Photo-realistic Rendering of Real-world Objects Based on Insufficient Measurement

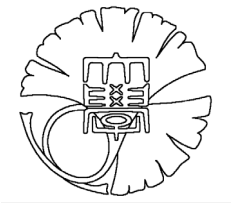
不十分な計測情報に基づく実物体の写実的な画像合成

by

Shuntaro Yamazaki

A Doctoral Dissertation

Submitted to the Graduate School of
The University of Tokyo



for the Degree of
Doctor of Computer Science
in Information Science and Technology

on December 16, 2003

Supervisor:

Katsushi Ikeuchi

Committee:

Tomoyuki Nishita (chair)

Reiji Suda

Takeo Igarashi

Takashi Kanai

Takeshi Naemura

The dissertation of Shuntaro Yamazaki is approved:

Tomoyuki Nishita Feb. 17, 2004

chair date

Reiko Kitamura Feb. 17. 2004

date

Takeo Igarashi Feb 17. 2004

date

Takashi Kanai Feb 17. 2004

date

Takeshi Naemura Feb. 17. 2004

date

The University of Tokyo

2004

Abstract

Displaying real objects in a virtual space by synthesizing photo-realistic virtual views based on the measured data sets has been studied for a long time in the computer vision and computer graphics communities. According to the analysis of the amount of geometric and photometric information required to render photo-realistic images, a variety of existing methods can be classified into two representative groups: model-based rendering and image-based rendering. In practice, however, it is often unfeasible to take sufficient measurements to achieve photo-reality due to the limitation on sensor accuracy, storage size and other practical restrictions. If the amount of the obtained data set is less than that of a theoretical lower bound, it is impossible to achieve photo-realistic rendering.

In this dissertation, we present three photo-realistic rendering methods designed to solve the problem described above. These methods are mutually complementary; The first and second methods take charge of, respectively, automatic and semi-automatic view synthesis from a sparse set of two-dimensional images. The third method was developed with the goal of rendering a variety of objects, including intricately-shaped objects, using possibly inaccurate geometric models and view-images.

The first method, *Unconstrained View-interpolation*, based on image morphing, enables a visually plausible synthesis of virtual three-dimensional views from only two-dimensional images without any other knowledge of the object wished to be rendered. Given two images, the intermediate view-image is synthesized by deforming and blending the given images so that the intermediate image is kept natural-looking. In order to achieve such a deformation, a fully automatic method for unconstrained image matching has been developed. In this method, the features of the images are first extracted by a filter bank composed of filters similar to those used in the human vision system. The mapping between the images is then sought by minimizing an objective function, taking into account the convexity of the mapping.

While the many computer vision algorithms, including the first method, do not neces-

sarily work well for the real data set, the second method, *Pop-up Light Field*, can achieve photo-realistic view synthesis even when the set of given images is too sparse. The key to our approach is an effective interface that enables the user to interactively improve the rendering quality. The process of rendering and modifying the object is repeated iteratively and interactively until its quality satisfies the user. The object wished to be rendered is modeled as a set of coherent layers, and then rendered by using the layered unstructured lumigraph rendering algorithm. In order to achieve anti-aliased rendering, we also propose a novel method of alpha matting.

The third method, *Microfacet Billboarding*, is a generic framework for modeling and rendering real-world scenes based on a measured data set. In contrast to the first two methods, this method takes both geometric model and view-images. First, the system determines the maximum resolution of the geometric model according to the consistency within the measured data set. The details of the object are then synthesized using view-dependent geometry and view-dependent texture mapping with alpha blending. The alpha matte for each view-image is estimated before rendering to overcome the limitation of sensor resolution when the object has intricate geometry like fur or pine needles. When the amount of geometry is sufficient, this method can be regarded as one of the methods of model-based rendering while, when the geometry is totally unreliable, the method converges to the method of image-based rendering.

要旨

実世界の風景や物体を計測した情報を利用して，仮想空間で物体の画像を写実的に合成する手法は，コンピュータグラフィクスやコンピュータビジョンの分野で盛んに研究されている．実物体の画像を写実的に合成するためには，対象物の三次元形状 (幾何情報) とテクスチャ(光学情報) が一定量以上必要であることが理論的に示されており，十分な量の幾何情報に基づき表示を行う Model-Based Rendering，十分な量のテクスチャ情報に基づき表示を行う Image-Based Rendering の手法が数多く提案されてきた．ところが，こうした手法を実際に適用する際には，計測装置の精度の制限や，計測コストや容量等の現実的な制約から，必ずしも十分な計測情報が利用できるとは限らない．従って，計測が困難な対象物や環境では写実的な再現を行うことができず，手法の適用範囲を制限してきた．

本論文では，このような問題を克服するために，三つの相補的な手法を提案する．

一つ目の方法である自動モーフィングに基づく視点画像補間法では，描画対象の幾何情報が全く利用できない場合に，形状の再構築を行うことなく，画像のモーフィング処理によって自由視点画像を合成する．まず，事前知識なしに，与えられた 2 次元画像間の密な対応を推定し，この対応点に基づき画像を変形・混合することによって自由視点画像を生成する．画像を対応付けするための特徴量の抽出手法として，人間の初期視覚で用いられているものと類似した，回転偏微分ガウスフィルタを用いたフィルタバンクを提案する．また，このようにして得られた画像特徴の対応付けを行うために，局所解を避けながら目的関数を最小化する手法を提案する．提案法では，透明物体やテクスチャのない均質な物体に対しても，写実的な視点画像補間が可能である．

二つ目の手法である Pop-up Light Field 法では，一つ目の手法と同様に幾何形状が利用できず, さらに入力画像から付加的な情報を得ることが難しい場合，利用者による対話的な幾何モデル生成を用いて, 写実的な表示を実現する．提案法では，取得した計測情報を基に Light Field 表示を行い，現実感を損なっている部分に対して層状構造 (Coherent Layers) 化処理を行うことによって描画画質を改善する．この処理

は利用者が対話的に行うことが可能で，そのためのユーザーインターフェースを提案する．また，Layer 化によって生じる，対象物の輪郭線付近におけるエイリアスを防ぐために，視点画像間で一貫性のあるアンチエイリアシングを実現する Coherence Matting 法を提案する．提案法では，Layer 化を繰り返すことにより，利用者の目的に応じた写実性を持つ自由視点画像の合成が可能である．

三つ目の手法である Microfacet Billboarding 法では，光学情報に加えて粗い幾何情報も利用し，既存の計測装置では取得が困難な微細な形状を持つ対象物に対しても写実的な描画を実現する．まず，得られた幾何情報の信頼度を計算し，精度に応じて対象物の形状を，大域的，局所的 (視点依存) モデルの組み合わせとして表現する．大域的形状に基づき視点依存の微小面集合を用いて対象物の形状を近似表現し，視点依存距離マップ，視点依存テクスチャマップを用いて詳細な描画を行う．また，テクスチャ画像の透過度推定を行うことにより，光学センサの解像度以下の形状を扱うことができ，本手法は，樹木や毛髪といった形状計測が困難な物体に対しても適用可能である．また，この手法は，計測結果の幾何情報の解像度が高いとき Model-Based Rendering 法に近づき，幾何情報の精度が低いとき Image-Based Rendering 法になるという特徴を持つ．

Acknowledgements

I would first like to thank my advisor, Katsushi Ikeuchi, for taking me on as his student and allowing me to pursue my research interests, and for his support and encouragement during the course of this work. He gave me the freedom to develop various new ideas and a great deal of advice when needed.

