MASTER

Region-based all-focused light field rendering using color-based focus measure

Kadermohideen Shahulhameed, A.A.

Award date:
2008

Link to publication
REGION-BASED ALL-FOCUSED LIGHT FIELD RENDERING USING COLOR-BASED FOCUS MEASURE

By
Kadermohideen Shahulhameed A.A.

Supervisor: Peter H. N.de With
Tutor: Goran Petrovic

October 2008
Abstract

This thesis builds on previous research in the broad area of image-based rendering and light field rendering in particular. The images synthesized using light field rendering exhibit aliasing artifacts when the light field is under-sampled. Under-sampling in light field rendering is necessary as otherwise, the size of the data-set increases dramatically, thus creating problems in data storage and data transmission. To render under-sampled light fields without aliasing we experimented with several existing techniques and found that the rendering quality of existing algorithms could be further improved. The main contribution of this thesis is a novel region-based “all-in-focus” light field rendering algorithm. A salient feature of our algorithm is that it requires no prior knowledge of the scene geometry. Instead, it uses a set of depth layers and renders each pixel of the synthesized image by assigning it to one of the depth layers. To map a pixel to a depth layer we use a color-based focus measure. We further improve the final rendering quality using a novel region-based template matching. The complete algorithm was implemented and the results compared to similar techniques for all-in-focus rendering using under-sampled light fields. The comparisons validate our approach and show that our algorithm yields better results compared to the existing techniques in all tested rendering scenarios.
Acknowledgements

I would first like to express my sincere thanks and gratitude to my supervisor Professor. Peter H. N.de With who showed me the importance of sincere efforts in research and for suggesting an interesting and challenging problem. My thanks are also to my mentor Goran Petrovic, for feedbacks that helped me direct my research. I would also like to thank the members of VCA particularly Sveta Zinger, Ping Li and Jungong Han for interesting and fruitful discussions. I would also like to express my deep gratitude towards the friends, particularly, Schobben and Zhenyu ye who gave a willing ear during my tough times. Lastly and yet perhaps most importantly, the constant support of my family helped me in every way possible to finish my work.
# Table of Contents

Abstract iii

Acknowledgements v

List of Tables ix

List of Figures xi

1 Introduction 1
   1.1 Image based rendering 2
   1.2 Problem Statement and approach 3
   1.3 Outline of the Thesis 4

2 Previous Work 5
   2.1 Two-Plane Parameterization 6
   2.2 Capturing a light field 6
   2.3 Rendering virtual views 9
   2.4 Dynamic Reparameterization 10
   2.5 Aliasing in light field rendering 12
   2.6 Frequency Domain Analysis of light fields 14
   2.7 All Focused Light Field Rendering 15
   2.8 Takahashi’s Algorithm 18

3 Proposed Method 23
   3.1 Motivation 23
   3.2 Color based focus measure 24
   3.3 Rendering for each Layer 27
   3.4 Region based All-focus rendering 28
   3.5 Normalized cuts for Image segmentation 31
   3.6 Complete Algorithm 32

4 Implementation and Results 35

vii
List of Tables

4.1 Light field information .............................................. 36
4.2 Analysis using RMSE on ‘Pebbles’ ................................ 38
4.3 Analysis using RMSE on ‘Jewel’ .................................. 42
4.4 Analysis using NSEPP on ‘Jewel’ ................................. 42
4.5 Analysis using NSEPP on ‘Pebbles’ ............................... 43
List of Figures

1.1 Image Based Rendering ............................................. 2
1.2 IBR spectrum ....................................................... 3
1.3 Traditional lightfield rendering ................................. 4

2.1 Light field Parameterization ....................................... 7
2.2 Building a light field database .................................... 7
2.3 Light Field Cameras ................................................. 8
2.4 Light field visualization ............................................. 9
2.5 Synthesizing virtual views ......................................... 10
2.6 Dynamic Reparameterization ...................................... 11
2.7 Frequency domain Analysis ....................................... 15
2.8 Filter in Frequency Domain ....................................... 16
2.9 Filter implementations ............................................. 17
2.10 Takahashi’s algorithm .............................................. 19
2.11 Logic behind Takahashi’s algorithm ......................... 21

3.1 Classification of focal planes .................................... 24
3.2 Lightfield Rendering revisited .................................... 25
3.3 Point-Ray Duality .................................................. 26
3.4 Wrapping Images .................................................. 29
3.5 Template Matching ................................................ 30
3.6 Image Segmentation ................................................. 31
3.7 Novel Algorithm ..................................................... 33
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 All-focused rendering using our algorithm</td>
<td>36</td>
</tr>
<tr>
<td>4.2 All-focused rendering using Takahashis algorithm</td>
<td>37</td>
</tr>
<tr>
<td>4.3 Comparison of results obtained using our algorithm with the ground truth</td>
<td>39</td>
</tr>
<tr>
<td>4.4 Histogram comparison</td>
<td>40</td>
</tr>
<tr>
<td>4.5 Comparison to existing algorithms</td>
<td>41</td>
</tr>
<tr>
<td>5.1 Spatial frequency based fusion</td>
<td>47</td>
</tr>
<tr>
<td>5.2 Wavelet based Image Fusion</td>
<td>48</td>
</tr>
<tr>
<td>5.3 Evaluation of Wavelet based methods</td>
<td>48</td>
</tr>
<tr>
<td>5.4 Bayesian Labeling cost</td>
<td>49</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The main focus of computer graphics is to synthesize and manipulate visual content \cite{SCK06}. Traditionally this was achieved using geometric primitives. The first step in this methodology is to generate models of the environment using a suitable CAD modeler and then represent it using a suitable polygon based description. This is followed by the actual creation of virtual view - image rendering.

Photorealism is a natural goal for image rendering, and in traditional graphics this goal requires complex object modeling. The pursuit of this has pushed the complexity of 3D models so much that the geometric primitives can now be smaller than one pixel in the final image. It is especially true when real-life features like reflection, refraction are part of the synthetic scene. The entire process of creating 3D models can therefore be an expensive and time consuming process requiring constant human interaction. The computation cost involved in synthesizing virtual views is also dependent on the scene complexity, i.e., the number of triangle primitives used in the geometric representation. Formerly, when models were simple and triangle primitives large, the ability to specify large connected regions with only three points was very efficient requiring limited storage and computation. Now that the 3D models contain nearly as many primitives as pixels in the final image, fast rendering of complicated scenes requires expensive graphics accelerators.
Recently, there emerged a competing means of synthesizing virtual views called Image Based Rendering (IBR) \[SCK06\]. In contrast to traditional geometry based approach, IBR techniques rely on images to synthesize virtual views of the scene as shown in Figure 1.1. IBR combines two disciplines, Computer Graphics and Computer Vision and uses the best of both techniques to render realistic images.

1.1 Image based rendering

IBR offers several attractive advantages over traditional geometry-based approach. First, it greatly simplifies the process of generating models of real world environments. In principle, it is possible to almost immediately reconstruct models of real-world environments by capturing images of the environment. Secondly, the synthesized images are naturally photo-realistic as the source data are actual images themselves. Finally, the cost of rendering is independent of scene complexity as the time taken to render an image depends only on the number of images to be processed \[SCK06\].

Most of the IBR algorithms in use today can be divided into one of three classes - methods with no geometry, with implicit geometry and with explicit geometry as shown in Figure 1.2 \[SCK06\]. Methods with no geometry rely on selecting samples from source images and interpolating between them. They assume no additional information about the geometry of the scene. Methods with implicit geometry have some indirect information about the structure of the scene (like point correspondences) to improve reconstruction. Methods with explicit geometry often use dense depth information about the scene, which significantly improves the reconstruction quality. However, acquiring depth information for a real scene is in general a

\[^{1}\text{LDI is Layered Depth Image}\]
difficult problem, hence we will focus our attention on methods that require little or no geometry. Specifically, we concentrate on light field rendering approach which does not use any scene geometry to render virtual views.

