox [9] is used to obtain the internal camera parameters.

To calibrate the external parameters, we first need to extract the features on the checkerboard. We assume that the top two rows of feature points will never be blocked by the foreground objects. The checkerboard boundary is located by searching for the red strips of the board in the top region of the image. Once the left and right boundaries are identified (as shown in Figure 8.4), we locate the top two rows of features by using a simple 5×5

linear filter:

$$h_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{8.1}$$

If the output of the above filter has an absolute value larger than a threshold, it is considered as a candidate feature location. After non-maximum suppression of the candidates, we further use a 9×9 linear filter to confirm their validity:

$$h_2 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 \\ 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 \\ 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{8.2}$$

A feature is valid only if the result of filter h_2 (absolute value) is also above a threshold. The above algorithm assumes that the calibration pattern is almost frontal in all the captured views. The thresholds of the two filters are chosen empirically.

The feature positions are then refined to sub-pixel accuracy by finding the saddle points, as in [9]. The corners below the second row are then extracted row by row. At each row, we predict the feature locations based on its previous two rows of features. The accurate position of the features are then found through the same approach above. The results of such feature extraction is shown in Figure 8.4. Notice that if the corner detector cannot find a feature along a column for a certain row due to various reasons such as occlusions, it will stop finding features below that row in that column.

Finally, to obtain the 6 external parameters (3 for rotation and 3 for translation) of the cameras, we use the algorithm proposed by Zhang [165]. The Levenberg-Marquardt method implemented in MinPack [93] is adopted for the nonlinear optimization. The above calibration process runs very fast on our processor (150–180 fps at full speed). As long as there are not too many cameras moving around simultaneously, we can perform calibration

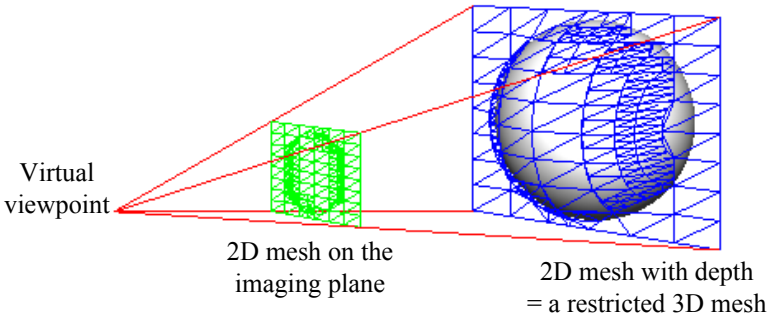


Figure 8.5: The multi-resolution 2D mesh with depth information on its vertices.

on-the-fly during the camera movement. In the current implementation, we constrain that at any instance at most one camera on each row can sidestep. After a camera has sidestepped, it will pan if necessary in order to keep the calibration board in the middle of the captured image.

8.3 Real-Time Rendering

8.3.1 Flow of the rendering algorithm

In this paper, we propose to reconstruct the geometry of the scene as a 2D multi-resolution mesh (MRM) with depths on its vertices, as shown in Figure 8.5. The 2D mesh is positioned on the imaging plane of the virtual view, thus the geometry is view-dependent (similar to that in [156, 130, 82]). The MRM solution significantly reduces the amount of computation spent on depth reconstruction, making it possible to be implemented efficiently in software.

The flow chart of the rendering algorithm is shown in Figure 8.6. A novel view is rendered when there is an idle callback or the user moves the viewpoint. We first construct an initial sparse and regular 2D mesh on the imaging plane of the virtual view, as is shown in Figure 8.7. For each vertex of the initial 2D mesh, we first look for a subset of images that will be used to interpolate its intensity during the rendering. Once such information has been collected, it is easy to identify the ROIs of the captured images and decode them when necessary. The depths of the vertices in the 2D mesh are then reconstructed. If a

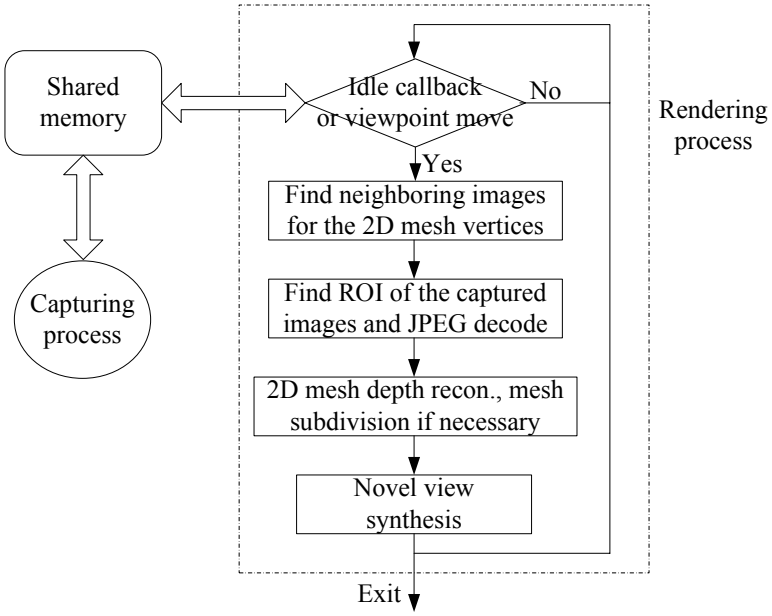


Figure 8.6: The flow chart of the rendering algorithm.

certain triangle in the mesh bears large depth variation, subdivision is performed to obtain more detailed depth information. After the depth reconstruction, the novel view can be synthesized through multi-texture blending, similar to that in the unstructured Lumigraph rendering (ULR) [11]. Lens distortion is corrected in the last stage, although we also compensate the distortion during the depth reconstruction stage. Details of the proposed algorithm will be presented next.

8.3.2 Finding close-by images for the mesh vertices

Each vertex on the 2D mesh corresponds to a light ray that starts from the virtual viewpoint and passes the vertex on the imaging plane. During the rendering, it will be interpolated from several light rays of some nearby captured images. We need to identify these nearby images for selective JPEG decoding and the scene geometry reconstruction. Unlike the ULR [11] and the MIT distributed light field camera [155] where the scene depth is known, we do not have such information at this stage, and cannot locate the neighboring images by

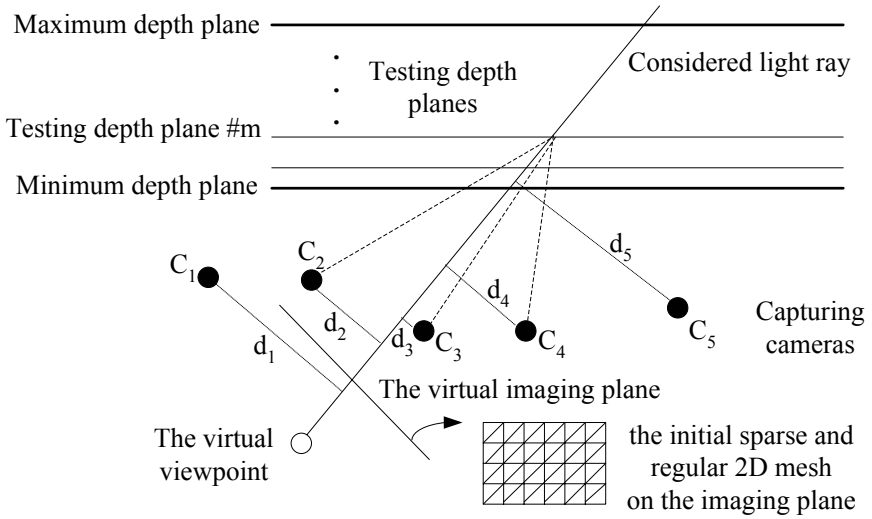


Figure 8.7: Locate the neighboring images for interpolation and depth reconstruction through plan sweeping.

angular differences of the light rays[1]. Instead, we adopted the distance from the cameras' center of projection to the considered light ray as the criterion. As shown in Figure 8.7, the capturing camera C_2, C_3 and C_4 have smaller distances, and will be selected as the 3 closest images. As our cameras are roughly arranged on a plane and point to roughly the same direction, when the scene is at a reasonably large depth, such distance is a good approximation of the angular difference used in the literature, yet it does not require the scene depth information.

8.3.3 ROI Identification and JPEG decoding

On the initial regular coarse 2D mesh, if a triangle has a vertex that select input image $\#n$ as one of the nearby cameras, the rendering of this triangle will need image $\#n$. In other words, once all the vertices have found their nearby images, given a certain input image $\#n$, we will be able to tell what are the triangles that need it during the rendering. Such information is used to identify the ROIs of the images that need to be decoded.

As shown in Figure 8.8, take image $\#n$ as an example. We back-project the triangles

[1] Although it is possible to find the neighboring images of the light rays for each hypothesis depth planes, we found such an approach too much time-consuming.

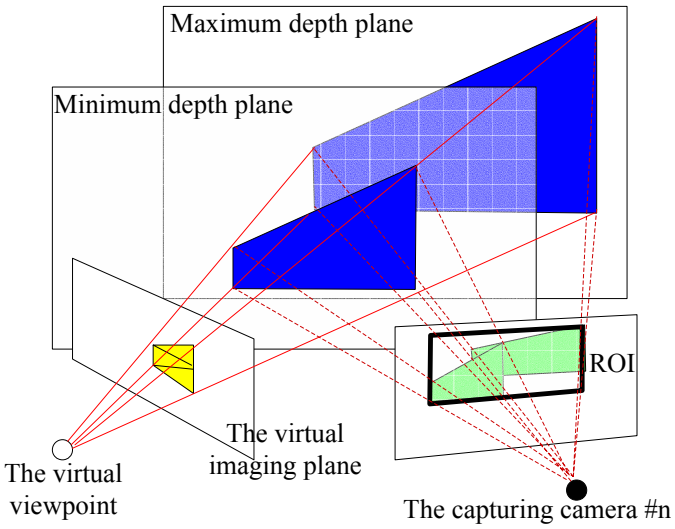


Figure 8.8: Determine the ROI of a certain input image.