I would also like to express my gratitude to three advisors outside the University of Tokyo. First I would like to thank my ex-advisor, Yoshihisa Shinagawa, of the University of Illinois at Urbana-Champaign for his guidance during my master course in science. The research on unconstrained image matching presented in this dissertation is based on the work accomplished under his guidance. I would also like to thank Kiwamu Kase, my co-advisor in Riken. I appreciate his valuable and significant comments on my research and life. The research on hardware-accelerated rendering presented in this dissertation are based on the techniques developed with him. Lastly I would like to express my deepest gratitude to Harry Shum, my mentor during my internship at Microsoft Research Asia in Beijing. The six months spent in Beijing were among the most exciting and pleasant days in my doctoral course. The research on an interactive system for image-based modeling and rendering was pursued under his guidance.

There are many people without whom this work would not have been possible. Ryusuke Sagawa and Hiroshi Kawasaki, my colleagues in the University of Tokyo Computer Vision Lab (CVL), worked with me in discussing new ideas, developing software and writing conference papers. The work with them was really enjoyable and exciting. My thanks also go to Kensuke Habuka and Jun Kamoshima in Shinagawa Lab, with whom I established the basic grounding in computer graphics and computer vision. I would also like to thank Jian Sun for enjoyable and successful joint research during the course of my internship at Microsoft Research Asia.

I would also like to thank the faculty of the CVL at the University of Tokyo. Yoichi Sato, Masataka Kagesawa, Ko Nishino, Ogawara Koichi, Ryo Kurazume, Atsushi Nakazawa,

Auethavekiat Supatana and Kazuhide Hasegawa deserve special thanks for discussing my research and for giving me valuable suggestions and advice. I would also like to thank Yoshinori Teshima, Tomoyuki Fujimori and all other staff of the Volume CAD group at Riken for their support and friendship. I owe, without question, the success of my dissertation to their frequent help.

I was fortunate to have many great people to work with during my stay at the University of Tokyo. My fellow graduate students, Jun Takamatsu and Robby Tan, have helped me to bear the strain of the difficult requirements for graduation. I would also like to express my appreciation to other graduate students in my office, Yasuhiko Uehara and Yuichiro Hirota, for taking my mind off my worries through trivial conversations and playing minesweeper.

I am also grateful to Marie Elm for taking the time to proofread this dissertation. She has been kind to spare her time for correcting my writing and giving appropriate advice about how to improve my English.

Lastly, I would like to thank my family and friends for their support during my nine years of study at the University of Tokyo. Most of all, the patience, support, and advice of my parents was of great value to me.

Shuntaro Yamazaki

March 10, 2004

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background

The synthesis of realistic virtual views remains one of the central research topics in computer graphics. The range of applications encompasses various fields, including: interactive catalogues for e-commerce, visual interfaces for long-distance communications and integrated environments of reality and virtual reality as well as visual effects commonly used in film production. Thanks to recent improvements in both microprocessor power and networking environments, it has become possible and necessary for us to exploit the benefits of realistic rendering even in daily life.

The ultimate goal of the research on realistic rendering is to display a scene on a screen so that it appears as if the object actually exists behind the screen. This description, however, is somewhat ambiguous to provide a quality measure for synthesized images. Instead, in computer graphics and computer vision communities, considerable effort has been put forth to synthesize the virtual view of real or imaginary scenes so that they look like the real photographs. When a synthetic image is indistinguishable from the real photograph, it is referred as *photo-realistic*. We also persue the photo-realistic rendering of real-world scenes in this dissertation.

The strategies for synthesizing photo-realistic views can be classified broadly into two groups according to how a scene that the user wishes to render is modeled. One is the approach in which skilled designers manually create the models of the objects that they wished to render. Then the scene is rendered applying certain optical simulation to the model. This approach is essential in rendering imaginary scenes; hence it is commonly adopted in such fields as film production. The other approach is one in which the designers

attempt to synthesize realistic views based on the models obtained by observing the real world.

As is reviewed in Section 1.2, the recent trend of the research on modeling scenes is to make the best of the latter approach, referred as *modeling from reality* [IS01], together with the former framework, if necessary. Owing to the rapid progress in hardware capabilities and software techniques of modeling scenes, it has become easy to bring real-world scenes into the digital world. Considering that one of the most important issues in the process of image synthesis is how to model the scene without requiring time-consuming manual operations by the user, it seems quite reasonable to base the primary method of modeling on the observation of real-world scenes.

In this dissertation, three different techniques for view-image synthesis based on measured data sets are proposed. All methods are designed so that the synthesized view-images look photo-realistic even when the data sets are sparsely-sampled.

The motivation of this research is that, in practice, it is difficult to take sufficient measurement to achieve photo-realistic rendering synthesis owing to some fundamental or practical reasons concerning measurement. For instance, it often happens that the opportunity for measuring the scene wished to be rendered occurs only once. When the amount of data obtained at the first measurement does not suffice for the purpose of photo-realistic image synthesis, users have to try their best to render the scene using the under-sampled data set by some means or other. The data deficiency may also arise from the limitation of optic sensors. It is extremely difficult, if not impossible, to measure the accurate shape of objects whose surfaces have elaborate details such as hairs or pine needles even with a state-of-the-art method of robust geometric modeling [HSIW96, CL96, WSI98]. When the scale of these shapes is smaller than the resolution of optic sensors, the obtained geometry may suffer from considerable noise.

In the remainder of this chapter, after reviewing extensive research on realistic image synthesis based on the measurement of real-world scenes, we briefly preview the three methods proposed in this dissertation. In Chapters 2 and 3, we present, respectively, automatic and semi-automatic methods for synthesizing photo-realistic virtual views without explicit geometric modeling. The first method is the *unconstrained view-interpolation*, which is based on the automatic matching of reference images. The second method, which we refer to as the *Pop-up Light Field*, is a system that enables the users to model and render the scene interactively from sparse light fields without any geometrical information. In Chapter 4, we then describe the method, which we call *Microfacet Billboarding*, for rendering scenes based on a set of noisy range images and sparse texture images. Finally,

in Chapter 5, we summarize the contributions of this thesis and suggest some possible directions for future extension of this work.

1.2 Previous Work

In the framework of modeling from reality, all phenomena occurring in a scene are captured through a measured data set; hence, novel synthetic views of the scene are generated directly or indirectly from the data set. When the virtual view of a scene is synthesized based on some information on the geometric model of the scene, the techniques of rendering refer to either the optical simulation on the geometric model or the warping of reference view-images with the support of such associated knowledge as pixel correspondences, depth maps and shapes of the object. On the other hand, if no geometric information on the object is available, the synthetic views have to be generated directly from input images by employing the algorithms of interpolation. Thus, the model required to render scenes consists of two elements: *geometric information* and *photometric information*. The geometric information on an object is equivalent to its shape, which can be obtained through the measurement for example using range finders and stereo reconstruction. The photometric information is related to the object appearance, which is usually acquired as a set of view-images of the object.

Based on the observation, existing research on the technique of synthesizing virtual views can be broadly classified into two approaches: model-based and image-based. The model-based approach represents the traditional way to generate virtual views of an object or scene. This approach is referred as *Model-Based Rendering* (MBR) because it usually relies on a geometric and photometric model of the object or scene wished to be rendered. The image-based approach, referred as *Image-Based Rendering* (IBR), represents, instead, an alternative to model-based rendering, and it synthesizes virtual views relying on real images taken as reference in place of the model of the scene. In order to produce novel views, reference images are usually interpolated or re-projected from source to target image. A detailed review of existing methods for model-based and image-based rendering is presented in Sections 1.2.1 and 1.2.2 respectively.