1.2. PROBLEM STATEMENT AND APPROACH

The light field rendering approach was first suggested by Levoy and Hanrahan [LH96] is an IBR algorithm that does not use any geometric information for synthesizing virtual views. It instead relies only on dense samples of the light field to synthesize virtual views. One of the major problems with light field rendering is its requirement of high sampling density. The sampling density required usually is impractically high and any reduction in sampling density causes serious ghosting and aliasing artifacts in synthesized images.

The Figure 1.3 shows an example of using traditional light field rendering techniques on sub-sampled light fields. The synthesized images contain either the background or foreground in focus while the rest of the scene contains ghosting and aliasing artifacts. The objective of this thesis is to counter this and generate an “all-in-focus” image using a sparse light field.

To achieve this goal, we drew inspiration from a similar rendering technique called Lumigraph...
CHAPTER 1. INTRODUCTION

(a) Focus on foreground  
(b) Focus on background

Figure 1.3: Traditional light field rendering using subsampled ‘Toys’ dataset

rendering, introduced by Gortler et al. [LH96]. Lumigraph is quite similar to light field rendering but it uses a scene model to counter and reconstruct virtual views from a sparse set of multi-view images. These scene models are usually acquired using hardware such as range finder. These hardware requirements however, puts a limitation on lumigraph and hence, most lumigraphs have been limited to scenes composed of a single object or a small cluster of scene elements [SCK06]. In contrast our work is aimed at real scenes so we attempted to acquire the geometry from the captured images and use it in light field rendering.

1.3 Outline of the Thesis

In this section we will give an outline of the thesis. Background on Image based rendering and light fields are covered in Chapter 2. In this chapter we also look at some techniques to improve rendering from under-sampled light fields. An alternate methodology to handle under-sampled light fields is proposed in Chapter 3. Next we discuss our implementation and the results obtained using our algorithm in Chapter 4. In the following Chapter 5 we discuss some other algorithms that we investigated and implemented. However the performance of these algorithms were lesser than existing algorithms. In Chapter 6 we give our conclusions on the project and some choices for extending the present work.
Chapter 2

Previous Work

In this chapter we will examine the development of IBR as a rendering technique, and examine previous works on IBR in general and light field rendering in particular. All IBR methods are essentially based on modeling the Plenoptic function. First proposed by Adelson and Bergen [AB91] as a 7-D function,

\[ P_7(x, y, z, \theta, \phi, \lambda, t) \]  

the Plenoptic function parameterizes every light ray using parameters - location \((x, y, z)\), orientation \((\theta, \phi)\), wavelength \(\lambda\) and time \(t\). The complete knowledge of this function would enable reconstruction of a virtual view, from any point in space and at any time. By removing two variables time \(t\) (assuming a static scene) and wavelength \(\lambda\), McMillan and Bishop [MB95] introduced the notion of Plenoptic modelling with their 5D complete Plenoptic function,

\[ P_5(x, y, z, \theta, \phi) \]  

They were also one of the first to look at IBR as a signal resampling problem. Levoy and Hanrahan [LH96] further simplified the 5D Plenoptic function to 4D by a simple observation that properties (intensity or chromacity) of light rays does not change along a line unless the light ray is blocked. Hence in an occlusion free and non-dispersive space we can simplify the 7D complete plenoptic function to a 4D light field function. This 4D function completely describes the flow of light through unobstructed space in a static scene with fixed illumination. Any image of the scene can be considered as a set of samples of this light field function. However,
parameterizing rays taking this fact into account is still a tricky problem. Levoy and Hanrahan \cite{LH96} came up with an interesting solution to record rays by their intersections with two planes.

2.1 Two-Plane Parameterization

We will now examine how to parameterize and store a light field using 2 planes. As mentioned earlier, the transparent space around an illuminated object is filled with light reflected off the surface of object. Each of the light rays reflected from the scene can be parameterized using the intersection of the light ray with two planes \((u, v)\) and \((s, t)\) as,

\[
P_4(u, v, s, t)
\]  \hspace{1cm} (2.3)

The parameterizing technique is shown in Figure 2.1. The two planes of course must be suitably positioned between the capturing camera and the scene. The plane indexed with \((u, v)\) is where the cameras are positioned and is referred to as the camera plane. The other plane indexed with \((s, t)\) is usually referred to as the image plane. In Figure 2.1 the light ray shown is parameterized with \((u', v', s', t')\). Similarly, we can store each ray in a database indexed with a suitable combination and query the database for a ray using the same combination. Figure 2.2 shows this in a 2D context. Though the two plane parameterizations is the most popular method to store and query a light field, other parameterizing techniques like sphere have been investigated in literature \cite{Levo6}. In the following sections we will describe how to capture a light field and synthesize virtual views using it.

2.2 Capturing a light field

Specialist camera arrays are usually required to capture light fields of real world objects. Yang et al. \cite{YEBM02} built a realtime distributed light field camera from an 8x8 grid of webcams for real-time light field rendering of dynamic scenes. Wilburn et al. \cite{WJV05} used 100 custom video cameras to construct an unique camera array. The camera designed by Wilburn is shown...
2.2. CAPTURING A LIGHT FIELD

Figure 2.1: The light ray shown is parameterized with \((u', v', s', t')\)

Figure 2.2: Duality explored by Gu et al. [GGC97] shows a light ray can be defined by two points, hence rendering a view is just a simple query of the database
in Figure 2.3(a). These camera arrays use the traditional uniform rectangular distribution of cameras over the camera plane. The advantage of using camera arrays to capture light fields is the control it gives over the sampling pattern. However, it is quite difficult to build a camera array [YEBM02]. Figure 2.4 shows the visualization of the light field captured by Levoy and Hanrahan [LH96] using a camera array. Lot of effort is also spend on building smaller and commercially viable light field cameras. The Figure 2.3(b) shows one successful construction of light field camera by Adobe. Light fields can also be captured using a hand held camera, however it is difficult to control the sampling pattern. An interesting work, on capturing light fields using hand held camera was done by Sharma et al. [SPBK02], who showed the feasibility of capturing light fields and rendering virtual views using an uncalibrated hand-held camera.

(a) Stanford Camera Array  
(b) Adobe Light field Camera

Figure 2.3: Light Field Cameras
2.3. RENDERING VIRTUAL VIEWS

Figure 2.4: The light field seen from (a) camera plane, and (b) focal plane. The boxed subimage is observed from a single point in the parameter plane. (Reproduced from [LH96])

2.3 Rendering virtual views

Synthesizing virtual views using a light field is trivial compared to capturing and parameterizing it. Levoy and Hanrahan [LH96] first demonstrated the suitability of densely sampled light field function to synthesize virtual views. The synthesis process is shown in Figure 2.5. To synthesize a view from a virtual camera \( C \) using the ray-shooting scheme, one has to identify the intersection of each viewing ray shot from \( C \) with the two planes. The points of intersection \((u_0, v_0)\) and \((s_0, t_0)\) characterize this viewing ray, and using this four tuple \((u_0, v_0, s_0, t_0)\) and previously obtained radiance values, the novel view from \( C \) is accurately constructed. Since we have only a discrete set of light rays, the rendering step usually resorts to different forms of approximations like nearest-neighbor, bilinear or quadrilinear interpolation to get the radiance value of the ray at \( C \).

\(^1\)parameterizing using 2D plane
CHAPTER 2. PREVIOUS WORK

Figure 2.5: To synthesize novel views shoot a ray from the camera center for each pixel and using the intersection point pick the appropriate ray from the lightfield database.