(indicated in yellow) that need image $\#n$ for rendering from the virtual imaging plane to the minimum depth plane and the maximum depth plane, and then project the resultant regions to image $\#n$. The ROI of image $\#n$ is the smallest rectangular region that includes both the projected regions. Afterwards, the input images that do not have an empty ROI will be JPEG decoded (partially).

8.3.4 Scene depth reconstruction

We reconstruct the scene depth of the light rays passing through the vertices of the 2D mesh using a plane sweeping method. Such method has been used in a number of previous algorithms [22, 126, 155], although they all reconstruct a dense depth map of the scene. As illustrated in Figure 8.7, we divide the world space into multiple testing depth planes. For each light ray, we assume the scene is on a certain depth plane, and project the scene to the nearby input images obtained in Section 8.3.2. If the assumed depth is correct, we expect to see consistent colors among the projections. The plane sweeping method sweeps through all the testing depth planes, and obtain the scene depth as the one that gives the highest consistency.

Care must be taken in applying the above method. First, the location of the depth

planes should be equally spaced in the disparity space instead of in depth. Let $d_{\min}$ be the minimum scene depth, and $d_{\max}$ be the maximum scene depth, M be the number of depth planes used. The $\#m$ depth plane ($m = 0, 1, \cdots, M-1$) is located at:

$$d_m = \frac{1}{\frac{1}{d_{\max}} + \frac{m}{M-1}(\frac{1}{d_{\min}} - \frac{1}{d_{\max}})} \tag{8.3}$$

Equation 8.3 is a direct result from the sampling theory by Chai et al. [16]. In the same paper they also developed a sampling theory on the relationship between the number of depth planes and the number of captured images, which is helpful in selecting the number of depth planes. Second, when projecting the test depth planes to the neighboring images, lens distortions must be corrected. Third, to improve the robustness of the color consistency matching among the noisy input images, a patch on each nearby image is taken for comparison. The patch window size relies heavily on the noise level in the input images. In our current system, the input images are very noisy. We have to use an 18×18 patch window to accommodate the noise. The patch is first down-sampled horizontally and vertically by a factor of 2 to reduce some computational burden. Different patches in different input images are then compared to give an overall color consistency score. Fourth, as our cameras have large color variations, color consistency measures such as SSD do not perform very well. We applied mean-removed correlation coefficient for the color consistency verification (CCV). The normalized mean-removed inner products of all pairs of nearby input images are first obtained. Given a pair of nearby input images, e.g., $\#i$ and $\#j$, the correlation coefficient of the two patches is defined as:

$$r_{ij} = \frac{\sum_k (I_{ik} - \bar{I}_i)(I_{jk} - \bar{I}_j)}{\sqrt{\Big[\sum_k (I_{ik} - \bar{I}_i)^2\Big]\Big[\sum_k (I_{jk} - \bar{I}_j)^2\Big]}} \tag{8.4}$$

where I_{ik} and I_{jk} are the k^{th} pixel intensity in the patch $\#i$ and $\#j$, respectively. $\bar{I}_i$ and $\bar{I}_j$ are the mean of pixel intensities in the two patches. Equation 8.4 was widely used in traditional stereo matching algorithms [33]. The overall CCV score of the nearby input images is one minus the average correlation coefficient of all the image pairs. The depth plane resulting in the lowest CCV score will be selected as the scene depth. Since

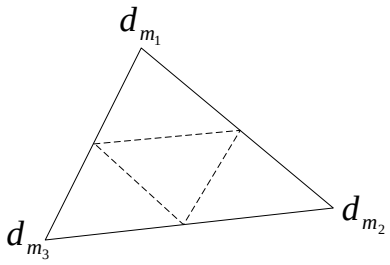


Figure 8.9: Subdivision of a mesh triangle.

the calculation of Equation 8.4 is very computationally expensive, we may perform early rejection if the intensities of the patches differ too much.

The depth recovery process starts with an initial regular and sparse 2D mesh, as was shown in Figure 8.7. The depths of its vertices are obtained with the above mentioned method. The sparse mesh with depth can serve well during the rendering if the scene does not have much depth changes. However, if the scene depth does change, a dense depth map is needed around those regions for satisfactory rendering results. We subdivide a triangle in the initial mesh if its three vertices have a large depth variation. As shown in Figure 8.9, let the depths of a triangle's three vertices be d_{m_1}, d_{m_2} and d_{m_3}, where m_1, m_2, m_3 are the indices of the depth planes. We subdivide this triangle if:

$$\max_{p,q\in\{1,2,3\},p\neq q} |m_p - m_q| > T \tag{8.5}$$

where T is a threshold set as 1 in the current implementation. During the subdivision, the midpoint of each edge of the triangle is selected as the new vertices, and the triangle is subdivided into 4 smaller ones. The depths of the new vertices are reconstructed under the constraints that they have to use neighboring images of the three original vertices, and their depth search range is limited to the minimum and maximum depth of the original vertices. Other than Equation 8.5, the subdivision may also stop if the subdivision level reaches a certain preset limit.

Real-time, adaptive conversion from dense depth map or height field to a mesh representation has been studied in literature [74, 12]. However, these algorithms assumed that a dense depth map or height field was available before hand. Therefore, they are similar

to the freeform sampling solutions discussed in Section 6.2. In contrast, our algorithm reconstructs a multi-resolution mesh model directly during the rendering, and belongs to the active incremental sampling. Szeliski and Shum [134] proposed a motion estimation algorithm based on a multi-resolution representation using quadtree splines, which could be considered as active incremental sampling too. Two dimensional splines are used to interpolate between control points; while we connect the mesh vertices with triangles. They used the residual flow for the subdivision of quadtree, which is applicable for two-view stereo. In comparison, our method is for multi-views. In addition, the geometry we reconstruct is view dependent, while the quadtree splines based method is for one of the stereo views.

The size of the triangles in the initial regular 2D mesh cannot be too large, since otherwise we may miss certain depth variations in the scene. A rule of thumb is that the size of the initial triangles/grids should match that of the object features in the scene. In the current system, the initial grid size is about 1/25 of the width of the input images. Triangle subdivision is limited to no more 2 levels.

8.3.5 Novel view synthesis

After the multi-resolution 2D mesh has been obtained, novel view synthesis is easy. Our rendering algorithm is very similar to the one in the ULR [11] except that our imaging plane has already been triangulated. The basic idea of ULR is to assign weights to nearby images for each vertex, and render the scene through multi-texture blending. In our implementation, the weights of the nearby images are assigned as Equation 7.2. During the multi-texture blending, only the ROIs of the input images will be used to update the texture memory when a novel view is rendered. As the input images of our system have severe lens distortions, we cannot use the 3D coordinates of the mesh vertices and the texture matrix in graphics hardware to specify the texture coordinates as in [11]. Instead, we perform the projection with lens distortion correction ourselves and provide 2D texture coordinates to the rendering pipeline. Fortunately, such projections to the nearby images have already been calculated during the depth reconstruction stage and can simply be reused.

8.3.6 Rendering results on synthetic scenes

We first report some rendering results of the proposed algorithm on some synthetic static scenes, *wineglass* and *skyvase*, as shown in Figure 8.10. Both scenes are synthesized with POV-Ray [111]. They have 64 input images, arranged regularly on a 2D plane as light field. Wineglass exhibits huge depth variation, transparency, shadowing and heavy occlusion, thus it is very challenging for geometry reconstruction. The skyvase scene features two semi-reflective walls which have their own textures and also reflect the vase in the foreground. Theoretically, no geometry reconstruction algorithm could work for this scene, because there are virtually two depths existing on the wall.

Figure 8.10 (a)–(c) are the rendering results when using a sparse mesh, a dense depth map and an adaptive mesh, all being view-dependent. The rendering position is at the center of four nearby capturing cameras (forming a rectangle) on the camera plane. As synthetic scenes has no noise, we use patch window 5×5 during the geometry reconstruction. Eight depth planes are used here for plane sweeping. (d)–(f) shows the geometric models we reconstructed. Both (b) and (c) have very good rendering quality, thanks to the view-dependent geometry reconstruction and the local color consistency verification. Most importantly, we reconstruct depth for 5437 vertices for the wineglass scene and 4507 vertices for the skyvase scene in the adaptive mesh approach, which are both much fewer than 76800 vertices if a per-pixel dense depth map is reconstructed.