As for the relationship between geometric and photometric information, Chai et al. [CTCS00] established the *plenoptic sampling* theory which clarifies that there exists a lower bound for the sampling rate necessary to synthesize virtual views without visual artifacts. They applied spectral analysis to the samples of geometric and photometric information, and

then formulated the problem of realistic image synthesis as the restoration of the original spectra from the samples. According to the theory, when the total amount of data given as input is above the theoretical lower bound, a virtual view of the scene can be synthesized by appropriately interpolating input data using either model-based or image-based methods mentioned above. However, when the total amount of obtained information is less than the lower bound, the synthetic view will suffer from visual artifacts caused by aliasing.

One of the solutions to the problem of realistic rendering based on an under-sampled data set is to estimate appropriate correspondence between given view-images and to apply *image morphing* so that the synthetic view between reference images looks visually plausible, if not physically valid. The method proposed in Chapter 2 is developed in order to overcome the difficulty of photo-realistic rendering due to data sparseness by warping and blending measured images. This approach is different from both model-based and image-based rendering in the sense that the method directly solves the problem in the deficiency in given models by the techniques of two-dimensional image processing without consideration of three-dimensional transformation. Therefore, the related work is reviewed separately in Section 1.2.3.

Another solution is to extract some information from the given data set up to the amount that suffices for the purpose of realistic rendering. For instance, when we have more than one view-image of the scene we wish to render, it may be possible to estimate the scene geometry using such methods as stereo reconstruction [Fau93]. It is, however, often the case that even state-of-the-art automatic stereo algorithms are inadequate for producing sufficiently accurate depth information for realistic rendering. When it is difficult to automatically extract the additional information, one possible alternative is to incorporate *interactive modeling* by the user. We propose an interactive system for photo-realistic rendering from a few images in Chapter 3. The existing methods related to the proposed method are reviewed Section 1.2.4.

The relationship of the prior techniques reviewed in the following sections is illustrated in Figure 1.2 to provide the reader a hint for better understanding; however, their positions may be somewhat inaccurate since it is difficult to evaluate the precise amount of information used in their method. The vertical and horizontal axes indicate, respectively, the accuracy of geometry and the amount of texture images used for rendering. The curve in the figure indicates the theoretical lower bound that is derived in the plenoptic sampling theory. As mentioned above, our proposing methods take on the data set possibly sparser than the lower bound.

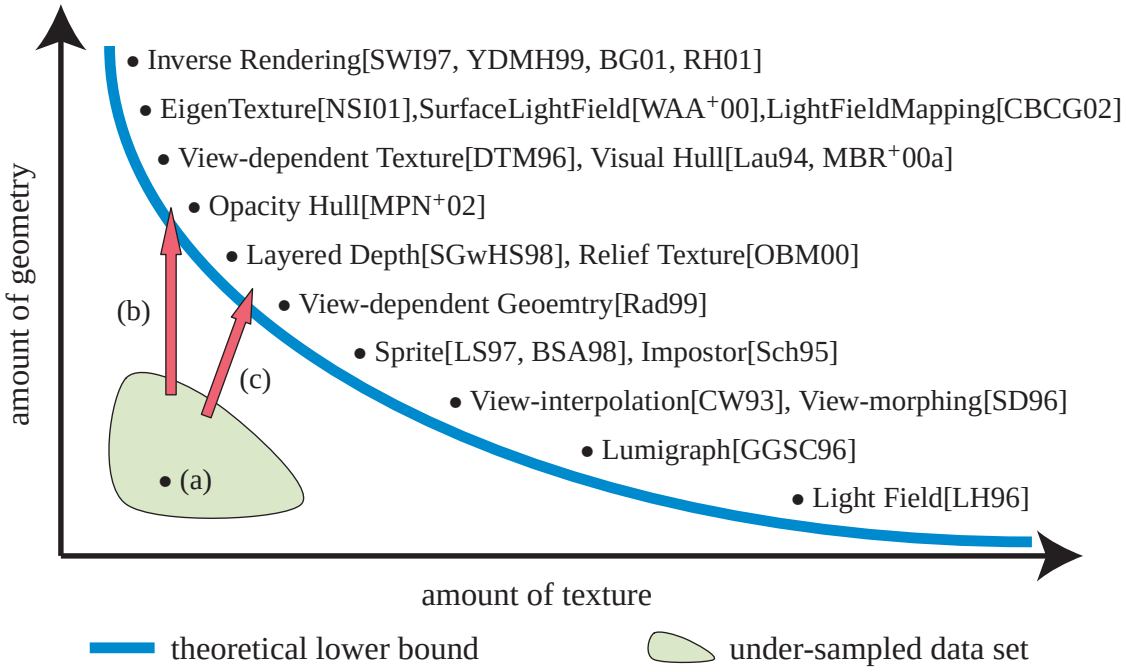


Figure 1.1: Related work arranged in a two-dimensional graph according to the amount of geometric and photometric information required in their rendering. In order to deal with the data set whose amount is possibly less than a theoretical lower bound, we propose three methods of rendering: (a) *Unconstrained view-interpolation* (in Chapter 2) based on the technique of automatic image morphing of two uncalibrated images, (b) *Pop-up Light Field* system (in Chapter 3) for rendering sparse light fields with the aid of the user's interactive modeling, and (c) *Microfacet Billboarding* method (in Chapter 4) for rendering intricately-shaped objects using view-dependent modeling and alpha matting.

1.2.1 Model-based Rendering

Model-based rendering recovers the geometric and photometric models of the object, and then renders the scene from desired virtual viewpoints based on the obtained model. Once the complete model of the scene is created, it is relatively simple to synthesize the views; hence, considerable effort in the research on model-based rendering has been devoted to creating accurate models of the scenes.

The first step in model-based rendering is to create the geometric model of the object wished to be rendered. The research on the techniques for construction of geometric models has a long history in computer vision. Various methods based on active optical sensors, such as structured light system [DT96], photometric stereo system [Woo80, Woo89] and laser range finder, or passive algorithms such as stereo reconstruction [Fau93], structure from motion [TK92, DSTT00] and volumetric reconstruction [FK98, KS99] have been proposed. Among them, the method based on the integration of range images obtained by laser range finders is receiving increasing attention because of its high accuracy and robustness. With the improvement in the algorithms to cope with the problems inherent to laser scanning, for example presence of noise [WSI98, NI02] and occlusion [CLF02, DMGL02], the range of possible applications [LPC+00, ISN+00] has been extended.

The photometric model of the object is then created based on the obtained geometric model and a set of view-images taken from usually various viewpoints by digital cameras. The approaches to photometric modeling are broadly classified into two groups; image-based representation and inverse rendering. In the image-based approach, ideally all possible views of the object wished to be rendered are captured and stored in a database. To synthesize the views of the object, the most appropriate view is selected from the database and displayed. Owing to the knowledge of accurate geometry to which the appearance images are mapped, a certain correlation between the samples observed at the same point on the surface can be exploited in order to compress the size of the database [NSI99], interpolate samples [NSI01, WAA+00] and factorize the database for hardware rendering [CBCG02]. In the inverse rendering approach, the bi-directional reflection distribution function (BRDF) on the object surface is directly estimated from observed images. Since this estimation problem is ill-posed, it is often assumed that the optical phenomena occurring in the scene follow some physically-based reflectance models [TS67, CT81]; then, the reflectance parameters in the model are estimated by fitting pixel values in observed images [SWI97], solving the inverse radiosity problem [YDMH99] or iteratively refining

reflectance paramters [BG01]. Ramamoorthi and Hanrahan formulated the problem of inverse rendering as the reconstruction of an original signal from the observed samples and provided the framework to solve many problems in inverse rendering [RH01].