2.4 Dynamic Reparameterization

The algorithm of Section 2.3 assumes the object is close to the image plane and uses the parametrization technique described in Section 2.1 to synthesize virtual views. As we know the image plane has to be fixed at a particular depth before parameterizing the captured light field. This forces apriori decision as to what part of the scene can be rendered without aliases. This of course is undesirable. To overcome this Isaksen et al. [IMG00] suggested “dynamic re-parameterization” where they introduced the concept of “focal plane” whose position depends on the user. The parameterization of a light ray in this technique is not fixed but rather depends on the choice of focal plane. A detailed description is given in Isaksen’s paper [IMG00]. To synthesize a virtual view using dynamic re-parameterization, the algorithm assumes the scene elements are located close to the chosen focal plane. Following this, a ray corresponding to a required pixel is shot and the intersection of this ray with the focal plane is identified. Next we find the pixels in the input camera samples which correspond to this point on the focal plane by back projecting it to the input cameras. The approach is motivated using a 2D example...
2.4. DYNAMIC REPARAMETERIZATION

(a) Given a ray \((u, v, f, g)\) at \(F\), one finds the rays \((u', v', s', t')\) and \((u'', v'', s'', t'')\) in the data cameras which intersect \(F\) at the same point \((f, g)\). The parameterization of light rays depends on the choice of \(F\) hence the name 'dynamic re-parameterization'.

(b) Wide Aperture Rendering (Reproduced from [IMG])

Figure 2.6: Dynamic Reparameterization
shown in Figure 2.6(a) where the ray \( r \) is interpolated using re-parameterization. Throughout our work, we use only re-parameterization to generate virtual views. Isaken et al. [IMG00] also introduced the concept of Wide-aperture reconstruction shown in figure 2.6(b). Wide-aperture reconstruction basically simulates a non-pinhole camera where a pixel in the final image depends on intensity contribution of a set of rays rather than the closest two rays.

### 2.5 Aliasing in light field rendering

Light field rendering, as with any rendering scheme is susceptible to aliasing artifacts. The amount of aliasing artifacts is closely related to light field sampling density [LH96]. The relation between sampling density and non-aliased rendering was first explored by Chai et al. [CTCS00]. To avoid aliasing altogether they suggested that an object’s light field must be sampled densely enough such that the maximum parallax disparity between adjacent images does not exceed one pixel. Otherwise, when attempts are made to reconstruct virtual view from under-sampled light fields, it is not possible to avoid aliasing. The number of images required to guarantee less-than-one-pixel disparity between adjacent images is disproportionately high, especially when it is a real world scene. Thus, physically recorded light fields are almost always a under-sampled representation of the complete light-field information.

Levoy and Hanrahan [LH96] had suggested that light field aliasing could be eliminated with proper prefiltering. Prefiltering can be accomplished optically by using a camera whose aperture is at least as large as the spacing between cameras. Otherwise, prefiltering can be accomplished by initially over-sampling along the camera-spacing dimensions and then applying a discrete low-pass filter, which models a synthetic aperture. In practice, it is usually impractical to do over-sampling since it requires cameras be positioned very close to each other or that they large aperture. Moreover, having a large aperture causes “depth of field” effects where only a part of image is in sharp focus. Hence, it has the undesired side effect of forcing an a priori decision as to what parts of the scene can be rendered in focus during reconstruction. Therefore the only practical way to reducing aliasing in conventional light field is to use band-limited
2.5. ALIASING IN LIGHT FIELD RENDERING

However, low-pass filtering removes desired high frequencies like sharp edges and view-dependencies, and hence results in blurred reconstructions as shown in Figure 2.9(d).

As described in Section 2.4, Isaksen et al. [IMG00] extended light-field rendering by introducing a movable focal surface, and demonstrated that focus-like effects are caused on the synthesized images when the scene elements are not close to the focal plane. This dynamic re-parameterization approach demonstrated that it is possible to reconstruct any scene element without ghosting artifacts. However, artifacts will be apparent for other scene elements for which the disparity is more than one pixel. To reduce the effect of the artifacts they increased the spatial support (aperture) of the reconstruction filter. This had the effect of diffusing the artifacts. However, this introduced new problems of its own. First, scene elements away from the focal plane were excessively blurred, secondly the view-dependent variations in reflectance were lost. In Figure 2.9(c) a dynamic re-parameterization reconstruction using wide-aperture is shown. The focal plane in this case is placed on the yellow toy.

Stewart et al. [SYGMO3] introduced a novel filter for under-sampled light fields by combining the wide-aperture rendering with bandlimited rendering. Their idea is basically to add high-pass data from a wide aperture reconstruction which as we know blurs “out-of-focus” regions reducing their high pass contribution back into a band limited reconstruction. This has the effect of blurring ghosting artifacts (caused by disparity between contributing images) while preserving parts of the image that are in focus. Reconstruction using Stewart’s filter is shown in Figure 2.9(e) clearly it suppresses aliasing compared to Figure 2.9(b) while not softening edges like the bandlimited filter of Figure 2.9(d). To understand the advantages and disadvantages of these filters and their significance for all-focused rendering we will look at them in the frequency domain in the following section.

\(^2\)low pass filtering
2.6 Frequency Domain Analysis of light fields

As seen earlier, the discrete light field is captured using cameras arranged in 2D grid, and the obtained light field is parameterized with \((u, v, s, t)\). Assuming that the cameras are placed with a distance of \(\Delta u\) and \(\Delta v\) and the pixel positions are separated by \(\Delta s\) and \(\Delta t\) the sampled light field \(l_s(u, v, s, t)\) and its Fourier transform \(L_s(\Omega_u, \Omega_v, \Omega_s, \Omega_t)\) are represented by,

\[
l_s(u, v, s, t) = \sum_{n_1, n_2, n_3, n_4 \in \mathbb{Z}} \delta(u - n_1 \Delta u)\delta(v - n_2 \Delta v)\delta(s - n_3 \Delta s)\delta(t - n_4 \Delta t) \tag{2.4}
\]

\[
L_s(\Omega_u, \Omega_v, \Omega_s, \Omega_t) = \sum_{m_1, m_2, m_3, m_4 \in \mathbb{Z}} F(P_4)(\Omega_u - \frac{2\pi m_1}{\Delta u}, \Omega_v - \frac{2\pi m_2}{\Delta v}, \Omega_s - \frac{2\pi m_3}{\Delta s}, \Omega_t - \frac{2\pi m_4}{\Delta t}) \tag{2.5}
\]

where \(\delta\) represents the Dirac delta function and \(F(P_4)\) is the fourier transform of the continuous light field \(P_4\).

The continuous light field function corresponds to the frequency band with all \(m_1, m_2, m_3, m_4\) identical to zero. All other values for these variables corresponds to aliasing sub-bands introduced by sampling. The aliasing effect occurs when these sub-bands overlap with the principal band. For simplicity we will look at the aliasing effect in 2D by taking a slice of the parameterizing planes. The 2D configuration of cameras is shown in Figure 2.7(a).

Chai et al. [CTCS00] did a detailed analysis of the light field spectrum under the assumption of negligible occlusions and non-lambertian surfaces and were able to prove that the signal spectrum of a light field in 2D is restricted to the shadowed area in Figure 2.7(b). The depth of the scene bounded by \(z_{\text{min}}\) and \(z_{\text{max}}\), also bounds the spectral support of the light field as shown in the Figure 2.7(b). In Figure 2.7(b) \(f_s\) denotes the focal length of the cameras, and \(K_{\Omega_s}\) denotes the maximum frequency determined by the complexity of the scene textures or image resolutions. Since the light field is sampled with a constant interval \(\Delta u\), replicas of the original spectrum occur repeatedly as shown in Figure 2.7(b).