8.3.7 Rendering results on real-world scenes

We have used our camera array system to capture a variety of scenes, both static and dynamic. The speed of rendering process is about 4-10 fps, depending on many factors such as the number of testing depth planes used for plane sweeping, the patch window size for CCV, the initial coarse regular 2D mesh grid size, the number of subdivision levels used during geometry reconstruction and the scene content. For the scenes we have tested, the above parameters can be set to fixed values. For instance, our default setting is 12 testing depth planes for depth sweeping, 18×18 patch window size, 1/25 of the width of the input

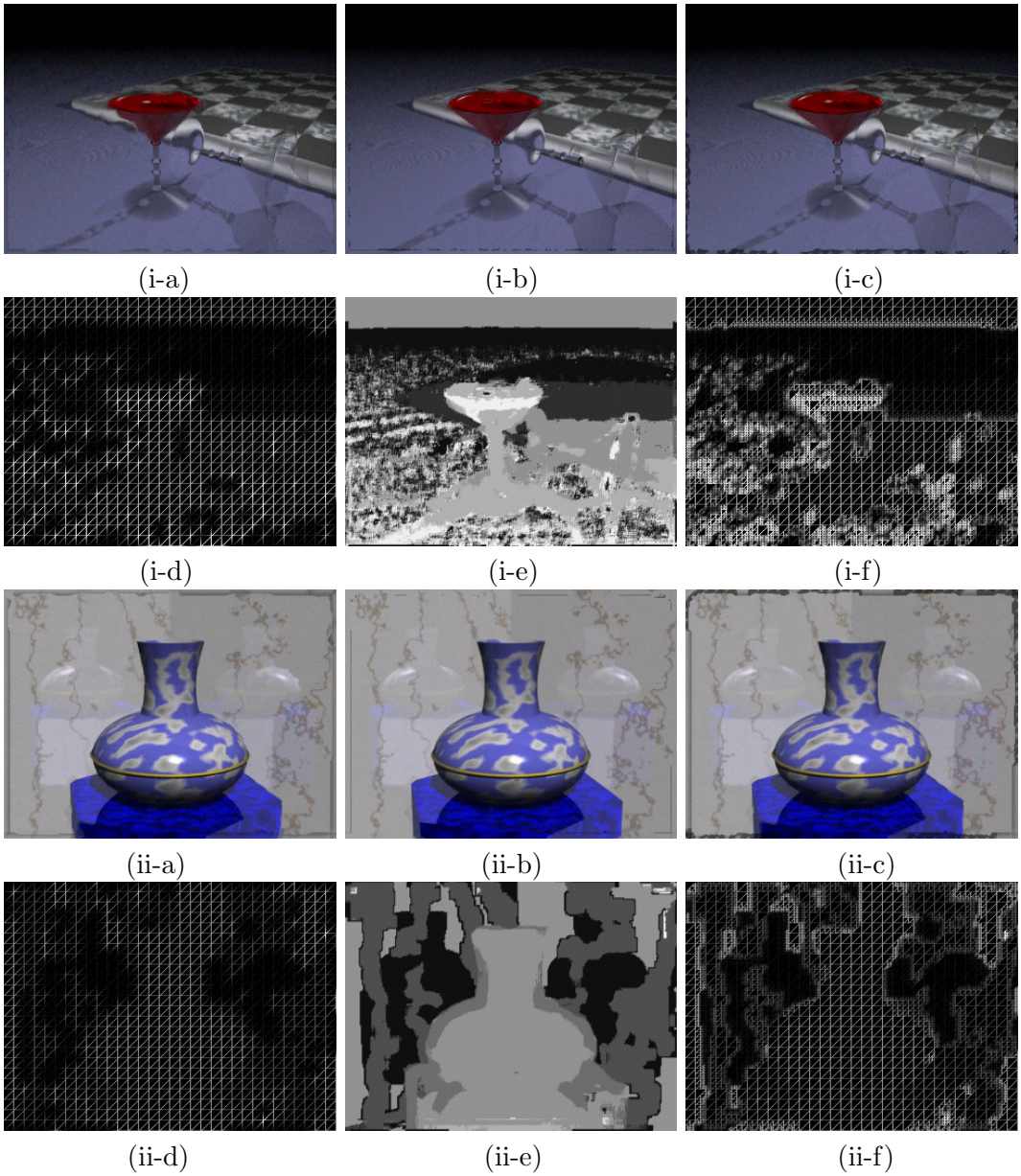


Figure 8.10: Synthetic scenes rendered with our proposed algorithm. (i) Scene *wineglass*; (ii) scene *skyvase*; (a)(d) use a sparse mesh to render; (b)(e) use a per-pixel depth map to render; (c)(f) use the proposed adaptive mesh to render.

images as initial grid size, and maximally 2 level of subdivision.

The time spent on each step of the rendering process under default setting is as follows. Finding neighboring images and ROI of them takes less than 10 ms. JPEG decoding takes 15-40 ms. Geometry reconstruction takes about 80-120 ms. New view synthesis takes about 20 ms.

The rendering results of some static scenes are shown in Figure 8.11. In these results the cameras are evenly spaced on the linear guide. The rendering positions are roughly on the camera plane but not too close to any of the capturing cameras. Figure 8.11(a)(b)(c) are results rendered with the constant depth assumption. The ghosting artifacts are very severe, because the spacing between our cameras is larger than most previous systems [155, 95]. Figure 8.11(d) is the result from the proposed algorithm. The improvement is significant. Figure 8.11(e) shows the reconstructed 2D mesh with depth information on its vertices. The grayscale intensity represents the depth – the brighter the intensity, the closer the vertex. Like many other geometry reconstruction algorithms, the geometry we obtained contains some errors. For example, in the background region of scene *toys*, the depth should be flat and far, but our results have many small "bumps". This is because part of the background region has no texture, which is prone to error for depth recovery. However, the rendered results are not affected by these errors because we use view-dependent geometry and the local color consistency always holds at the viewpoint.

Figure 8.12 gives the comparison of the rendering results using a dense depth map and our adaptive mesh, similar to that in Figure 8.10 but for real-world scenes. Again, using adaptive mesh produces rendering images at almost the same quality as using dense depth map, but with a much smaller computational cost.

8.3.8 Discussions

Our current system has certain hardware limitations. For example, the images captured by the cameras are at 320×240 pixel^2 and the image quality is not very high. This is mainly constrained by the throughput of the Ethernet cable. Upgrading the system to

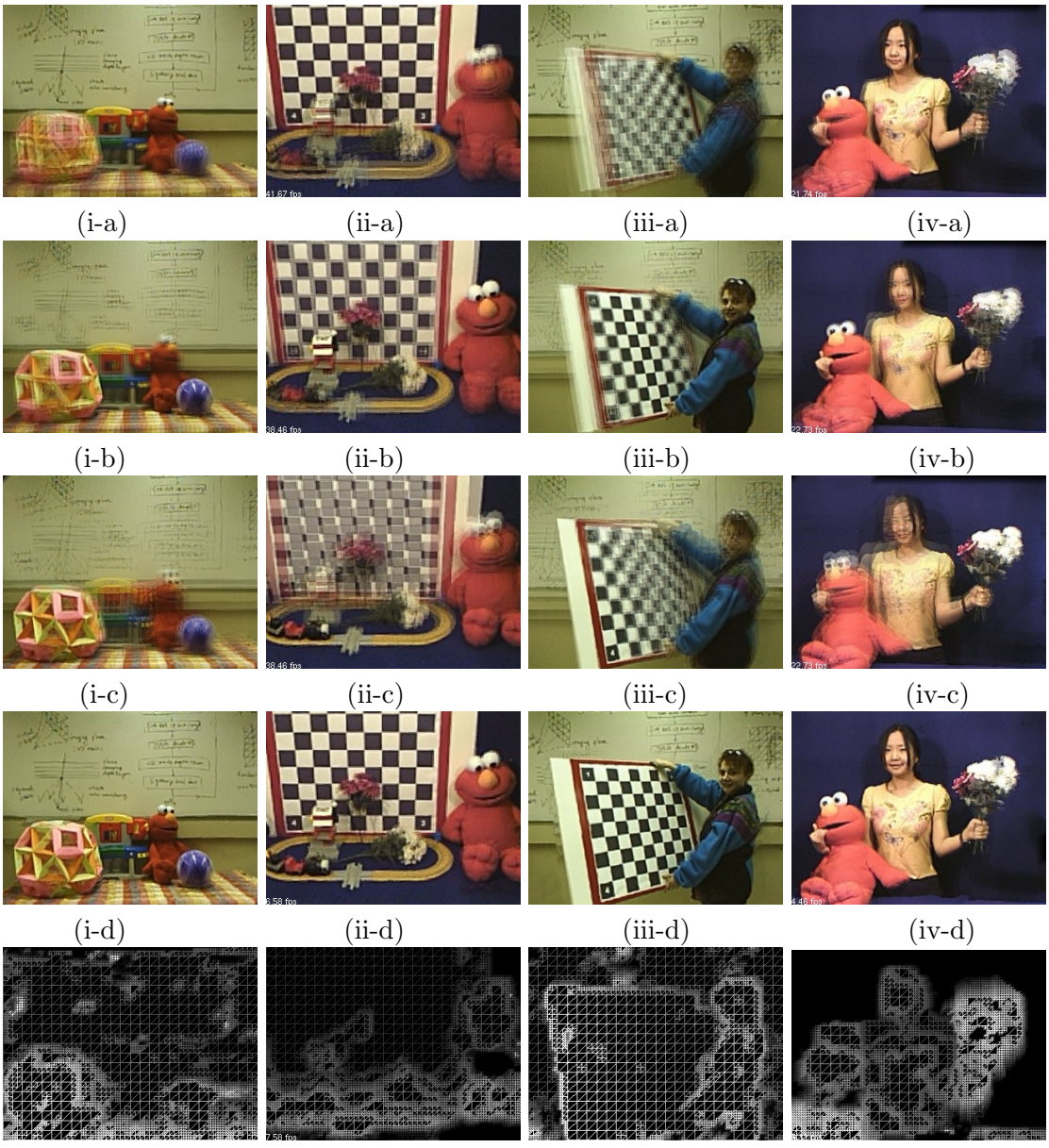
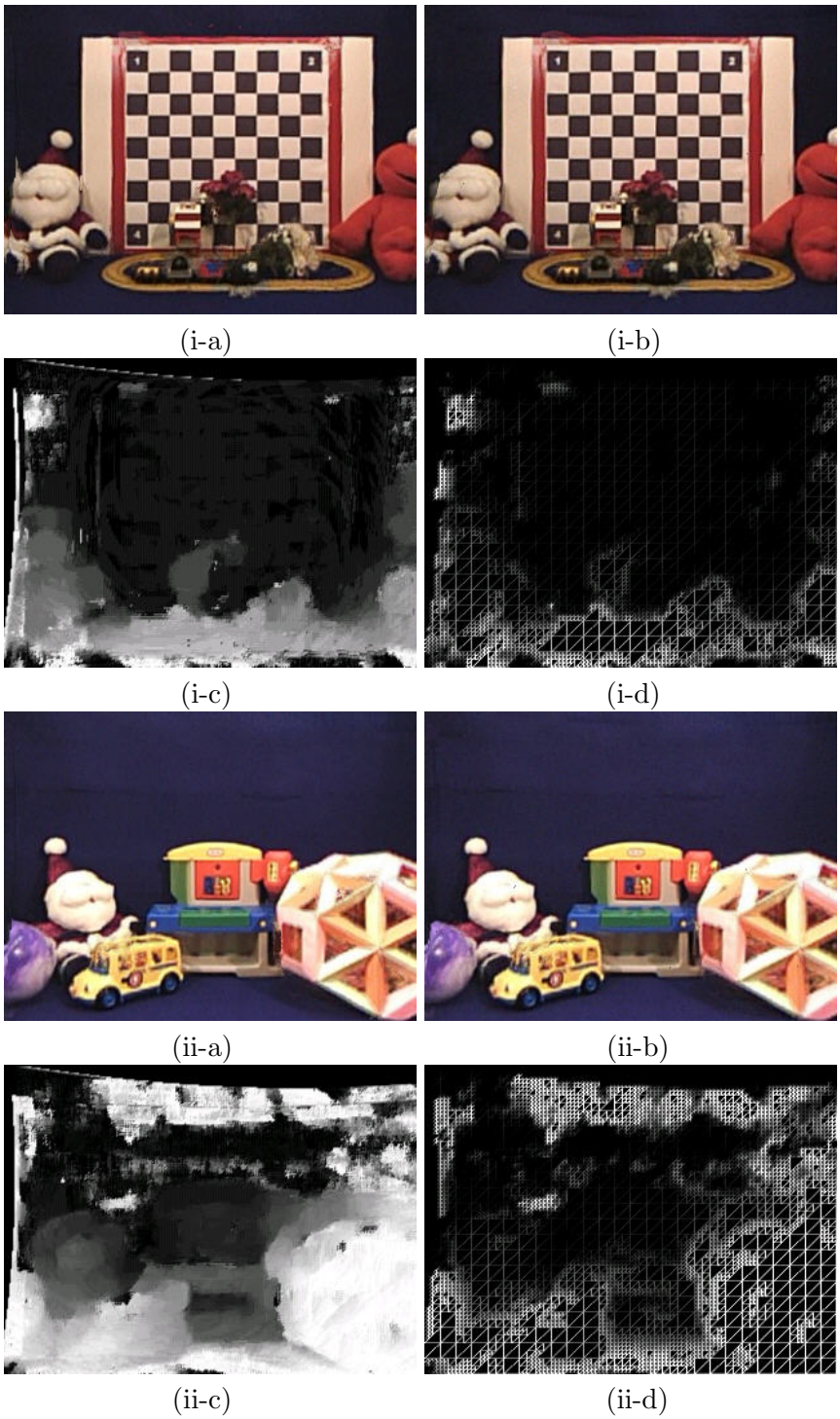