Once the model of a scene has been created, it is a straightforward task to render the scene by such methods as *polygon-based rendering*. When the reflectance parameters on the object surface are known, a virtual view of the object can be synthesized by drawing polygons with the colors computed according to some reflectance model. When the photometric model is image-based, the scene is rendered by using polygons with view-dependent texture mapping [DTM96], which generates the texture images by interpolating reference textures according to the angle formed by the viewing direction and the camera direction of the images.

When the scale of the scenes to be rendered is so large that the resolution of the geometric model can be higher than that of the rendering screen, *point-based rendering* is used to accelerate the rendering speed and to save storage space. It is often the case that a single model is composed of over 10 million points on the object surface when the scene is modeled with the aid of laser range finders. In such a case, it is good idea to splat the point onto the screen instead of polygons. The use of points as the primitive to render continuous surfaces was first proposed by Levoy and Whitted [LW85]. The fundamental problem in point-based rendering is detecting and closing any holes so as to correctly reconstruct the surface; this problem was solved by the introduction of the hierarchical Z-buffer by Grossman and Dally [GD98]. Rusinkiewicz and Levoy [RL00a] developed the system called QSplat, which was one of the first successful systems for point-based rendering based on a multi-resolutional representation using a bounding sphere hierarchy. Botsch et al. proposed a better hierarchical data structure based on the octree representation with a clever encoding scheme [BWK02]. Dachsbacher et al. proposed the sequential point tree representation which enabled the system to retrieve the points at the optimal level in the hierarchical structure on rendering [DVS03].

The method presented in Chapter 4 can be regarded as a variant of model-based rendering when the obtained range images are sufficiently accurate and consistent between views to create a single geometric model. In the proposed method, the scene is rendered by a set of discrete geometric primitives, which are stored in a hierarchical data structure based on octree representation. When the obtained geometry is sufficiently accurate, the method gets close to point-based rendering. On the other hand, when the acquired geometry is unreliable, the system tries to generate the synthetic views mainly from the acquired images; hence, it becomes image-based.

The prior work which is the most relevant to our method is the opacity hull proposed by Matusik et al [MPN+02]. In this method, an opacity map for each reference view-image is generated and used to compensate for the limited resolution of optical sensors. While the opacity maps are applied to a static geometric model created by the image-based visual hull [MBR+00a] in the opacity hull rendering, we adopt the view-dependent geometry [Rad99] to solve the difficulty in integrating erroneous partial models. We also propose an effective method of estimating alpha maps from acquired color images.

1.2.2 Image-based Rendering

Image-based rendering interpolates input reference images and produces novel views in principle without any explicit geometric reconstruction. The problem of image-based rendering can be formulated as a two step process consisting of sampling and rendering. In the sampling stage, a set of light rays is sampled from a function called plenoptic function [AB91] that describes all possible light rays running in the space wished to be rendered. In the rendering stage, the continuous plenoptic function is reconstructed with the captured samples [ZC03]. The plenoptic function $l(x, y, z, \theta, \phi, \lambda, t)$ is a seven-dimensional function that models a three-dimensional dynamic environment by recording the light rays at every space location (x, y, z), toward every possible direction (θ, ϕ), over any range of wavelengths λ and at any time t.

This approach is less generic than model-based rendering since it is impossible to produce the phenomena that are not modeled in the plenoptic function, for example, the change of illumination. In addition, the range and domain of the function is reduced in most of practical methods as reviewed below since the complete reconstruction of plenoptic function is impractical due to its high dimensionality. This limits the range of acceptable viewpoints, typically within a convex spanned by viewpoints of reference images, and makes it difficult to display the object in other environments. However, image-based rendering has such advantages over model-based rendering that a rendering time is independent from scene complexity and geometric reconstruction of the object, often difficult, is in principle unnecessary.

Many researchers have proposed the techniques for image-based rendering by simplifying the seven-dimensional plenoptic function defined in the infinite ray space in order to decrease its high dimensionality. Zhang [ZCar] introduced the surface plenoptic function, that is, a six-dimensional simplification of the original plenoptic function that assumes that light rays do not attenuate in the empty space. Using the representation,

two-dimensional light rays emanated from the regular scene surface can be described. When the geometric model of the surface is known, this approach is equivalent to the surface light fields [WAA+00] for dynamic scenes. McMillan and Bishop [MB95b] introduced plenoptic modeling based on a five-dimensional function obtained by ignoring wavelength and time dimensions in the full plenoptic function. To render a novel view from the representation, the reference images are cylindrically projected and then warped to the viewpoint using their epipolar relationship and some visibility tests. The light field rendering [LH96] and the lumigraph [GGSC96] reduce the full plenoptic function to a four-dimensional function which is parameterized over viewpoint and viewing direction by using two virtual planes. The light field rendering assumes no knowledge about the scene geometry and applies pre-filtering during the capturing to reduce the light field signal's bandwidth, while the lumigraph reconstructs a rough geometry for the scene with an octree algorithm to facilitate the rendering with a small amount of images. Although the lumigraph allows irregular sampling with a tracked handheld camera, unstructured lumigraph rendering [BBM+01a] deals with the problem in a generalized manner. When the geometric model of the scene is given, unstructured lumigraph rendering is known to be equivalent to model-based rendering with view-dependent texture mapping except for the way texture images are interpolated. By restricting that both the cameras and the viewers are on a plane, the plenoptic function can be reduced to a three-dimensional function in concentric mosaics [SH99]. Similarly, a two-dimensional function that allows the cameras and the viewers to turn around at a point is used in image mosaicing [GH97, Mil75] or panoramic mosaic [SS97]. Another parameterization of a two-dimensional function is that the cameras and the viewers can move along a certain 2-manifold with the viewing direction fixed, which is used in QuickTime VR [Che95] and manifold hopping [SWCT02].

All these methods of image-based rendering assume that the reference images are sufficiently given, that is, the plenoptic function used in the method is well sampled. When input reference images do not suffice for the purpose of realistically rendering novel views, it is necessary to extract additional knowledge of the correspondence between reference view-images from the input images in order to compensate for the deficiency of a given data set. As long as the obtained knowledge on image correspondence is physically valid, it is related to the geometric model of the object. Shade et al. [SGwHS98] proposed the method of image-based rendering using a representation, called the layered depth images (LDI), composed of a view-image and corresponding depth of the scene. Chang et al. [CBL99] extended the LDI representation to the hierarchical structure called LDI tree. Pfister et al. [PZvBG00] proposed the three-dimensional representation composed

of a set of LDIs; this representation is referred to as the layered depth cubes (LDC), and utilizes elliptical tangent disks in the rendering. These methods are strongly related to the method of point-based rendering in the sense that the scene is represented as a dense set of pixels with depth. On the other hand, the method of rendering based on a small number of large planar primitives with detailed textures is appropriate to the situation when dense geometry is unavailable. Layers and sprites have been proved successful [LS97, BSA98, SGwHS98] in rendering complex scenes without detailed geometric models. Lengyel and Snyder proposed the coherent layers [LS97] constructed from three-dimensional models for efficient rendering. Schaufler proposed a variant of the layered methods called impostors [Sch95, Sch98a, Sch98b], in which the scene geometry is divided into single or multiple layers according to the distance from the viewpoint and the object is rendered from new viewpoints by warping reference view-images on the layers.