They also discussed image rendering using the light field, where they showed that if the original spectrum does not overlap neighboring replications (the anti-aliasing condition), the ideal
2.7. **ALL FOCUSED LIGHT FIELD RENDERING**

As explained in Section 2.5, when attempts are made to synthesize virtual views using the light field, objects near the focal plane are synthesized clearly and appear “in-focus”, while those that are not near the focal plane appear “out-of-focus” and diffused. The depth of the focus depends on the sampling density of the input images and resolutions, and in most cases is too narrow. To solve this problem of range of depth in light field rendering, Isakasen et al. [IMG00] first suggested using multiple focus planes. Similar work was done by other groups [SSY+04, TKN04, KAC04, YMG04]. We will first briefly review the reconstruction

---

### Figure 2.7: Frequency domain Analysis

Box filter shown in Figure 2.7(b) would reconstruct the continuous signal without aliasing artifacts. The slope of the box corresponds to the depth of the chosen focal plane $z_0$. The filter is optimized when $z_0$ is equal to $Z_{opt}$, which is defined as,

$$
\frac{1}{Z_{opt}} = \frac{1}{2} \left( \frac{1}{Z_{max}} + \frac{1}{Z_{min}} \right)
$$

(2.6)

For completeness, the profiles of various filters discussed in Section 2.5 are given in Figure 2.8. The depth of the chosen focal plane is set $Z_{opt}$. In the next section we will describe concepts of “all-in-focus” rendering.

2.7 **All Focused Light Field Rendering**

As explained in Section 2.5, when attempts are made to synthesize virtual views using the light field, objects near the focal plane are synthesized clearly and appear “in-focus”, while those that are not near the focal plane appear “out-of-focus” and diffused. The depth of the focus depends on the sampling density of the input images and resolutions, and in most cases is too narrow. To solve this problem of range of depth in light field rendering, Isakasen et al. [IMG00] first suggested using multiple focus planes. Similar work was done by other groups [SSY+04, TKN04, KAC04, YMG04]. We will first briefly review the reconstruction
CHAPTER 2. PREVIOUS WORK

Figure 2.8: Profiles of different spatial filters in frequency domain

(a) Quadrilinear filtering
(b) Bandlimited filtering
(c) Wideaperture filtering
(d) Stewart filter
2.7. *ALL FOCUSED LIGHT FIELD RENDERING*

Figure 2.9: light field Rendering (a) assuming that the scene element is at infinity (b) with focal plane in front (c) using wide aperture reconstruction and putting the focal plane at yellow toy (d) using bandlimited reconstruction (e) using Stewart's filter (f) using Takahashi's algorithm
strategies adopted by each group.

Isakasen et al. [IMG06] used different focal planes to build a geometric proxy of the scene by pre-processing of sample images, and then used this proxy to synthesize virtual views. Yu et al. [YMG04] also used pre-processing of the sample images to classify them into surface cameras “scams” which are then used to render parts of the scene depending on user input. Both these methods were based on the smoothness in the ray samples used for interpolation. Bayesian approach was adopted by Shum et al. [SSY04] to render “all-in-focus” images. They also used user input extensively to separate the scene elements into layers which are used in rendering an all-in-focus image. Takahashi et al. [TKN04] used spatial consistency of different filters to estimate the depth of different scene elements. In contrast to the above approaches of generating “depth maps” of the scene the focus plane approach of Kubota et al. [KAC04] is based on estimating the Point spread function (PSF) and using Projection onto Convex Sets (POCS) to reconstruct the all-in-focus image.

Since our objective was to perform “all-in-focus” light field rendering on the fly and without preprocessing or user input we limited our research to the ideas of Takahashi et al. [TKN04] and Kubota et al. [KAC04]. Modeling the PSF in light field rendering is not immediate and requires extensive and accurate measurements, moreover it is data-set specific [KAC07]. Hence the spread parameters have to be changed for each different data set. Moreover the algorithm of Takahashi et al. [TKN04] generates a depth image, which is an advantage as it can be plugged into a camera walk algorithm like the one by Choudhury et al. [CSC06]. This will enable generation of fast renderings for a camera walk. Hence we decided to use the Takahashi’s algorithm as the starting point. In the next section we will explore this algorithm in more detail, and in the following sections describe our improvements to it.

2.8 Takahashi’s Algorithm

The algorithm basically consists of two steps. Firstly, for a given viewpoint, images are synthesized by moving the assumed depth of the focal plane used in dynamic-light field rendering.
2.8. **Takahashi’s Algorithm**

This produces a set of images each of which are focused on a particular part of the scene. Then using a focus measure the areas in focus in each image are identified and combined into one in real time. To identify the regions in focus from a synthesized image Takahashi et al. [TKNo4] used a “focal measure” for each pixel using which they were able to identify whether the pixel is “in-focus” or not. We will motivate the choice of Takahashi’s focal measure using an example. In Figure 2.11 we see the results of three different spatial filters (quadrilinear, wide-aperture and camera-skipped).

![Diagram of Takahashi’s algorithm](image)

**Figure 2.10: Takahashi’s algorithm**

---

**Algorithm 1: Takahashi Algorithm**

```plaintext
foreach i = 1 to n do
    Z ← \(Z_{\text{min}} + i \times \frac{Z_{\text{max}} - Z_{\text{min}}}{n}\);
    QuadFilter\(_i\) = QuadrilinearRendering(Z, (x, y));
    WideFilter\(_i\) = WideAperture(Z, (x, y), w);
    diff\(_i\) ← QuadFilter\(_i\) - WideFilter\(_2\);

foreach pixel p in all-focused rendering do
    Do a template matching with square template of size Blk \(\times\) Blk;
    Assign p to \(Z_k\), such that \(\text{diff}_k(p)\) is minimal for \(k \in [1, n]\);
    if \(p \leq \text{threshold}\) then
        \(p = \text{QuadFilter}_k\);
    else
        do a weighted blending from all QuadFilters;

return all-focused rendering
```

3-camera-skipped reconstruction is quadrilinear reconstruction by sub-sampling the light field by 50% before using quadrilinear filter
In each of the three filters the focal plane used to synthesize the virtual view is situated in the foreground. One can note that between the three synthesized images there is a high amount of correlation in the foreground while the background is less correlated. This was the basic premise of Takahashi’s work, and to identify the scene elements that are in focus, they synthesized images using various filters for a viewpoint. Next, they used the difference between the images to identify the sections of the image that are in focus. They further validated this focus measure by analyzing the focus measure in frequency domain.

The algorithm is given in pseudo-code as Takahashi Algorithm. In the pseudo-code, \( l_d \) represents the discrete light field, \( Z_{\text{min}} \) and \( Z_{\text{max}} \) are the minimum and maximum depth, \((x, y)\) is the position were we desire to create all-focused rendering, \( w \) is the width of the aperture filter and \( n \) is the number of focal planes used to generate all focused rendering. In the algorithm we have only used quadrilinear and Wide aperture filter, to keep the pseudo code simple.
Figure 2.11: Takahashi’s algorithm - In the synthesized images one can notice high amount of correlation in the foreground compared to the background. This implies the focal plane is situated close to the foreground.
Chapter 3

Proposed Method

In this chapter we will describe an alternative region based “all-in-focus” rendering using a color based focal measure. We will begin by explaining the redundancy in Takahashi’s algorithm, then motivate our choice of the focus measure and the need for region based approach to “all-in-focus” rendering.