Figure 8.11: Scenes captured and rendered with our camera array. (i) Scene *toys*; (ii) scene *train*; (iii) scene *girl and checkerboard*; (iv) scene *girl and flowers*; (a) rendering with a constant depth at the background; (b) rendering with a constant depth at the middle object; (c) rendering with a constant depth at the closest object; (d) rendering with the proposed method; (e) multi-resolution 2D mesh with depth reconstructed on-the-fly, brighter intensity means smaller depth.



(i-a) (i-b) (i-c) (i-d) (ii-a) (ii-b) (ii-c) (ii-d)

Figure 8.12: Real-world scenes rendered with our proposed algorithm. (i) Scene *train*; (ii) scene *toys*; (a)(c) use a per-pixel depth map to render; (b)(d) use the proposed adaptive mesh to render.

Gigabit Ethernet or using more computers to handle the data could solve this problem. For dynamic scenes, we notice that our system cannot catch up with very fast moving objects. This is due to the fact that the cameras are not synchronized.

We find that when the virtual viewpoint moves out of the range of the input cameras, the rendering quality degrades quickly. Similar effect was reported in [155, 136]. The poor extrapolation results are due to the lack of scene information in the input images during the geometry reconstruction.

Since our geometry reconstruction algorithm resembles the traditional window-based stereo algorithms, they share some limitations. For instance, when the scene has large depth discontinuity, our algorithm does not perform very well along the object boundary (especially when both foreground and background objects have strong textures). In the current implementation, our correlation window has very large size (18×18) in order to tolerate the noisy input images. Such a big correlation window tends to smooth the depth map. Figure 8.13 (i-d) and (iii-d) shows the rendering results of two scenes with large depth discontinuity. Notice the artifacts around the boundaries of the objects. To solve this problem, one may borrow ideas from the stereo literature [51, 53], which will be our future work[2]. Alternatively, since we have built a reconfigurable camera array, we may reconfigure the arrangement of the cameras, as will be described in the next section.

8.4 Self-Reconfiguration of the Cameras

In Section 7.3.1 we have presented an active rearranged sampling algorithm using global rearrangement. In that work we assumed that all the capturing cameras can move freely on the camera plane. Such assumption is very difficult to implement in practical systems. In this section, we present a local rearrangement algorithm for the self-reconfiguration of the cameras, given that they are constrained on the linear guides.

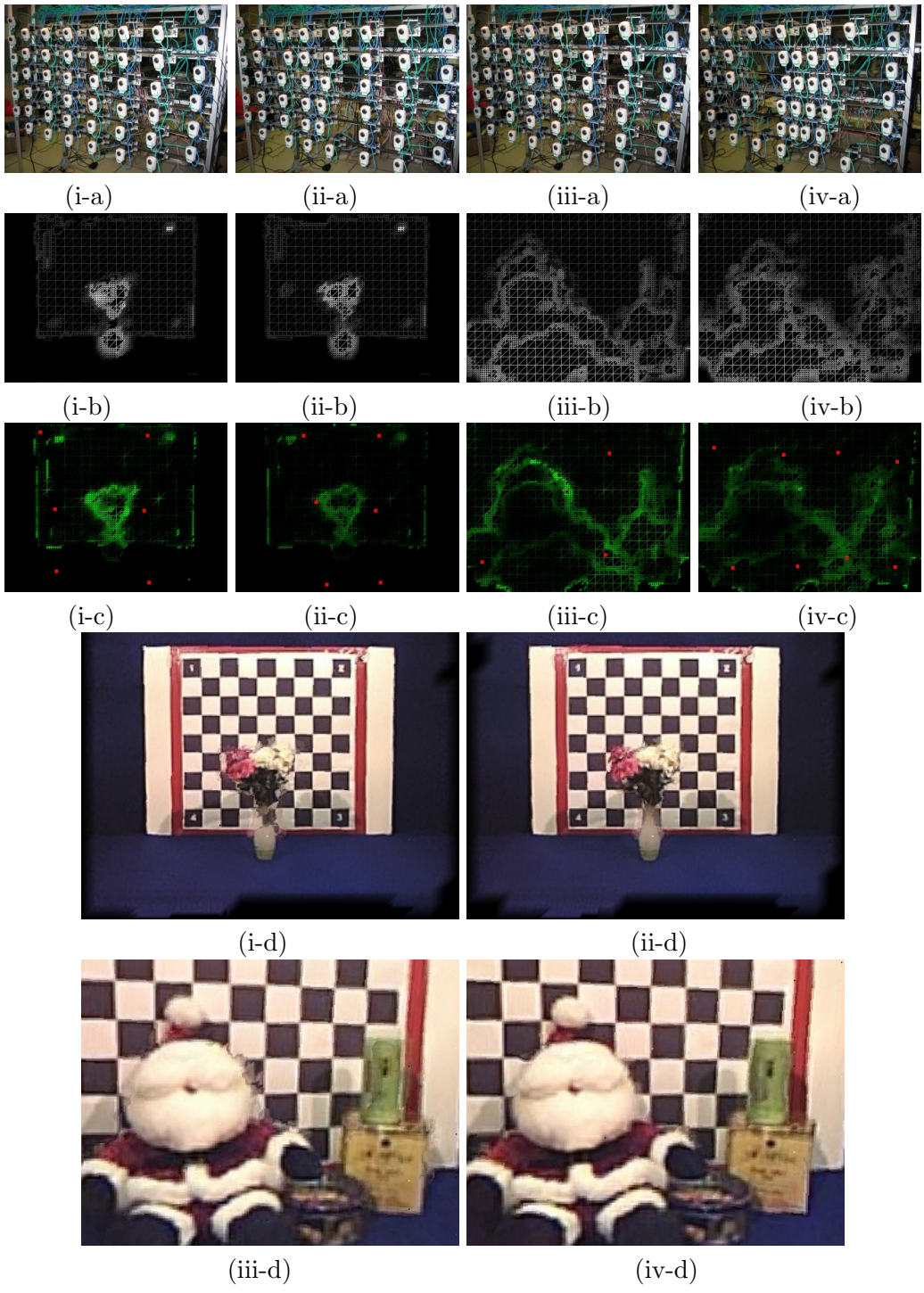


Figure 8.13: Scenes rendered by reconfiguring our camera array. (i) Scene *flower*, cameras are evenly spaced; (ii) scene *flower*, cameras are self-reconfigured (6 epochs); (iii) scene *Santa*, cameras are evenly spaced; (iv) scene *Santa*, cameras are self-reconfigured (20 epochs); (a) the camera arrangement; (b) reconstructed depth map, brighter intensity means smaller depth; (c) the CCV score of the mesh vertices and the projection of the camera positions to the virtual imaging plane (red dots), darker intensity means better consistency; (d) rendered image.