The three methods proposed in this dissertation can be classified into image-based rendering in the sense that all of them do not need the complete geometric model of the object wished to be rendered and can generate synthetic views chiefly or only from view-images.

1.2.3 View-interpolation based on Image Morphing

When the geometric structure of the scene is unknown and the given view-images are too sparse to achieve photorealistic rendering by the methods mentioned in the previous sections, one possible solution is to synthesize the virtual views by interpolating the reference images based on the possibly non-physically based correspondence between the images. In this approach, three-dimensional geometry of the object is not considered at all in the pixel location computation; instead, the synthetic virtual view is generated on the rendering screen by using the image correspondence, which is referred to as *image morphing* or *image metamorphosis* [BN92].

Before applying image metamorphosis to a set of image pairs, the correspondence between the images must be estimated from the images. One of the first attempts to estimate plausible correspondence between images arose in the field of computer vision, aiming to determine the motion of objects in given images. This motion is referred as *optical flow*. The fundamental constraint most commonly used in optical flow estimation is based on the assumption that pixel intensity in the scene is constant between frames [BB95]. Since this assumption is satisfied only when the differences between the images are sufficiently small, systematic errors are conspicuous when the motion between the frames is

large. When some prior information on the objects in the image concerned is available, it helps the method to solve the correspondence problem. This approach requires a knowledge of the features observed in the scene or a training set of specific kinds of images to infer novel views; hence, the range of application is limited to such images as human faces [NMP96, CET98] or medical images [MV98]. On the other hand, Shinagawa and Kunii proposed the method of automatically matching images using only image intensity as a constraint [SK98b]. A set of nonlinear filters, called critical point filters (CPF), is used to extract image features, and dense correspondences between images are then estimated in a multi-resolutional hierarchy. This method works well for the images of objects with similar intensity distribution. In general, however, parts of the images in different colors do not match since the criteria used to match images is the difference in intensity itself although nonlinear filters are applied. In addition, the presence of noise in the image can easily disrupt matching because CPF sensitively responds to the peak and pit of the intensity. Based on the prior techniques for unconstrained image matching, a novel technique for automatic image matching is proposed in Chapter 2. This method yields dense correspondence between images that is supposed to look natural to humans; hence, realistic rendering can be achieved applying image metamorphosis to the obtained reference images.

Once the correspondence between reference images is obtained, the intermediate views between the images can be synthesized by *view-interpolation* [CW93] which consists of warping and blending the reference images. When the correspondence is physically based, for example, geometrically-valid, the intermediate view can be synthesized by applying the three-dimensional image warping proposed by McMillan and Bishop [MB95b, MB95a]. On the other hand, when the correspondence is non-physically based, the technique of image metamorphosis [BN92] is used. Seitz and Dyer [SD96] proposed the intermediate method called *view-morphing* that can blend two different images as if the viewpoint in the synthetic images moved around the object in the morphing sequence. Since the image correspondence estimated by the method proposed in Chapter 2 is unnecessarily physically-based, no constraint other than pixel correspondence is available in rendering. Therefore, the warping and blending of reference view-images are performed using simple tri-linear interpolation in our view-interpolation.

1.2.4 Interactive System for Image-based Modeling and Rendering

With all state-of-the-art techniques for estimating the correspondeces between given images, it is still difficult to estimate plausible correspondences, for example, when the difference between reference images is large. Another approach to combat the problem of data sparseness is to incorporate user interaction attempting to increase the reality of the synthetic images.

Many image-based interactive modeling systems use only one image, which imposes or assumes certain geometric constraints on the scenes. For instance, tour into the picture [HAA97] models the scene by a simple spidery mesh. In single view metrology [CRZ99], two-dimensional projections of three-dimensional parallel lines must be present in the input image, so that the user can click on them for computing vanishing points. An elaborate modeling system was proposed in [MCDD01] where depth values were assigned to pixels in a single picture. Interactive single view systems are difficult to generalize to a sparse light field, which still consists of many images. Furthermore, while it is a straightforward task to perform interactive image segmentation on a single image, consistent propagation of image regions to different images is a challenging task.

The *user interface* (UI) for designing a multi-view interactive modeling system has been a challenge. Most available movie editing tools are, in fact, manual systems, that require frame-by-frame editing and consistency maintenance. Debevec et al. proposed an interactive modeling system that makes use of a depth map derived from a sparse set of views and an effective modeling UI [DTM96]. Plenoptic editing [SK98a] first recovers a three-dimensional voxel model from a sparse light field, and then applies traditional three-dimensional warping to the recovered model. Thus, this automatic system shares the same shortcomings with stereo reconstruction. Recently, feature-based light field morphing [ZWGS02] has been proposed to morph two light fields. The key is an easy-to-use UI for feature specification. The feature polygons are then used to compute the visibility map and perform ray space warping. Consistency is maintained by ray correspondence.

In the method proposed in Chapter 3, an intuitive and easy-to-use UI to facilitate pop-up light field construction is developed. The key to the UI is the concept of a human-in-the-loop in which the user specifies where aliasing occurs in the rendered image. The user input is reflected in the input light field images where pop-up layers can be modified. The user feedback is instant through a hardware-accelerated real-time pop-up light field renderer.

1.3 Thesis Overview

There are three main technical contributions in this thesis which correspond to the breakdown of the following chapters.

- Unconstrained view-interpolation
- Interactive system for image-based modeling and rendering
- Photo-realistic rendering of intricately-shaped objects

These methods are mutually complementary; the first two methods were developed with the goal of achieving automatic and semi-automatic view synthesis from a sparse set of view-images without any geometric information, while the third method takes charge of the rendering based on a relatively large amount of measured data composed of geometric and photometric models.

In this section, each of these topics will be briefly discussed with some simple illustrations to give the reader a hint of what follows.

1.3.1 Preview of Unconstrained View-Interpolation

When the total amount of geometric and photometric information given as input is below a certain theoretical lower bound, it is inevitable for synthetic images to suffer from visual artifacts; this problem, called *aliasing*, is caused by incorrect interpolation of input data. Since the artifacts cannot be removed without any additional information, the common approach to reduce them is either by blurring images so that the aliasing effects are inconceivable or by taking more measurements of objects.

In this dissertation, two different approaches to this problem are proposed; one is automatic, as described below, and the other is semi-automatic, previewed in the next section.

The aliasing in synthesized images caused by the deficiency of given data can be removed by properly interpolating the signals in the data. In order to interpolate input data without aliasing, the plausible, if not correct, mapping between given data must be known before rendering. In our automatic method, image morphing is applied to synthesize views that are not included in given data set. The corresponding points between given images are estimated by unconstrained automatic matching between image features. The features are extracted by filters similar to those used in human eyes; hence, the automatically estimated correspondence is supposed to plausible for human perception.