3.1 Motivation

Consider the three filters used by Takahashi’s algorithm in Figure 2.10. Each of these three filters operate by using a set of discrete ray samples for synthesizing virtual views. The discrete ray samples used for both quadrilinear rendering and camera-skipped rendering are present in ray samples used by the wide-aperture filter. This is validated by description of wide-aperture filter in Section 2.4. Hence intuitively, we see an element of redundancy, and this prompted us to attempt “all-in-focus” rendering with just wide aperture filter. To achieve this end, we used an observation by Isaksen [1MG00]. Isaksen classified focal planes as good or bad depending on the smoothness of the color in the samples used to interpolate a pixel in virtual view. For example, in Figure 3.1 plane $u_v$ is classified as good while the other planes are considered bad to synthesize pixel corresponding to ray $r$. This observation can be used as a focus measure.
CHAPTER 3. PROPOSED METHOD

3.2 Color based focus measure

To see how this can be used as a focus measure, we will look into the rendering process shown in Figure 3.2. Here we attempt to synthesize pixel corresponding to ray \( r \) from the ray samples of cameras \( C_n \) and \( C_{n+1} \). If we interpolate using linear techniques, the intensity of ray \( r \) is given by,

\[
I(r) = \frac{C_n C}{C_n C_{n+1}} \times I(b_n) + \frac{C C_{n+1}}{C_n C_{n+1}} \times I(b_{n+1})
\]  

(3.1)

where \( I \) refers to the physical property (intensity or chromacity) of the ray used for interpolation. The result of this interpolation will be blurry because it is obtained by interpolating physically distinct points \( A \) and \( B \) as shown in Figure 3.2. However, if we assume a geometric proxy, and approximate its position using a focal plane \( f \) and then interpolate using techniques of Isaksen we get,

\[
I(r) = \frac{C_n C}{C_n C_{n+1}} \times I(p_n) + \frac{C C_{n+1}}{C_n C_{n+1}} \times I(p_{n+1})
\]  

(3.2)

This interpolation will be more accurate and the synthesized pixel will be in focus, as the rays
3.2. **COLOR BASED FOCUS MEASURE**

used for interpolation correspond to the same physical point on the surface of the object. The choice of \( p_n \) and \( p_{n+1} \) used for interpolation of the virtual ray, depends on the choice of the focal plane. Only for the focal plane that is close to the surface being interpolated, the color of pixels are the same. Hence, if elements in a scene are at different depth, using a particular focal plane will indicate high color similarity in rays used for interpolation, only in the part of the scene that are close to the chosen focal plane. This way we can identify the areas in the scene close to a particular focal plane.

It is reasonable to assume that the spatial support for a virtual ray extends to more cameras, as the same 3D point is projected into multiple cameras. This assumption is also validated by the analysis of Gu et al. [GGC97]. Hence, we decided to use the ray samples in a chosen width (from wide aperture filter) rather than the two closest rays. This increase in spatial support, makes our focus measure more robust. This assumption is of course not valid at boundaries between objects at different depth. However, we cannot handle them without depth information so we will omit that case for now and indicate how to handle it in Section 4.5. It is easy to pick the needed ray samples thanks to the regular structure of the light field. This point will be explained using an example. In Figure 3.3 a 2D point \((a, b)\) is projected into 4 cameras.
arranged in a line, and all the individual ray samples corresponding to the point all lie in the same line as shown in the same Figure 3.3. However, the slope of this line depends on the choice of the focal plane. So given a focal plane and a specific width of the filter, we can pick all the required samples by identifying the line corresponding to the focal plane.

Figure 3.3: The point (a,b) corresponds to 4 different light rays of the light field and they all lie on a straight line in our database. Hence, there is no need to do repeated ray intersection with camera and image planes

Objects in an image are expected to be bigger than a pixel, and if a pixel is in-focus when rendered using focal plane \( f \) then the nearby pixels can also be expected to be in focus when rendered using the same \( f \). Using focus measure on a single location without due consideration of neighborhood tends to produce unstable results. The focus metrics, hence should involve surrounding pixels around each pixel to capture its local characteristics. These neighborhoods (template windows), are usually square windows centered on the pixel being synthesized.

Finally, we are in a position to define the new focus measure \( f_n(x, y) \) for each pixel \((x, y)\) of a synthesized image. The focus measure \( f_e(x, y) \) for each pixel generated using a focal plane at
3.3.  **RENDERING FOR EACH LAYER**

$z_n$, with a spatial support $w$, is given by,

$$A_n(x, y) = \min(C_{r_i} + C_{g_i} + C_{b_i}) \quad i \in w$$  \hspace{1cm} (3.3)

$$B_n(x, y) = \max(C_{r_i} + C_{g_i} + C_{b_i}) \quad i \in w$$  \hspace{1cm} (3.4)

$$\text{sub}_n(x, y) = |A_n(x, y) - B_n(x, y)|$$  \hspace{1cm} (3.5)

$$f_n(x, y) = \sum\sum_{-M<l,M \quad k<2M} \frac{\text{sub}_n(x + k, y + l)}{(2M + 1)^2}$$  \hspace{1cm} (3.6)

The parameters $C_{r_i}, C_{g_i}, C_{b_i}$ corresponds to the amplitudes of the red, green and blue channels of the $i^{th}$ ray used in the interpolation of pixel in position $(x, y)$. $A_n(x, y), B_n(x, y)$ and $\text{sub}_n(x, y)$ are temporary variables used in the calculation of the focus metric $f_n$. $\text{sub}_n(x, y)$ will be low if the pixel is in focus and large if the pixel synthesized at position $(x, y)$ is not in focus. However, $\text{sub}_n(x, y)$ does not take into account the local characteristics. We do that using Equation (3.6) which shows the template matching, employed to factor in the adjacent pixels.

Instead of Equation (3.6) we can also use the cross-correlation between templates applied to $A$ and $B$ as a focus measure. This has been suggested by Liu et al. \[LCM+06\]. This is not only a more costly operation but gives lot more false positives. The biggest advantage of using this focal measures compared to Takahashi is that we just need one filter compared to the three filters used by Takahashi. In the next section, we will describe the wrapping process used to identify $p_u$ and $p_{u+1}$ shown in Figure 3.2 and then motivate the need for region based “all-in-focus”.

### 3.3  Rendering for each Layer

In the previous section we saw how the color similarity between rays used for synthesizing a virtual view can be used as “focus measure”. In this section we will describe how to identify the rays that should be used to synthesize virtual views. We basically use the regular structure of camera array used to capture light field as shown in Figure 3.4. In the figure, we attempt to get the intensity of ray $r$ using the focal plane $F$. To do this we identify the intersection of this ray with $f$ and then we back project this intersection point to existing image samples $C_u$ and $C_{u+1}$.
CHAPTER 3. PROPOSED METHOD

To identify \( p_u \) and \( p_{u+1} \) (the intensity values from existing camera samples corresponding to the pixel being synthesized) the similarity of triangles can be used as shown in Figure 3.4. The triangles \( CM' L' \), \( CML \) are similar, hence we can use property of similar triangles to find \( LM \).

\[
\frac{col/2 - i}{d} = \frac{LM}{Z + d + \text{dist}} \tag{3.7}
\]

\[
LM = (1 + \frac{Z + \text{dist}}{d})(col/2 - i) \tag{3.8}
\]

Next we use \( LM \) to find \( p_u \) as the triangles \( C_u B_u M'' \) and triangle \( C_u B_u M \) are similar.

\[
\frac{p_u - col/2}{B_u M} = \frac{d}{C_u B_u} \tag{3.9}
\]

\[
p_u = (dx - LM) \frac{d}{Z + d} + col/2 \tag{3.10}
\]

Similarly we can find the ray \( p_{u+1} \) captured by camera \( C_{u+1} \)

\[
\frac{col/2 - p_{u+1}}{B_{u+1} M} = \frac{d}{C_{u+1} B_{u+1}} \tag{3.11}
\]

\[
p_{u+1} = col/2 - (dx + LM) \frac{d}{Z + d} \tag{3.12}
\]

Once \( p_u \) and \( p_{u+1} \) are found we can do a weighted interpolation for \( r \) as

\[
I(r) = (dx) p_u + (px - dx) p_{u+1} \tag{3.13}
\]

where \( dx \) is the distance between the camera position used to synthesize novel view and camera \( C_u \) whereas \( px \) is the distance between the two closest cameras used to capture the light field.