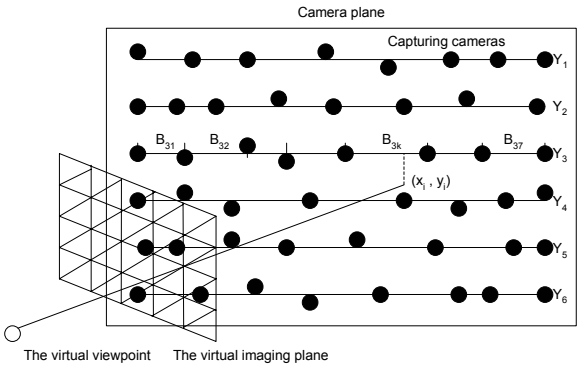


Figure 8.14: Self-reconfiguration of the cameras.

8.4.1 The proposed local rearrangement algorithm

Figure 8.13 (i-c) and (iii-c) shows the local inconsistency score obtained while reconstructing the scene depth (Section 8.3.4). They are directly used for our active rearrange sampling algorithm. It is obvious that if the consistency is bad (high score), the reconstructed depth tends to be wrong, and the rendered scene tends to have low quality. Our camera self-reconfiguration (CSR) algorithm will thus move the cameras to where the score is high.

Our CSR algorithm contains the following steps:

1. *Locate the camera plane and the linear guides* (as line segments on the camera plane). The camera positions in the world coordinate are obtained through the calibration process. Although they are not strictly on the same plane, we use an approximated one which is parallel to the checkerboard. The linear guides are located by averaging the vertical positions of each row of cameras on the camera plane. As shown in Figure 8.14, denote the vertical coordinates of the linear guides on the camera plane as $Y_j, j = 1, \cdots, 6$.

2. *Back-project the vertices of the mesh model to the camera plane.* Although during the depth reconstruction the mesh can subdivided, during this process we only make use of the initial sparse mesh (Figure 8.7). In Figure 8.14, one mesh vertex was back-projected as (x_i, y_i) on the camera plane. Notice such back-projection can be performed even if there are multiple virtual views to be rendered, thus the proposed CSR algorithm is applicable

[2]I am working on this problem right now, and hopefully I can get some result before the defense.

to situations where there exist multiple virtual viewpoints.

3. *Collect CCV score for each pair of neighboring cameras on the linear guides.* The capturing cameras on each linear guide naturally divide the guide into 7 segments. Let them be B_{jk}, where j is the row index of the linear guide, k is the index of bins on that guide, $1 \leq j \leq 6$, $1 \leq k \leq 7$. If a back-projected vertex (x_i, y_i) satisfies

$$Y_{j-1} < y_i < Y_{j+1} \quad and \quad x_i \in B_{jk}, \tag{8.6}$$

the CCV score of the vertex is added to the bin B_{jk}. After all the vertices have been back-projected, we obtain a set of accumulated CCV scores for each linear guide, denoted as S_{jk}, where j is the row index of the linear guide, k is the index of bins on that guide.

5. *Determine which camera to move on each linear guide.* Given a linear guide j, we look for the largest $S_{jk}, 1 \leq k \leq 7$. Let it be S_{jK}. If the two cameras forming the corresponding bin B_{jK} are not too close to each other, one of them will be moved towards the other (thus reducing their distance). Notice each camera is associated with two bins. To determine which one of the two cameras should move, we check their other associated bin and move the camera with a smaller accumulated CCV score in its other associated bin.

6. *Move the cameras.* Once the moving cameras have been decided, we issue them commands such as "move left" or "move right"[3]. Once the cameras are moved, the process waits until it is confirmed that the movement has finished and the cameras are re-calibrated. Then it jumps back to step 1 for the next epoch of movement.

8.4.2 Results

We show results of the proposed CSR algorithm in Figure 8.13. In Figure 8.13 (i) and (iii), the capturing cameras are evenly spaced on the linear guide. Figure 8.13(i) is rendered behind the camera plane, and Figure 8.13(iii) is rendered in front of the camera plane. Due to depth discontinuities, some artifacts can be observed from the rendered images (Figure 8.13 (i-d) and (iii-d)) along the object boundaries. Figure 8.13(b) is the reconstructed depth

[3]We can only send such commands to the sidestep servos, because the servos were hacked for continuous rotation. The positions of the cameras after movement is unpredictable, and can only be obtained through the calibration process.

of the scene at the virtual viewpoint. Figure 8.13(c) is the CCV score obtained during the depth reconstruction. It is obvious that along the object boundaries, the CCV score is high, which usually means wrong/uncertain reconstructed depth, or bad rendering quality. The red dots in Figure 8.13(c) are the projections of the capturing camera positions to the virtual imaging plane.

Figure 8.13 (ii) and (iv) shows the rendering result after CSR. Figure 8.13 (ii) is the result of 6 epochs of camera movement, and Figure 8.13 (iv) is after 20 epochs. It can be seen from the CCV score map (Figure 8.13(c) that after the camera movement, the consistency generally gets better. The cameras have been moved, which is reflected as the red dots in 8.13(c). The cameras moves towards the regions where the CCV score is high, which effectively increases the sampling rate for the rendering of those regions. Figure 8.13 (ii-d) and (iv-d) shows the rendering results after self-reconfiguration, which is much better than 8.13 (i-d) and (iii-d).

8.4.3 Discussions

The major limitation of our self-reconfigurable camera array is that the motion of the cameras are generally slow. When the computer write a command to the serial port, the command will be buffered in the Mini SSC II controller for $\sim$15 ms before sending to the servo. After the servo receives the command, there is also a long delay (hundreds of ms) before it moves enough distance. Therefore, during the self-reconfiguration of the cameras, we have to assume that the scene is either static or moving very slowly, and the viewer is not changing his/her viewpoint all the time. During the motion of the cameras, since the calibration process and the rendering process run separately, we observe some jittering artifacts of the rendered images when the moved cameras have not been fully calibrated.

There is no collision detection in the current system while moving the cameras. Although the calibration process is very stable and gives fairly good estimation of the camera positions, collision could still happen. In 8.4.1, we have a threshold for verifying whether two cameras are too close to each other. The current threshold is set as 10 cm, which is reasonably safe

in all our experiments.

8.5 Summary

We have presented a self-reconfigurable camera array in this paper. Our system was large scale (48 cameras), and had the unique characteristic that the cameras were mounted on mobile platforms. A real-time rendering algorithm was proposed, which is highly efficient and flexible to be implemented in software, thanks to the active incremental sampling based geometry reconstruction algorithm. We also proposed a novel self-reconfiguration algorithm to move the cameras based on active rearranged sampling, and achieved better rendering quality compared with static camera arrays.

Chapter 9

Conclusions and Future Work

This dissertation set out to address the following two questions:

> *How many images are needed for IBR? If such number is limited, where shall we capture these images?*

In this final chapter we summarize the contributions we have presented to answer this question. We also describe some new directions for future work that these contributions raise.

9.1 Contributions

There are three major contributions of this thesis.

Uniform IBR sampling analysis. We proposed a novel method to analyze the Fourier spectrum of IBR scenes, which is able to handle both non-Lambertian surface and occlusions. We showed that in both cases the required sampling rate is higher than Lambertian and non-occluded ones. Considering that the IBR sampling problem is a multi-dimensional sampling problem, we also applied the generalized sampling theorem for IBR sampling. We are able to reduce the sampling rate by a factor of 50% in theory, and achieve better rendering quality for complex scenes. We also concluded that rectangular sampling is still preferable for most scenes thanks to its simplicity.

A very general framework on freeform sampling and active sampling. Compared with the traditional uniform sampling theorem, the freeform sampling framework has more practical considerations such as the reconstruction method, the reconstruction set, the sampling noise, etc. General solutions of freeform sampling were described in this dissertation, including decremental sampling, incremental sampling and rearranged sampling. We also presented active sampling as a special case of freeform sampling, where the function values of the sampled signal on the reconstruction set is unknown. We applied it to IBR and designed several algorithms, which demonstrated that active sampling is superior to the traditional uniform sampling method.

The self-reconfigurable camera array. We built the world's first self-reconfigurable camera array, where the cameras are mobile. We developed a very efficient algorithm for the real-time rendering of dynamic scenes. Active sampling was widely used in the algorithm to improve the rendering speed. The source code of the rendering algorithm was distributed online[1] to inspire more work along this direction. We also showed that by moving the cameras around for active sampling, we can improve the rendering quality, especially at object boundaries.

9.2 Future Work

This dissertation opens up some new interesting directions for further research in various topics. For instance:

IBR sampling analysis with scene geometry. When part of the scene geometry is known or reconstructed, it is not clear what the real minimum sampling rate is. Theoretically, the minimum sampling rate should be determined only by the scene content. However, as discussed in Chapter 6, in practice, the sampling rate will be determined by the reconstruction/rendering algorithm as well. On the other hand, if we are free to apply

[1] http://amp.ece.cmu.edu/projects/MobileCamArray/

the best possible reconstruction/rendering algorithm, what will be the minimum sampling rate? Discovery of such theory is very important for guiding the construction of practical IBR systems.