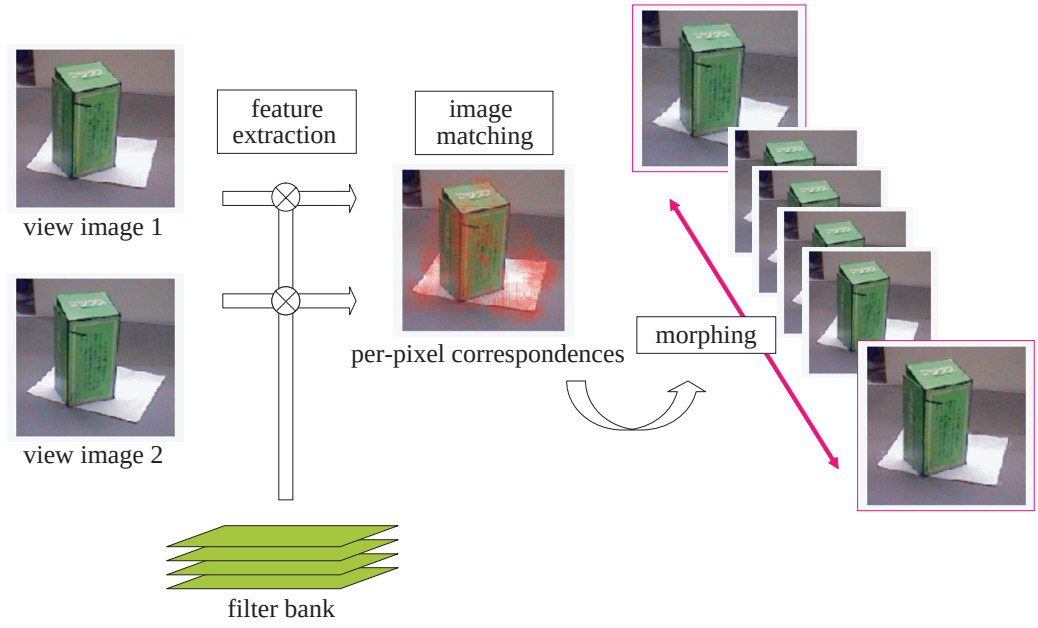


Figure 1.2: The overview of morphing-based rendering system proposed in Chapter 2. This method takes only a set of images as inputs and performs alias-free rendering without any geometric reconstruction. The algorithm consists of two steps: automatic determination of plausible mapping between a pair of images and interactive image morphing with the aid of hardware acceleration.

The advantage of rendering with image morphing over rendering by using geometric information estimated by such methods as stereo estimation is that the point correspondence between views does not need to be accurate geometrically; it is enough for the correspondences to be plausible in human eyes. The difference is obvious when the appearance of objects in given images have such view-dependency as specular reflection.

1.3.2 Preview of Interactive System for Image-Based Modeling and Rendering

Another approach to achieve realistic rendering with an under-sampled data set is to provide an effective interface that enables the user to modify scenes during rendering, if necessary, so that aliasing becomes inconceivable. The advantage of this system is that the user can improve the quality of rendering as much as he or she likes without taking additional measurements of the object. It is also possible for the system to achieve photo-realistic rendering of real-world scenes with no geometry and a few images of the object, gradually

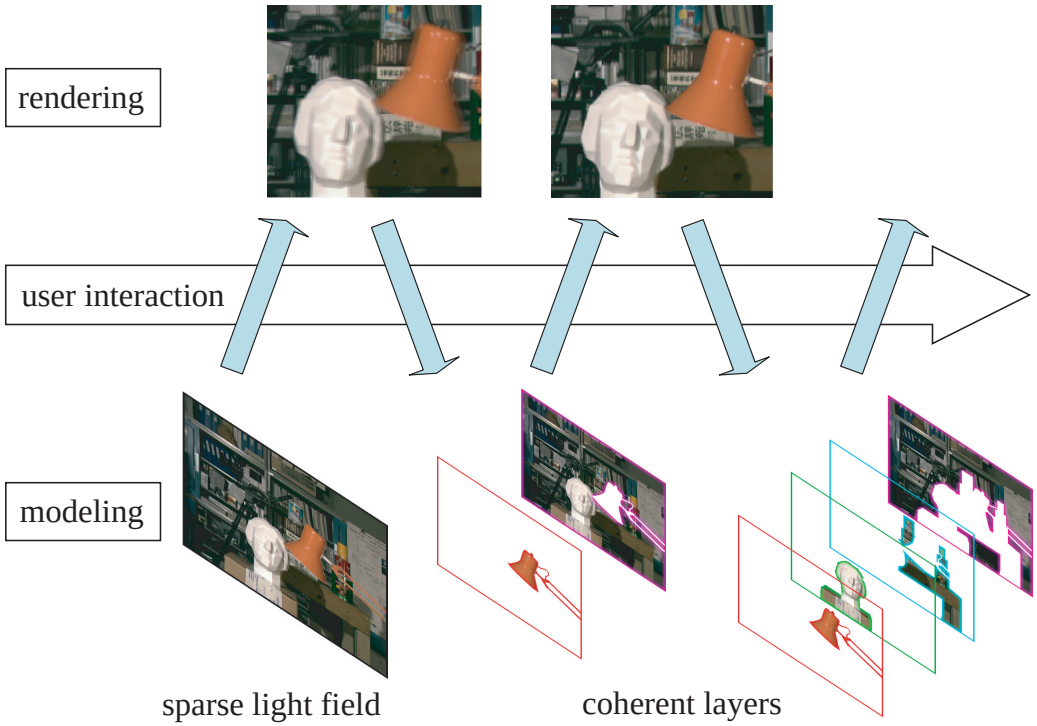


Figure 1.3: The overview of the pop-up light field system proposed in Chapter 3. This system takes a sparse light field as inputs and performs alias-free rendering with the aid of user's interactive modeling. The processes in this system form a cyclic loop consisting mainly of user-assisted modeling and hardware-accelerated rendering. The system first renders the light field with layered unstructured lumigraph rendering. When the synthetic view has the artifacts caused by the sparseness in a given light field, the user specifies the location where aliasing occurs. The system iteratively applies the user's modification to the original light field; hence, the reality in the rendered image can be increased as much as the user wants.

improving the quality of rendering.

The major challenge to this approach is to determine how to represent object scene in a sufficiently compact form, that is to say, how accurate a geometry is needed to achieve alias-free rendering given a set of images. Obviously, photo-realistic rendering with any data set is theoretically possible by modeling the complete geometry manually, but is not practically feasible.

Our solution to the problem is to model the scene by a layered structure, called *coherent layers*, each of which can be regarded as an approximation of the geometry for a single object in the scene. From the observation in the plenoptic sampling theory that the aliasing in synthetic images tends to occur where the depth discontinuity exists, it is possible to

achieve anti-aliased rendering by approximating the scene with a set planar primitives, which represent the objects whose variances of depth are within a certain value.

In order to overcome the aliasing caused by the limited resolution of the camera, the contribution of sub-pixels colors around boundary of each layer are estimated as the opacity of each pixel, by *coherence matting* which maintains the consistency of opacity between views.

In the system, the virtual view of the scene is first synthesized by unstructured lumigraph rendering using given data consisting of color images. Geometry of the scene can also be used if available. Due to a deficiency of either or both geometric and photometric information, aliasing may be observed in the synthetic image. The user then specifies where aliasing occurs on the rendering screen, and performs image segmentation so that the scene becomes a set of coherent layers.