3.4 Region based All-focus rendering

In this section we will motivate the need for region based “all-in-focus” rendering. Previously, in section 3.2 we saw that template matching is necessary to improve the robustness of the focus measure. The size and shape of a chosen template implies expectation of local continuity of scene elements and that the neighboring pixels in each respective image are in correspondence.
Figure 3.4: Interpolation using similar triangles
CHAPTER 3. PROPOSED METHOD

It also assumes that the pixels within this template are at similar depth. This assumption is false when the template is used across a surface discontinuity such as the case shown in Figure 3.5(a). In this situation, dissenting pixels outside of the object projection will worsen the overall matching score, even if a true match occurs. This is why square template windows typically fail to identify these discontinuities. Hence, determining correct template for a pixel is very critical and to overcome this problem. One suggested method by Liu et al. [LCM+06] is to use a library of templates. This however is not stable as there is not a structured way to distinguish between a good and a bad template. Instead we suggest a region based template matching as shown in Figure 3.5(b). The assumption is that regions of the image which have same properties like intensity, color and texture are likely to be at the same depth. Hence if we segment the virtual image based on these properties and then apply template mapping should give better results. In the next section, we will give a brief description about graph based image segmentation which is a popular technique to segment images.

(a) Traditional Template matching  
(b) Template matching based on regions

Figure 3.5: In region based template matching only pixels that lie in the same region are used for the smoothing process.
3.5 Normalized cuts for Image segmentation

Image segmentation is the process of splitting an image into regions that have similar properties like color, intensity and texture. Segmentation has numerous applications in computer vision [EMR+02]. Normalized graph cuts, first proposed by Shi and Malik [SM00] is a popular graph based image segmentation. In this technique the pixels in image is first represented as a weighted undirected graph \( G = (V, E) \) of \( V \) vertices and \( E \) edges. The nodes of this graph correspond to pixels of the image. Every pair of nodes \((i, j)\) is connected by an edge, and the weight on each edge \(w(i, j)\) is a function of the similarity between nodes \(i\) and \(j\). The graph \( G = (V, E) \) is then segmented into two disjoint complementary parts \( I_1 \) and \( I_2 \), by removing the edges connecting these two parts. The degree of similarity between these two parts can be computed as the total weight of the edges that have been removed, denoted as \( \text{cut}(I_1, I_2) = \sum_{u \in I_1, t \in I_2} w(u, t) \). The optimal partitioning of a graph is the one that minimises this value. Malik and Shi proposed using fraction of the total edge connections to all the nodes in the graph as cut cost instead of using total edge weight connecting the two partitions, which is is called the normalised cut (\( \text{Ncut} \)) and is defined as follows:

\[
\text{Ncut}(I_1, I_2) = \frac{\text{cut}(I_1, I_2)}{\text{asso}(I_1, V)} + \frac{\text{cut}(I_1, I_2)}{\text{asso}(I_2, V)} \quad (3.14)
\]

![Figure 3.6](image1.jpg)  
![Figure 3.6](image2.jpg)

Figure 3.6: (a) Image and its (b) segmentation based on Normalized cuts

where \( \text{asso}(I_1, V) = \sum_{u \in I_1, t \in V} w(u, t) \). The biggest advantage offered by using \( \text{Ncut} \) is that it prevents cutting out small isolated nodes in a graph. A more detailed analysis can be found in
the original paper \cite{SM00}. Recently a multi-scale normalized cuts approach was suggested by Cour et al. \cite{CBS05}. We used this algorithm to segment images into “super-pixels” as shown in Figure 3.6.

### 3.6 Complete Algorithm

**Algorithm 2: Novel Region-Based Algorithm** \( (d, Z_{\text{min}}, Z_{\text{max}}, (x, y), w, Blk, n, \text{threshold}) \)

```plaintext
foreach i=1 to n do
    Z ← \( Z_{\text{min}} + i \times \frac{Z_{\text{max}} - Z_{\text{min}}}{n} \);
    \( sub_i \) ← color difference in \( w \) for each pixel when rendering using \( \text{WideAperture}(Z,(x,y),w) \);
Generate \( I_{\text{intermediate}} \) using \( Z_{\text{opt}} \);
Segment \( I_{\text{intermediate}} \);
foreach segment seg in \( I_{\text{intermediate}} \) do
    foreach pixel p in seg do
        \( fn \) ← region based template matching using utmost a square template of size \( Blk \times Blk \);
        calculate sum of \( fn \) over all pixels in the segment;
        if \( \text{sum} \leq \text{threshold} \) then
            Assign seg to \( Z_k \), to \( k \in [1,n] \);
            render all pixels in segment using quadrilinear filter
        else
            do a weighted blending from all the \( Z_i \);
return all-focused rendering
```

The complete algorithm is described as follows. The algorithm first generates \( n \) different wide aperture rendering by placing the focal planes uniformly between the maximum and minimum depth. The maximum and minimum depths are assumed to be given. In each of the rendering, it generates pixel based focus measure \( sub_i \) using color-based focus-measure for each pixel according to Equation 3.5. In the next step the algorithm synthesizes image using \( Z_{\text{opt}} \) (Equation 2.6 of Plenoptic sampling theory), and segments this image into regions and forces template matching within each of the individual regions (generating \( f_n \)). The algorithm then assigns each segment to a particular depth by estimating the minimum \( f_n \) over all the pixels that constitute the segment. The segmentation technique utilized is a modified version

\footnote{The choice of \( Z_{\text{opt}} \) is not without motivation. It is the best possible position to render using only one plane according to the Plenoptic theory \cite{CTCS00}.}
3.6. **COMPLETE ALGORITHM**

of normalized cuts. The “all-in-focus” image generation process consists of selecting each region of the segmentation and assigning it to appropriate depth depending on the region based template matching. Figure 3.7 shows the schematic diagram of the proposed multi-focus image fusion method.

The **NOVEL REGION-BASED ALGORITHM** is given in pseudo-code. In this pseudo-code $l_d$ refers to the discrete light field, $Z_{\text{min}}$ minimal depth, $Z_{\text{max}}$ maximum depth, $(x, y)$ is the position were we desire to create all-focused rendering, $w$ is the width of the aperture filter and $n$ is the number of focal planes used to generate all focused rendering and threshold is user defined to remove false positives.

![Figure 3.7: Region based algorithm for all focussed lightfield rendering](image)
Chapter 4

Implementation and Results

In this chapter we will discuss how the different algorithms were implemented and the testing methodology used to verify the correctness of our algorithm.

4.1 Implementation

Implementation of the algorithm was done in Matlab on an Intel Dual Core IBM Thinkpad. The lightfield data-set used for testing the algorithm was downloaded from the “New Light Field Stanford” archive and “MIT light Field” archive. The ‘Toys’ data-set downloaded from MIT was already sub-sampled and no more sub-sampling was done before testing our rendering algorithm. The ‘Jewel’ and ‘Pebbles’ data set from Stanford was sub-sampled by 50% before using it to test our algorithm. More information about the data-set is given in Table 4.1

As part of this project a lightfield rendering GUI was also implemented in C++ using OpenCV. Using the GUI it is possible to simulate various light field rendering effects such as translation, focal depth, filter-size and depth. As we have always used ‘Toys’ data set in previous chapters to describe the rendering effects we will first present the “all in focus” rendering on ‘Toys’ obtained using our algorithm. The all-in-focus rendering and the depth map generated are shown in Figure 4.1. Next we describe the testing methodology used to evaluate our technique.
CHAPTER 4. IMPLEMENTATION AND RESULTS