Real-time processing of dynamic IBR data. While the extension of IBR from static scenes to dynamic scenes seems straightforward, many new research problems arise. For instance, dynamic IBR requires all its stages to be "real-time", as the scene is constantly changing. This includes real-time capturing, real-time storage, real-time calibration, real-time tracking, real-time rendering, real-time compression/streaming, etc. Some of these problems have already been studied in the literature. For example, the Stanford multi-camera array [149] is capable of capturing and storing videos from many cameras at 30 fps to SCSI disk arrays. They used a dedicated hardware board to perform MPEG-2 compression of the captured videos. A relatively complete survey on IBR compression algorithms is available in [161]. Online streaming of light field/concentric mosaics has been studied in [163, 164, 115]. In Chapter 8, we have discussed some solutions for the real-time calibration and rendering from multiple mobile cameras. However, more work is needed, such as a better geometry reconstruction algorithm for heavily occluded scenes. Using our camera array to track moving objects is another problem that is very interesting.

Vision sensor network. We strongly believe that it is beneficial to add more and more functionalities to the sensor (camera), such as compression, networking and mobility. Such migration in functionality from the central computer to the sensors not only reduces the load of the central computer, making the whole system more scalable, but also allow the sensors to distribute in a wider area, making it a true vision sensor network. In practice, due to bandwidth constraints, we expect to have limited resolution on the vision sensors. Synthesizing high-quality novel views from such low-resolution vision sensors is a new problem, and may borrow ideas from the super-resolution literature. The active sampling framework proposed in this dissertation may also be used to figure out the best distribution of these vision sensors.

Multi-view image/video processing. The multiple views captured for a scene can not only be used to perform rendering but also many other tasks. Most current image/video processing research topics can benefit from the availability of multiple views of the same object. To name a few, they include image/video compression, image/video restoration, image/video segmentation, image/video scene analysis, pattern recognition, etc.

Other applications of active sampling. As a very general framework, active sampling may be used in many other applications. Recently there has been increasing interest in the application of active sampling in information retrieval. We have proposed to use active learning for hidden annotation [157]. Tong and Chang have also applied support vector machine (SVM) based active learning for obtaining the user's concept of query [141]. Naphade et al. [96] also used SVM for the active annotation of video databases. Another application of active sampling may be in the area of computer tomography (CT). In CT, one circle of scanning might only cover a small area of the scene in order to improve the precision. Due to the huge cost of storage and reconstruction, active sampling can be applied to select regions that are of most importance, or guarantee that all the scanned regions have the same reconstruction quality. In image-based relighting [104], active sampling may be a good choice to reduce the number of light patterns applied for the scene in order to achieve relighting from arbitrary lighting conditions.

Bibliography

[1] E. H. Adelson and J. R. Bergen, *The plenoptic function and the elements of early vision*, M. Landy and J. A. Movshon (Edt), Computational Models of Visual Processing, The MIT Press, Cambridge, MA (1991), 3–20.

[2] G. Agarwal, D. Rathi, P. K. Kalra, and S. Banerjee, *A system for image based rendering of walk-throughs*, Computer Graphics International, 2002.

[3] A. Aldroubi and K. Gröchenig, *Nonuniform sampling and reconstruction in shift-invariant spaces*, SIAM Review **43** (2001), no. 4, 585–620.

[4] D. G. Aliaga and I. Carlbom, *Plenoptic stitching: a scalable method for reconstructing 3d interactive walkthroughs*, Proc. SIGGRAPH, 2001, pp. 443–450.

[5] S. Avidan and A. Shashua, *Novel view synthesis in tensor space*, Proc. CVPR, 1997.

[6] T. Beier and S. Neely, *Feature-based image metamorphosis*, Proc. SIGGRAPH, 1992, pp. 35–42.

[7] J. F. Blinn and M. E. Newell, *Texture and reflection in computer generated images*, Communications of the ACM **19** (1976), no. 10, 542–546.

[8] S. Borman and R. L. Stevenson, *Super-resolution from image sequences - a review*, Midwest Symposium on Circuits and Systems, 1998.

[9] J.-Y. Bouguet, *Camera calibration toolbox for matlab, http://www.vision.caltech.edu/bouguetj/calib_doc/*, 1999.

[10] A. Broadhurst and R. Cipolla, *A statistical consistency check for the space carving algorithm*, Proc. 11th British Machine Vision Conference, 2000.

[11] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen, *Unstructured lumigraph rendering*, Proc. SIGGRAPH, 2001, pp. 425–432.

[12] S. Sethuraman C. B. Bing and H.S. Sawhney, *A depth map represen-tation for real-time transmission and view-based rendering of a dy-namic 3d scene*, 1st Intl. Symposium on 3D Data Processing Visualization and Transmission, 2002.

[13] B. Cabral, M. Olano, and P. Nemec, *Reflection space image based rendering*, Proc. SIGGRAPH, 1999, pp. 165–171.

[14] E. Camahort and D. Fussell, *A geometric study of light field representations*, Tech. Report TR99-35, Department of Computer Sciences, The University of Texas at Austin, 1999.

[15] E. Camahort, A. Lerios, and D. Fussell, *Uniformly sampled light fields*, 9th Eurographics Workshop on Rendering, 1998.

[16] J.-X. Chai, S.-C. Chan, H.-Y. Shum, and X. Tong, *Plenoptic sampling*, Proc. SIGGRAPH, 2000, pp. 307–318.

[17] C. Chang, G. Bishop, and A. Lastra, *Ldi tree: a hierarchical representation for image-based rendering*, Proc. SIGGRAPH, 1999, pp. 291–298.

[18] S. E. Chen, *Quicktime vr - an image-based approach to virtual environment navigation*, Proc. SIGGRAPH, 1995, pp. 29–38.

[19] S. E. Chen and L. Williams, *View interpolation for image synthesis*, Proc. SIGGRAPH'93, 1993, pp. 279–288.

[20] W.-C. Chen, J.-Y. Bouguet, M. H. Chu, and R. Grzeszczuk, *Light field mapping: efficient representation and hardware rendering of surface light fields*, Proc. SIGGRAPH, 2002, pp. 447–456.

[21] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, *Active learning with statistical models*, Journal of Artificial Intelligence Research **4** (1996), 129–145.

[22] R. T. Collins, *A space-sweep approach to true multi-image matching*, Proc. CVPR, 1996.

[23] T. N. Cornsweet, *Visual perception*, Academic Press, 1971.

[24] IPIX® Internet Pictures Corp., http://www.ipix.com/.

[25] P. Debevec, *Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography*, Proc. SIGGRAPH, 1998, pp. 189–198.

[26] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar, *Acquiring the reflectance field of a human face*, Proc. SIGGRAPH, 2000, pp. 145–156.

[27] P. Debevec, C. J. Taylor, and J. Malik, *Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach*, Proc. SIGGRAPH, 1996, pp. 11–20.

[28] P. Debevec, A. Wenger, C. Tchou, A. Gardner, J. Waese, and T. Hawkins, *A lighting reproduction approach to live-action compositing*, Proc. SIGGRAPH, 2002, pp. 547–556.

[29] P. Debevec, Y.-Z. Yu, and G. Borshukov, *Efficient view-dependent image-based rendering with projective texture-mapping*, 9th Eurographics Rendering Workshop, 1998.

[30] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2 ed., Wiley-Interscience, 2000.

[31] D. E. Dudgeon and R.M. Mersereau, *Multidimensional digital signal processing*, Prentice-hall signal processing series, Prentice Hall, 1984.

[32] P. Eisert, E. Steinbach, and B. Girod, *3-d shape reconstruction from light fields using voxel back-projection*, Vision, Modeling and Visualization Workshop, 1999.

[33] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy, *Real time correlation-based stereo: Algorithm, implementations and applications*, Technical Report 2013, INRIA (1993).

[34] S. Fleishman, D. Cohen-Or, and D. Lischinski, *Automatic camera placement for image-based modeling*, Computer Graphics Forum, 2000.

[35] J. Foote and D. Kimber, *Flycam: practical panoramic video*, Proc. ICME, 2000.

[36] D. A. Forsyth and J. Ponce, *Computer vision: A modern approach*, Prentice Hall, 2002.

[37] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, *The lumigraph*, Proc. SIGGRAPH, 1996, pp. 43–54.

[38] N. Greene and P. Heckbert, *Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter*, IEEE Computer Graphics and Applications **6** (1986), no. 6, 21–27.

[39] N. Greene and M. Kass, *Approximating visibility with environment maps*, Tech. Report 41, Apple Computer, 1994.

[40] Ladybug^{TM} Point Grey, http://www.ptgrey.com/products/ladybug/index.html.

[41] X. Gu, S. J. Gortler, and M. Cohen, *Polyhedral geometry and the two-plane parameterization*, 8th Eurographics Rendering Workshop, 1997.

[42] R. Gupta and R. I. Hartley, *Linear pushbroom cameras*, IEEE Trans. on PAMI **19** (1997), no. 9, 963–975.

[43] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, 2000.

[44] M. Hasenjäger, H. Ritter, and K. Obermayer, *Active learning in self-organizing maps*, E. Oja and S. Kaski (Edt), Kohonen Maps, Elsevier, Amsterdam (1999), 57–70.