1.3.3 Preview of Photo-realistic Rendering of Intricately-Shaped Objects

The synthetic virtual view of real-world scenes is rendered using sets of range images and color images taken from various viewpoints. The problem in rendering based on a practical data set is that, when each range image given as input contains considerable amount of geometric errors, it can be difficult to create a global model that is consistent for each view, even with such robust techniques of geometric integration [HSIW96, CL96, WSI98].

The proposed solution to the difficulty in consistent modeling is to adopt a two-level representation of the object shape: global geometry and local geometry. After integrating given range images into a volumetric representation, the maximal resolution of the volume is determined by analyzing the consistency of the shape in the volume. Each range image given as input is then modified so that unobserved parts of the object are filled with appropriate depths brought from global geometry. Thus, the shape of the object to be rendered is modeled as a globally-defined volumetric geometry and a set of locally-defined depth maps.

On the other hand, photometric information on the object, that is, a set of color images, also has a difficulty in consistent integration. Hence, the images are represented as view-dependent textures on the object. Different from other methods of view-dependent texture mapping, opacity value for each pixel in the color image is automatically estimated and then interpolated between views.

The scene is then rendered by a set of texture-mapped polygons with hardware accel-

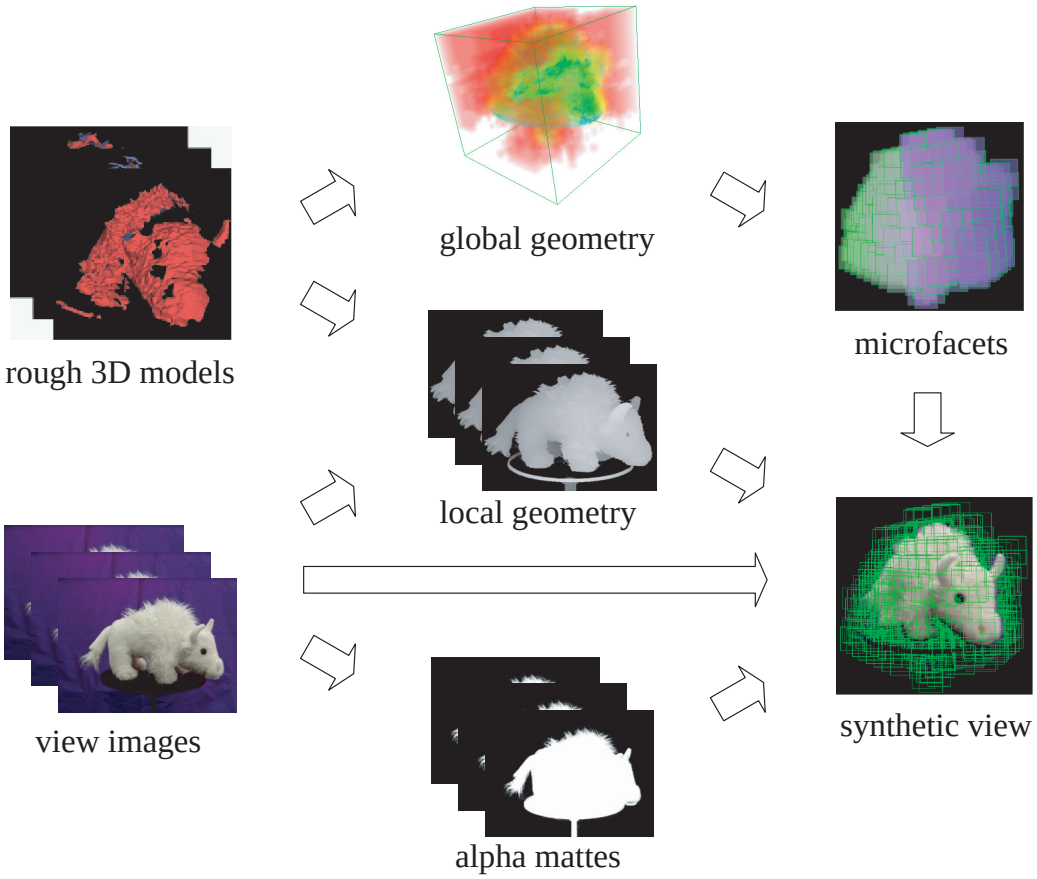


Figure 1.4: The overview of microfacet billboarding proposed in Chapter 4. This system takes a set of range images and colors images as inputs. The algorithm consists of two steps: the modeling process and the rendering process. The modeling step usually takes some time, while the rendering step is interactive.

eration. Based on estimated global geometry, we generate a set of rendering primitives, onto which the detailed shaped is rendered using local geometry. The details of the object's appearance are then synthesized using color and opacity images using the technique of view-dependent texture mapping.

Chapter 2

Unconstrained View-interpolation

2.1 Introduction

In this chapter, the method of rendering based on a sparse set of images without any geometric model is presented. The proposed technique can be considered as purely image-based rendering in the sense that it does not reconstruct any geometric model explicitly. Instead, it synthesizes the virtual views only from the reference images by warping and blending the images in the image space, that is, in the two-dimensional coordinate system.

The development of purely image-based modeling and rendering has great impact on both industry and academia since one of the chief problems of rendering scenes based on the measured data set is that it is difficult to create the geometric model, while the image of the scene can be easily taken by using digital camera. For instance, geometric modeling often requires such time-consuming processes as camera calibration, setup of lighting environment, measurement by a special hardware, integration and simplification of obtained data set for rendering purpose, etc. It can easily be imagined that most people are unfamiliar with these techniques and simply do not want to use such a complicated system. In addition, it is also difficult to measure the accurate shapes of such objects as transparent, shiny and intricately-shaped surfaces.

When it is hopeless to recover scene geometry from a set of two-dimensional images, another solution to the problem in realistic view synthesis without aliasing is to interpolate the reference view-images using the mapping between the images. This approach is referred to as *view-interpolation* [CW93]. Determining the correspondence between two or more images is one of the most common and well-studied problems in computer vision. However, even if we take into consideration such constraints on given images as epipo-

lar geometry, constant reflectance or small changes between images, it is difficult to find correct correspondences.

In contrast to prior work [CW93, SD96], our method of view-interpolation automatically extracts the plausible, if not physically-valid, mapping between the images without any assumption on the scene geometry, and then image morphing is applied to synthesize views that are not included in the given set of images.

The advantage of rendering with image morphing as opposed to rendering by using geometric information estimated by such methods as stereo reconstruction is that the point correspondence between views does not need to be accurate geometrically; it is enough for the correspondences to be plausible in human eyes. The difference is obvious when the appearance of objects in given images has such view-dependency as specular reflection. In addition, this method can be further extended to the problem of the interpolation between unrelated images, such as morphing of face images and feature-based texture morphing [LLSY02].

The proposed method consists of two parts: determination of the mapping between reference images, and view-interpolation using image morphing.

In the first step, per-pixel correspondence between two images without depending on either the camera pose or objects is estimated only from the images. When no geometrical or physical information on what appears in given images is available, the difficulties in solving the correspondence problem consist of two parts. One is the selection of features by which the desirable correspondence between images is defined. Since the corresponding points may differ in their image intensities, it is essential to define the similarities between the images in a plausible manner. Another difficulty involved is how to prevent the mapping from being converged into a local minimum without such defects as strong distortion when the images come with numerous features and distant correspondences.

In the second step, the synthetic virtual views are rendered by applying morphing to reference view-images with the determined mapping. The morphing technique is composed of warping and blending of images; both of them are effectively implemented with commodity graphics hardware to achieve interactive rendering.