<table>
<thead>
<tr>
<th></th>
<th>Toys</th>
<th>Jewel</th>
<th>Pebbles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Images</td>
<td>256</td>
<td>289</td>
<td>289</td>
</tr>
<tr>
<td>Pixels per Image</td>
<td>$320 \times 240$</td>
<td>$672 \times 420$</td>
<td>$512 \times 256$</td>
</tr>
<tr>
<td>View Point</td>
<td>$90^\circ \times 90^\circ$</td>
<td>$90^\circ \times 90^\circ$</td>
<td>$90^\circ \times 90^\circ$</td>
</tr>
<tr>
<td>Output Resolution</td>
<td>$320 \times 240$</td>
<td>$672 \times 420$</td>
<td>$512 \times 256$</td>
</tr>
<tr>
<td>Type</td>
<td>real-world</td>
<td>real-world</td>
<td>real-world</td>
</tr>
<tr>
<td>Sub-sampling</td>
<td>none</td>
<td>50%</td>
<td>50%</td>
</tr>
</tbody>
</table>

Table 4.1: Properties of the light field used for testing the rendering algorithm

Figure 4.1: All-focused rendering using our algorithm

(a) Focal plane on foreground
(b) Focal plane on background
(c) All-focused rendering
(d) Estimated Depth
4.2. TESTING METHODOLOGY

To test the accuracy of our approach, the rendered images were compared to the ground truth captured by camera. We used Root Mean Square Error (RMSE) and Number of Significant Error Per Pixel (NSEPP) to quantify the error between the two. RMSE is given by,

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{I} \sum_{j=1}^{J} (R(i, j) - O(i, j))^2}{I \times J}}
\]

where \( R \) is the rendered image, \( O \) is the original image and \( I \times J \) is the dimension of the image. As the name indicates, NSEPP is the number of pixels that differ from the ground truth by a user specified threshold averaged over all the pixels in the image. To show the improvement our algorithm offered over that of Takahasi et al. [TKN04] and Plenoptic Sampling [CTCS00] we synthesized images using these techniques, evaluated their individual RMSE/NSEPP and compared it to our algorithm (in all the cases 5 focal planes were used). However, we will first compare our results with the ground truth.
4.3 Comparision with Ground truth

In this section we will describe the result of comparing our rendering with the ground truth. The light field of ‘Pebbles’ and ‘Jewel’ initially contained $17 \times 17$ images each. We sub-sampled the light field by discarding every alternate row and column. The algorithm from Section 3.6 was then used to generate image at position $(3,3)$ of the pebbles data-set and at position $(10,10)$ for the ‘jewel’ data-set. The rendered images were then compared to the ground truth. The results of this comparison are shown in Figure 4.3 along with the scaled difference images. A histogram comparison of the images converted to grayscale at 8bpp was also done and the results are shown in Figure 4.4. The histogram comparison shows the similarity between the original and our rendering. The maximum of absolute difference between the rendering and ground truth was 49 in the case of ‘Pebbles’ and 91 in the case of ‘Jewel’. Next we will discuss the comparison of our algorithm with other existing ones.

4.4 Comparision with other Rendering Algorithms

Both Takahashi’s algorithm [TKNo4] and rendering using optimal plane of plenoptic sampling theory [CTCS00] were implemented in Matlab and compared to our algorithm using the error metric of Section 4.2. The results of the comparison are shown in the Table 4.2-Table 4.5. The threshold used for NSEPP is 10. In all the tested cases our algorithm outperformed other methods. Visual comparison of our algorithm to Takahashi’s algorithm is shown in Figure 4.5. By comparing our rendering algorithm with the ground truth, we can also see the advantages our algorithm offers over Takahashi’s algorithm [TKNo4].

<table>
<thead>
<tr>
<th></th>
<th>Proposed algorithm</th>
<th>Takahashi’s algorithm</th>
<th>Optimal Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position(3,3)</td>
<td>550.49</td>
<td>584</td>
<td>772.4</td>
</tr>
<tr>
<td>Position(5,3)</td>
<td>427.1</td>
<td>501.2</td>
<td>749.6</td>
</tr>
<tr>
<td>Position(7,3)</td>
<td>770.5</td>
<td>810.3</td>
<td>902.4</td>
</tr>
<tr>
<td>Position(10,10)</td>
<td>661.3</td>
<td>710.2</td>
<td>901.4</td>
</tr>
</tbody>
</table>

Table 4.2: Error analysis using RMSE on ‘pebbles’ light field
Figure 4.3: Comparison of results obtained using our algorithm with the ground truth.
Figure 4.4: Histogram comparison of rendering to ground truth in 8bpp gray scale
4.4. COMPARISON WITH OTHER RENDERING ALGORITHMS

Figure 4.5: Comparison to existing algorithms-It is clearly seen that edges are better preserved using our algorithm.
CHAPTER 4. IMPLEMENTATION AND RESULTS

Proposed algorithm | Takahashi’s algorithm | Optimal Plane
--- | --- | ---
Position(8,8) | 1575.0 | 2048.3 | 3350.8
Position(10,8) | 2061.6 | 2304.5 | 2761.7
Position(6,10) | 1907.6 | 2107.8 | 2674.8
Position(10,10) | 2117.8 | 2247.8 | 2781.7

Table 4.3: Error analysis using RMSE on ‘Jewel’ light field

| Position(8,8) | Proposed algorithm | Takahashi’s algorithm | Optimal Plane |
|---|---|---|
| Position(10,8) | 0.0307 | 0.0366 | 0.1224 |
| Position(6,10) | 0.0344 | 0.0395 | 0.0795 |
| Position(10,10) | 0.0296 | 0.0357 | 0.0857 |

Table 4.4: Error analysis using NSEPP on ‘Jewel’ light field

4.5 Discussions

From the above examples it is clear that the region based approach is an effective way to render under-sampled light fields. However, in the strength of the region based approach lies its weakness as well. Segmentation is an important step in our rendering methodology and that is usually a time consuming process. The approximation algorithm used for image segmentation runs in $O(mn)$ where $n$ is the number of pixels and $m$ is the number of steps Lanczos algorithm takes to converge [SMoo]. Hence the time taken by our algorithm increases rapidly as compared to Takahashi’s algorithm when the resolution of light field images increases. We did not analyze the running time in Matlab as it is optimized to matrix operations, hence Takahashi’s algorithm will be much faster than our algorithm. This was indeed observed in the testing phase when for generating all in focused rendering using 80 segments the time taken by our algorithm was nearly ten times that of Takahashi’s algorithm. However, we believe when rendering in C++ this gap will narrow down.

The reason for our gains in the quality of rendering is the choice of color as focal measure. In designing our focus measure described in Section 3.2 we implicitly make a couple of assumptions:

- any point in the scene that is visible from the novel viewpoint is also visible in the nearby
4.5. DISCUSSIONS

Proposed algorithm | Takahashi’s algorithm | Optimal Plane
---|---|---
Position(3,3) | 0.022 | 0.025 | 0.055
Position(5,3) | 0.013 | 0.016 | 0.051
Position(7,3) | 0.059 | 0.065 | 0.09
Position(10,10) | 0.047 | 0.056 | 0.084

Table 4.5: Error analysis using NSEPP on ‘pebbles’ light field

four cameras;

- the projections of the same physical point in the scene should have a higher level of color similarity than the projections from different physical points.

They are not always valid, the first one doesn’t hold in occluded areas and the second one is not valid for non-lambertian surfaces. Nevertheless, this metric is quite robust, as is seen from experiments with real images. This indicates that our approach is not sensitive to the violation of the assumptions since it is very likely that the overall dissimilarity of the physical point is still the smallest even though the point is occluded in one or two nearby cameras. However, occluded points can still be handled in our framework once the depth information is obtained. Occluded points always lie further away than the occluding points which lie on a plane closer to the camera than occluded points. As soon as a foreground point is detected, all pixels corresponding to it are ignored for further calculations. Therefore, occlusions can be modeled correctly however we did not include it in our reconstruction method.
In addition to the proposed region based approach to “all-in-focus” rendering that we described in Chapter 3, we explored different options in our quest to improve the quality of light field rendering. In these sections we will briefly describe our approaches and the roadblocks encountered in each of these approaches. One of the first techniques explored to circumvent the problem of aliasing in light field rendering was filtered blending. It was introduced by Eismann and Magnor [EM07] as a viewpoint dependent anisotropic reconstruction filter. They used a band limited filter, the cutoff frequency of which depends on the depth uncertainty at a particular viewpoint. This technique indeed reduces the dependency on geometry but attempts to synthesize using a under-sampled lightfield data set still results in blurring. Next we attempted a spatial frequency based “Depth from focus” approach.