[45] H. Hirschmüller, *Improvements in real-time correlation-based stereo vision*, CVPR 2001 Stereo Workshop/IJCV 2002, 2001.

[46] H. Hoppe, *Progressive meshes*, Proc. SIGGRAPH, 1996, pp. 99–108.

[47] I. Ihm, S. Park, and R. Lee, *Rendering of spherical light fields*, Pacific Graphics, 1997.

[48] CBS Broadcasting Inc., http://www.cbs.com/.

[49] A. Isaksen, L. McMillan, and S. J. Gortler, *Dynamically reparameterized light fields*, Proc. SIGGRAPH, 2000, pp. 297–306.

[50] 360 One VR^{TM} Kaidan, http://www.kaidan.com/.

[51] Takeo Kanade and Masatoshi Okutomi, *A stereo matching algorithm with an adaptive window: theory and experiment*, IEEE Trans. on PAMI **16** (1994), no. 9, 920–932.

[52] S.B. Kang, R. Szeliski, and P. Anandan, *The geometry-image representation tradeoff for rendering*, Proc. ICIP, 2000.

[53] Sing Bing Kang, Richard Szeliski, and Jinxiang Chai, *Handling occlusions in dense multi-view stereo*, Proc. CVPR, 2001.

[54] A. Katayama, K. Tanaka, T. Oshino, and H. Tamura, *Viewpoint-dependent stereoscopic display using interpolation of multi-viewpoint images*, Proc. SPIE, vol. 2409, 1995.

[55] H. Kawasaki, K. Ikeuchi, and M Sakauchi, *Light field rendering for large-scale scenes*, Proc. CVPR, 2001.

[56] D. Kimber, J. Foote, and S. Lertsithichai, *Flyabout: spatially indexed panoramic video*, Proc. ACM Multimedia, 2001.

[57] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, *Motion-compensated interframe coding for video conferencing*, Proc. NTC, 1981.

[58] A. Krogh and J. Vedelsby, *Neural network ensembles, cross vali-dation, and active learning*, G. Tesauro, D. Touretzky and T. Leen (Edt), Advances in Neural Information Processing Systems, 7, the MIT Press, Cambridge, MA (1995), 231–238.

[59] A. Kubota and K. Aizawa, *A novel image-based rendering mehtod by linear filtering of multiple focused images acquired by a camera array*, Int. Conf. Image Processing, 2003.

[60] M. P. Lai and W. C. Ma, *A novel four-step search algorithm for fast block motion estimation*, IEEE Trans. CASVT **6** (1996), no. 3, 313–317.

[61] A. Laurentini, *The visual hull concept for silhouette based image understanding*, IEEE trans. on PAMI **16** (1994), no. 2, 150–162.

[62] S. Laveau and O. Faugeras, *3-d scene representation as a collection of images and fundamental matrices*, Tech. Report 2205, INRIA, 1994.

[63] S. Lee, K.-Y. Chwa, J. Hahn, and S.Y. Shin, *Image morphing using deformation techniques*, Journal of Visualization and Computer Animation **7** (1996), no. 1, 3–23.

[64] J. Lengyel, *The convergence of graphics and vision*, IEEE Computer **31** (1998), no. 7, 46–53.

[65] H. Lensch, *Techniques for hardware-accelerated light field rendering*, Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 1999.

[66] M. Levoy and P. Hanrahan, *Light field rendering*, Proc. SIGGRAPH, 1996, pp. 31–42.

[67] M. Lhuillier and L. Quan, *Image interpolation by joint view triangulation*, Proc. CVPR, 1999.

[68] J. Li and J. Kuo, *Progressive coding of 3d graphics models*, Proceedings of the IEEE **86** (1998), no. 6, 1052–1063.

[69] J. Li, K. Zhou, Y. Wang, and H.-Y. Shum, *A novel image-based rendering system with a longitudinally aligned camera array*, Proc. EUROGRAPHICS, 2000.

[70] M. Li, M. Magnor, and H.-P. Seidel, *Hardware-accelerated visual hull reconstruction and rendering*, Proc. Graphics Interface, 2003.

[71] Z. C. Lin and H. Y. Shum, *On the number of samples needed in light field rendering with constant-depth assumption*, Proc. CVPR, 2000.

[72] Z.-C. Lin, T.-T. Wong, and H.-Y. Shum, *Relighting with the reflected irradiance field: representation, sampling and reconstruction*, International Journal of Computer Vision **49** (2002), no. 2-3, 229–246.

[73] Y. Linde, A. Buzo, , and R. Gray, *An algorithm for vector quantizer design*, IEEE Trans. on Communications **28** (1980), no. 1, 84–95.

[74] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, and N. Faust, *Real-time, continuous level of detail rendering of height fields*, Proc. SIGGRAPH, 1996, pp. 109–118.

[75] A. Lippman, *Movie maps: an application of the optical videodisc to computer graphics*, Proc. SIGGRAPH, 1980, pp. 32–43.

[76] D. Lischinski and A. Rappoport, *Image-based rendering for non-diffuse synthetic scenes*, Rendering Techniques, 1998.

[77] B. Lok, *Online model reconstruction for interactive visual environments*, Proc. Symposium on Interactive 3D Graphics, 2001.

[78] W. Lorensen and H. Cline, *Marching cubes: a high resolution 3-d surface construction algorithm*, Proc. SIGGRAPH, 1987, pp. 163–169.

[79] D. Marchand-Maillet, *Sampling theory for image-based rendering*, Master's thesis, EPFL, 2001.

[80] W. R. Mark, L. McMillan, and G. Bishop, *Post-rendering 3d warping*, Proc. 1997 Symposium on Interactive 3D Graphics, 1997.

[81] V. Masselus, P. Peers, P. Dutre, and Y. D. Willems, *Relighting with 4d incident light fields*, Proc. SIGGRAPH, 2003, pp. 613–620.

[82] W. Matusik, C. Buehler, and L. McMillan, *Polyhedral visual hulls for real-time rendering*, Proceedings of Eurographics Workshop on Rendering, 2001.

[83] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan, *Image-based visual hulls*, Proc. SIGGRAPH, 2000, pp. 369–374.

[84] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan, *Image-based 3d photography using opacity hulls*, Proc. SIGGRAPH, 2002, pp. 427–437.

[85] L. McMillan, *An image-based approach to three-dimensional computer graphics*, Ph.D. thesis, Department of Computer Science, University of North Carolina at Chapel Hill, 1997.

[86] L. McMillan and G. Bishop, *Plenoptic modeling: an image-based rendering system*, Proc. SIGGRAPH, 1995, pp. 39–46.

[87] J. Meehan, *Panoramic photograph*, Watson-Guptill, 1990.

[88] D. L. Milgram, *Computer methods for creating photomosaics*, IEEE Trans. on Computers **24** (1975), no. 11, 1113–1119.

[89] G. Miller, E. Hoffert, S. E. Chen, E. Patterson, D. Blackketter, S. Rubin, S. A. Aplin, D. Yim, and J. Hanan, *The virtual museum: interactive 3d navigation of a multimedia database*, The Journal of Visualization and Computer Animation **3** (1992), no. 3, 183–197.

[90] G. Miller, S. Rubin, and D. Ponceleon, *Lazy decompression of surface light fields for precomputed global illumination*, Eurographics Rendering Workshop, 1998.

[91] MiniSSC-II, *Scott edwards electronics inc., http://www.seetron.com/ssc.htm.*

[92] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *Mpeg video: Compression standard*, Kluwer Academic Publishers, 1996.

[93] J. J. Moré, *The levenberg-marquardt algorithm, implementation and theory*, G. A. Watson, editor, Numerical Analysis, Lecture Notes in Mathematics **630** (1977), 105–116.

[94] J. Mulligan, V. Isler, and K. Daniilidis, *Trinocular stereo: a real-time algorithm and its evaluation*, IEEE Workshop on Stereo and Multi-Baseline Vision, 2001.

[95] T. Naemura, J. Tago, and H. Harashima, *Real-time video-based modeling and rendering of 3d scenes*, IEEE Computer Graphics and Applications **22** (2002), no. 2, 66–73.

[96] M. R. Naphade, C. Y. Lin, J. R. Smith, B. Tseng, and S. Basu, *Learning to annotate video databases*, SPIE Conference on Storage and Retrieval on Media databases, 2002.

[97] S. Nayar, *Catadioptric omnidirectional camera*, Proc. CVPR, 1997.

[98] U. Neumann, T. Pintaric, and A. Rizzo, *Immersive panoramic video*, Proc. ACM Multimedia, 2000.

[99] T. Nishita, T. Fujii, and E. Nakamae, *Metamorphosis using bézier clipping*, 1st Pacific Conference on Computer Graphics and Applications, 1993.

[100] M. Oliveira, *Image-based modeling and rendering techniques: a survey*, RITA - Revista de Informática Teórica e Aplicada **IX** (2002), no. 2, 37–66.

[101] M. Oliveira and G. Bishop, *Relief textures*, Tech. Report TR99-015, Dept. of Computer Science, University of North Carolina, 1999.

[102] A. V. Oppenheim, A. S. Willsky, S. N. Nawab, S. H. Nawab, H. Nawad, and S. H. Nawab, *Signals and systems*, 2 ed., Prentice Hall, 1996.

[103] J. O'Rourke, *Art gallery theorems and algorithms*, The International Series of Monographs on Computer Science, Oxford University Press, 1987.

[104] P. Peers and P. Dutré, *Wavelet environment matting*, Proc. 14th Eurographics Workshop on Rendering, 2003.

[105] S. Peleg and M. Ben-Ezra, *Stereo panorama with a single camera*, Proc. CVPR, 1999.