5.1 Spatial Frequency based all in focus

Depth from defocus offers a direct solution to fast and dense range estimation in conventional photography [FS02]. It is a computationally efficient method that can distinguish between the regions that are in focus based on spatial frequency and thereby obtain structure of the scene. In images captured using physical camera, the out-of-focus areas are regions without high frequency detail, and the in-focus areas are regions with more spatial frequency. The idea behind “all focus” rendering in conventional photography is that measure regions in image
with high frequency and then combine these regions into one “all in focus” image.

Spatial frequency is usually measured using the Tenengrad operator or Discrete cosine Transform (DCT). The Tenengrad operator maximizes,

$$\nabla I(x, y) = \sqrt{\nabla I_x^2 + \nabla I_y^2}$$

(5.1)

where $\nabla I_x$ and $\nabla I_y$ are obtained using Sobel masks. The other technique is based on dividing images into $n \times n$ pixel blocks and then apply Discrete Cosine Transform (DCT) to individual blocks. The spatial frequency component in the block can be measure using the DCT coefficients $c_{ij}$ and the $m_{ij}$ magnitude of frequency for a coefficient at position $(i, j)$ is given by $m_{ij} = \sqrt{i^2 + j^2}$. The focus measure $f$ can then be defined by,

$$f = \sum_{0 \leq i < n} \sum_{0 \leq j < n} c_{ij} m_{ij}$$

(5.2)

Using these focus measures the all in focus image can be constructed. This approach is directly unsuitable to our problem as the out of focus regions has both low frequency and high frequency information. However the Stewart filter discussed in Section 2.5 suppresses high frequency information in “out of focus” areas while maintaining the same in low frequency region. We investigated whether we can use this filter along with “depth from focus” techniques to synthesize all in focus image. The images synthesized using this method indeed attested to the strength of our reasoning. This area, can still be explored to improve the quality of rendering.

### 5.2 Wavelet based Image Fusion

*Image fusion* can offer some possible solutions to our problem with light field rendering. According to *Wikipedia* Image Fusion “is the process of combining relevant information from two or more images into a single image”. Image fusion has numerous application in medical imaging and remote sensing and can be done either in transform domain (Principle Component Analysis (PCA), Discrete Cosine Transform (DCT) or Wavelets) or in the pixel domain. Our problem can also be looked at as an Image fusion problem, where our aim is to combine the images synthesized with different parts of scene being in focus into one “all-in-focus” image.
5.3. BAYESIAN ENERGY MINIMIZATION

The other technique we explore was bayesian energy minimization. Many Computer Vision problems can be naturally formulated in terms of energy minimization \cite{BVZ01}. Geman and Geman \cite{GG84} proposed a bayesian interpretation of many of these energy functions, and also proposed a discontinuity-preserving energy function based on Markov Random Fields (MRF’s).
CHAPTER 5. ALTERNATIVE APPROACHES

Figure 5.2: Block diagram of generic wavelet based image fusion (Reproduced from [Stao8])

Figure 5.3: On Comparing “Wavelet based all focus” to “Region based method” we see more ghosting and a loss of chrominance information in the wavelet methodology
These energy functions can be minimized effectively using Graph cuts [BVZ01]. An effective and concise overview of graph cuts and the problems it is applicable to can be found in the work of Kolmogorov and Zabih [KZ04]. To give a brief overview here, graph cuts is basically an energy minimization framework in a spatially coherent space. Given a set of labels and an input image each pixel \( f \) of the image is associated with a label \( \theta \). The total energy cost over an image whose pixels are assigned particular labels is defined by,

\[
E(f, \theta) = Q(f, \theta) + \sum_{p,q \in N} V(f_p, f_q)
\]  

(5.3)

where \( Q \) is the cost of assigning the label \( \theta \) to \( f \) and \( V \) is the cost of different labels to pixels that fall in same neighborhood \( N \). The goal is to decrease the overall penalty shown in Equation 5.3. To solve this problem effectively a max flow approximation algorithm is applied to the graph representing the image. Our approach was to segment a synthesized image into foreground and background using labels. The focus measure discussed in Section 3.2 was used as the pixel labeling cost. If the variation in color of rays used for interpolation is large then the energy associated with the labeling is large. This technique however did not work, the main reason being the assignment cost \( Q \) was not accurate enough as shown in the Figure 5.4.

Figure 5.4: Bayesian labeling - The labeling cost is not accurate as can be seen from these two figures. Ideally for \( Z_{\text{min}} \) the background should be approximately white and for \( Z_{\text{max}} \) the foreground should be approximately white.
Chapter 6

Conclusions and future work

Light field rendering is a simple and effective image rendering technique which has potential applications in many fields including the upcoming 3DTV. One of the severe limitations of light field rendering is its inability to handle scenes with various depths. In such cases, only a part of the scene at a chosen depth can be rendered accurately and the other parts of the scene at a different depth cannot be rendered without aliasing. To handle such cases, the only solution offered by theory was to increase the sampling density of the camera array. This is impossible in a number of cases as the size of data explodes. To counter this and render using sub-sampled light fields is needed.

We have presented a novel region-based all-in-focus light field rendering algorithm that reduces artifacts when rendering using light fields. We accomplish this without increasing the camera-plane sampling density or preprocessing. By reducing the number of necessary camera-plane samples, this method allows reconstruction with fewer images, thereby reducing storage space for the light field and decreasing run-time memory requirements. The spatial domain analysis, gives clear motivation for the algorithm and the experimental results validate our approach. The experimental results also show clear gains over existing techniques for all-in-focus rendering. Using standard metrics like RMSE and NSEPP we proved that the gain is atleast 8% over existing light field rendering techniques. Finally, we discuss the potential of applying image fusion techniques to light field rendering especially techniques like depth-from-focus.
The biggest challenges faced while carrying out the project was getting acclimatized to the concept of light field rendering. Though it is a relatively simple concept background information was quite sparse. Another huge problem was obtaining light field data to test algorithms as it is difficult to build camera arrays. At the start of the project we had planned on extending light field to dynamic scenes, however because of constraints in capturing dynamic light field, and the suitability of light field rendering to static scenes. Hence, we had to alter the goal and search for a suitable problem. This also proved to be quite a challenge. In the next section the possibilities for further work is discussed.

6.1 Future work

A limitation of this algorithm which is common to all shape-from-stereo algorithms, is the matching errors due to color ambiguity. However, the errors due to incorrectly estimated depth are generally no worse than the ghosting artifacts when rendering directly from an under-sampled light field [CTCS03]. Further research directions include a possible Bayesian-based focus metric against a color-based metric. Our algorithm does not explicitly deal with occlusions or specular surfaces. It would be worthwhile to extend the algorithm to handle occlusions and specularity. The suitability of this algorithm to light field camera walks can also be explored. Camera walks are trivially possible by caching the required disparity information at nodal points [CSC06]. However it is difficult to generate disparity images in a fast and efficient way. We also explored in Section 5.1 the possibility of using spatial frequency to generate an all-focused light field rendering. This focus metric can be improved to provide an alternative means of all-focused rendering. Designing and optimizing a hardware implementation of the algorithm is also an interesting problem. These can be the topics for future research.
Bibliography


[FMR+02] Jordi Freixenet, Xavier Munoz D. Raba, Joan Marti, and Xavier Cufi, Yet another survey on image segmentation: Region and boundary information integration, Proceedings
of the 7th European Conference on Computer Vision (ECCV ’02), 2002, pp. 408–422.