[106] S. Peleg and J. Herman, *Panoramic mosaics by manifold projection*, Proc. CVPR, 1997.

[107] S. Peleg, B. Rousso, A. Rav-Acha, and A. Zomet, *Mosaicing on adaptive manifolds*, IEEE Trans. on PAMI **22** (2000), no. 10, 1144–1154.

[108] W. B. Pennebaker and J. L. Mitchell, *Jpeg: Still image data compression standard*, 1 ed., Kluwer Academic Publishers, 1993.

[109] B. T. Phong, *Illumination for computer generated pictures*, Communications of the ACM **18** (1975), no. 6, 311–317.

[110] R. Pito, *A solution to the next best view problem for automated surface acquisition*, IEEE Trans. on PAMI **21** (1999), no. 10, 1016–1030.

[111] Pov-Ray, *http://www.povray.org*.

[112] P. Rademacher, *View-dependent geometry*, Proc. SIGGRAPH, 1999, pp. 439–446.

[113] P. Rademacher and G. Bishop, *Multiple-center-of-projection images*, Proc. SIGGRAPH, 1998, pp. 199–206.

[114] R. Ramamoorthi and P. Hanrahan, *A signal-processing framework for inverse rendering*, Proc. SIGGRAPH, 2001, pp. 117–128.

[115] P. Ramanathan, M. Kalman, and B. Girod, *Rate-distortion optimized streaming of compressed light fields*, Proc. ICIP, 2003.

[116] M. K. Reed, *Solid model acquisition from range images*, Ph.D. thesis, Columbia University, 1998.

[117] D. G. Ripley, *Dvi-a digital multimedia technology*, Communications of the ACM **32** (1989), no. 7, 811–822.

[118] A. Said and W. A. Pearlman, *A new fast and efficient image codec based on set partitioning in hierarchical trees*, IEEE Trans. on CSVT **6** (1996), no. 3, 243–250.

[119] H. S. Sawhney and R. Kumar, *True multi-image alignment and its application to mosaicing and lens distortion correction*, IEEE Trans. PAMI **21** (1999), no. 3, 235–243.

[120] D. Scharstein and R. Szeliski, *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms*, International Journal of Computer Vision **47** (2002), no. 1/2/3, 7–42.

[121] H. Schirmacher, W. Heidrich, and H. P. Seidel, *Adaptive acquisition of lumigraphs from synthetic scenes*, Proc. EUROGRAPHICS, 1999.

[122] ———, *High-quality interactive lumigraph rendering through warping*, Graphics Interface, 2000.

[123] H. Schirmacher, M. Li, and H.-P. Seidel, *On-the-fly processing of generalized lumigraphs*, Proc. EUROGRAPHICS, 2001.

[124] S. M. Seitz and C. R. Dyer, *Physically-valid view synthesis by image interpolation*, Proc. Workshop on Representation of Visual Scenes, 1995.

[125] ———, *View morphing*, Proc. SIGGRAPH, 1996, pp. 21–30.

[126] ———, *Photorealistic scene reconstruction by voxel coloring*, Proc. CVPR, 1997.

[127] J. Shade, S. Gortler, L.-W. He, and R. Szeliski, *Layered depth images*, Proc. SIGGRAPH, 1998, pp. 231–242.

[128] H.-Y. Shum and L.-W. He, *Rendering with concentric mosaics*, Proc. of SIGGRAPH, 1999, pp. 299–306.

[129] H.-Y. Shum, L.-F. Wang, J.-X. Chai, and X. Tong, *Rendering with manifold hopping*, International Journal of Computer Vision **50** (2002), no. 2, 185–201.

[130] G. G. Slabaugh, R. W. Schafer, and M. C. Hans, *Image-based photo hulls*, Tech. Report HPL-2002-28, HP Labs, 2002.

[131] P. P. Sloan, M. F. Cohen, and S. J. Gortler, *Time critical lumigraph rendering*, Symposium on Interactive 3D Graphics, 1997.

[132] R. Swaminathan and S.K. Nayar, *Polycameras: camera clusters for wide angle imaging*, Tech. Report CUCS-013-99, Columbia University, 1999.

[133] R. Szeliski, *Image mosaicing for tele-reality applications*, Tech. Report CRL94/2, DEC Cambridge Research Lab, 1994.

[134] R. Szeliski and H.-Y. Shum, *Motion estimation with quadtree splines*, IEEE Trans. PAMI **18** (1996), no. 12, 1199–1210.

[135] ———, *Creating full view panoramic image mosaics and texture-mapped models*, Proc. SIGGRAPH, 1997, pp. 251–258.

[136] Richard Szeliski, *Prediction error as a quality metric for motion and stereo*, Proc. ICCV, 1999.

[137] D. S. Taubman and M. W. Marcellin, *Jpeg2000: Image compression fundamentals, standards, and practice*, Kluwer Academic Publishers, 2001.

[138] TotalView^{TM} Be Here Technologies, http://www.behere.com/.

[139] S. Teller, M. Antone, Z. Bodnar, M. Bosse, S. Coorg, M. Jethwa, and N. Master, *Calibrated, registered images of and extended urban area*, International Journal of Computer Vision **53** (June 2003), 93–107.

[140] Carnegie Mellon Goes to the Super Bowl, http://www.ri.cmu.edu/events/sb35/tksuperbowl.html.

[141] S. Tong and E. Chang, *Support vector machine active learning for image retrieval*, Proc. ACM Multimedia, 2001.

[142] M. Turk and A. Pentland, *Eigenfaces for recognition*, Journal of Cognitive Neuroscience **3** (1991), no. 1, 71–86.

[143] M. Unser, *Sampling - 50 years after shannon*, Proceedings of the IEEE **88** (2000), no. 4, 569–587.

[144] M. Uyttendaele, A. Eden, and R. Szeliski, *Eliminating ghosting and exposure artifacts in image mosaics*, Proc. CVPR, 2001.

[145] P. P. Vaidyanathan and T. Q. Nguyen, *Eigenfilters: a new approach to least-squares fir filter design and applications including nyquist filters*, IEEE Trans. on CAS **34** (1987), no. 1, 11–23.

[146] S. Vedula, S. Baker, and T. Kanade, *Spatio-temporal view interpolation*, 13th ACM Eurographics Workshop on Rendering, 2002.

[147] T. Werner, R. D. Hersch, and V. Hlavác, *Rendering real-world objects using view interpolation*, Proc. ICCV, 1995.

[148] T. Werner, V. Hlavác, A. Leonardis, and T. Pajdla, *Selection of reference views for image-based representation*, Proc. ICPR, 1996.

[149] B. Wilburn, M. Smulski, H.-H. K. Lee, and M. Horowitz, *The light field video camera*, Proc. of Media Processors, 2002.

[150] G. Wolberg, *Digital image warping*, IEEE Computer Society Press, 1990.

[151] T.-T. Wong, C. W. Fu, P.-A. Heng, and C.-S. Leung, *The plenoptic illumination function*, IEEE Trans. on Multimedia **4** (2002), no. 3, 361–371.

[152] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle, *Surface light fields for 3d photography*, Proc. SIGGRAPH, 2000, pp. 287–296.

[153] D. N. Wood, A. Finkelstein, J. F. Hughes, C. E. Thayer, and D. H. Salesin, *Multiperspective panoramas for cel animation*, Proc. SIGGRAPH, 1997, pp. 243–250.

[154] Y. Xiong and K. Turkowski, *Creating image-based vr using a self-calibration fisheye lens*, Proc. CVPR, 1997.

[155] Jason C. Yang, Matthew Everett, Chris Buehler, and Leonard McMillan, *A real-time distributed light field camera*, Eurographics Workshop on Rendering, 2002.

[156] R. Yang, G. Welch, and G. Bishop, *Real-time consensus-based scene reconstruction using commodity graphics hardware*, Proc. Pacific Graphics, 2002.

[157] C. Zhang and T. Chen, *An active learning framework for content-based information retrieval*, IEEE Trans. on Multimedia **4** (2002), no. 2, 260–268.

[158] ———, *Active scene capturing for image-based rendering*, Tech. Report AMP03-02, Carnegie Mellon University, 2003.

[159] ———, *Non-uniform sampling of image-based rendering data with the position-interval error (pie) function*, Proc. VCIP, 2003.

[160] ———, *Spectral analysis for sampling image-based rendering data*, IEEE Trans. on CSVT **13** (2003), no. 11, 1038–1050.

[161] ———, *A survey on image-based rendering - representation, sampling and compression*, EURASIP Signal Processing: Image Communication **19** (2004), no. 1, 1–28.

[162] ———, *View-dependent non-uniform sampling for image-based rendering*, Proc. ICIP, 2004.

[163] C. Zhang and J. Li, *Compression of lumigraph with multiple reference frame (mrf) prediction and just-in-time rendering*, IEEE Data Compression Conference, 2000.

[164] ———, *Interactive browsing of 3d environment over the internet*, Proc. VCIP, 2001.

[165] Z. Zhang, *A flexible new technique for camera calibration*, Technical Report, MSR-TR-98-71 (1998).

[166] Z.-P. Zhang, L.-F. Wang, B.-N. Guo, and H.-Y. Shum, *Feature-based light field morphing*, Proc. SIGGRAPH, 2002, pp. 457–464.

[167] Z.-Y. Zhang, *Image-based geometrically-correct photorealistic scene/object modeling (ibphm): a review*, Asian Conference on Computer Vision, 1998.

[168] J. Y. Zheng and S. Tsuji, *Panoramic representation of scenes for route understanding*, Proc. ICPR, 1990.