2.2 Determining Plausible Maping between Images

In this section, the method of determining per-pixel correspondence between two images only from the images is proposed. As mentioned in the previous section, the key to solving

this problem is: how to select features by which the desirable correspondence between images is defined and how to prevent the mapping from being converged into a local minimum without defects such as strong distortion when the images come with numerous features and distant correspondences. The algorithm of determining the correspondences is as shown in Algorithm 2.1.

input: Image I_{src}
input: Image I_{dst}
output: Mapping M
local: FilterBank F
local: FeatureVectorField V_{src}
local: FeatureVectorField V_{dst}
local: CostOfMapping c
1: $M \leftarrow$ Identity()
2: $F \leftarrow$ SetupFilterBank()
3: $V_{src} \leftarrow$ ExtractFeature(I_{src}, F)
4: $V_{dst} \leftarrow$ ExtractFeature(I_{dst}, F)
5: **repeat**
6: $c \leftarrow$ Cost(M)
7: **for all** feature vectors $v_{src} \in V_{src}$ **do**
8: $M \leftarrow$ RefineCorresponce(v_{src}, V_{dst}, M)
9: **end for**
10: **until** $c <$ Cost(M)

Algorithm 2.1: Unconstrained image matching

2.2.1 Problem Formulation

Let $I : \mathbb{D} \to \mathbb{V}$ be an image color distribution, where $\mathbb{D}$ is a set of pixel coordinates and $\mathbb{V}$ is the space of pixel values. The problem to be solved is then formulated as the estimate of optimal mapping $m : \mathbb{D} \to \mathbb{D}$ from images I_{src} to I_{dst} in the sense that an appropriate measure ε_D^2 which expresses the difference between $I_{src}(m(\cdot))$ and $I_{dst}(\cdot)$ is minimized. We allow the images to be upsampled, that is, the resolution of $\mathbb{D}$ may be higher than that of pixel grids. In addition, mapping is restricted to a bijection, that is, m^{-1} is uniquely defined at any point in $\mathbb{D}$.

Hereafter, mainly the method of matching two dimensional gray images is discussed

for simplicity, although the proposed framework could be applied to two and three (and probably more) dimensional color images, as shown in Section 2.4.

2.2.2 Feature Extraction

In order to solve correspondence problems, the algorithm attempts to match features in one image with corresponding features in the other. The intensity of pixels in an image should not be used directly since the corresponding pixel may have different intensities in the concerned problem. Unlike solving other correspondence problems in computer vision like stereo photogrammetry, we may have to find the correspondence between points on two unrelated objects. Nevertheless, matching is expected to appear natural to humans. This observation implies that it is helpful to apply the feature detector actually used in human vision.

Similar to the method in [MBSL99], we use *partial derivative of a Gaussian* which well approximates visual receptive fields in early vision [You85]. When the two dimensional images are matched, we use a set of rotated Gaussian derivative filters $K_{\sigma,n,\theta}$ defined as

$$K_{\sigma,n,\theta}(x,y) = G_{\sigma}^{(n)}(x\cos\theta - y\sin\theta) \times G_{\sigma}^{(0)}(x\cos\theta + y\sin\theta) \tag{2.1}$$

where $G_{\sigma}^{(n)}$ is the n-th derivative of a Gaussian with standard deviation σ.

If some of the filters in a *filter bank* have linear dependency, filter response may be redundant. By applying singular value decomposition to the matrix consisting of the vectors of filter kernels, the number of bases required for n-th Gaussian derivative will be $n+1$ [JM92]. Based on this, we used 20 filters consisting of two first derivatives and three second derivatives of Gaussian with four different spatial scales, as shown in Figure 2.1.

At the beginning of the image matching, a filter bank $F = \{F_i\}(i = 1,\ldots,N_f)$ composed of $K_{\sigma,n,\theta}$ with N_f different parameter settings is applied to each of the given images, and then the filter responses are stored in the form of vector fields with the same size as the original images. We call these vector fields *feature vector fields*. The correspondences between images are determined only by the fields.

2.2.3 Composition of Objective Function

Optimal mapping m from I_{src} to I_{dst} is estimated by maximizing the similarities between $I_{src}(m(\cdot))$ and $I_{dst}(\cdot)$ under an appropriate constraint on the regularity of the mapping. This

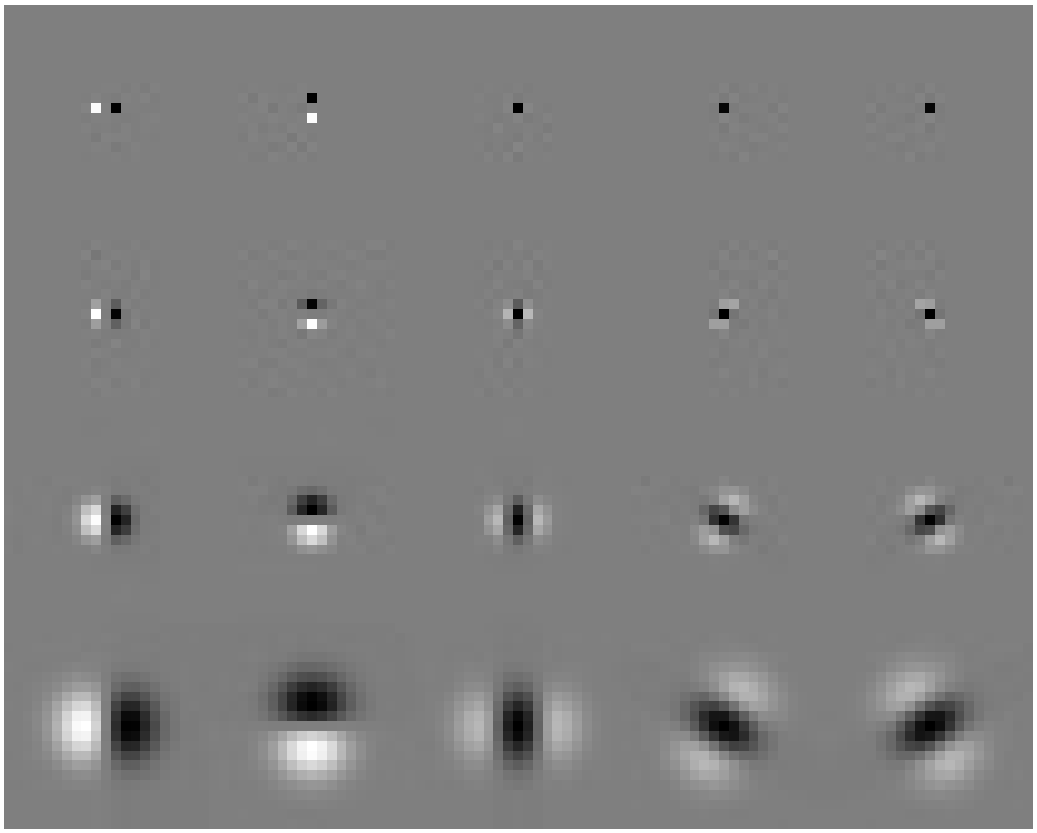


Figure 2.1: The filter bank used to generate feature vectors. We used 24 filters composed of first and second partial derivatives of Gaussian, with 4 scales, in 2 and 3 orientations for the first and second derivatives respectively. The filters have zero mean and each of them is divided by its L_1 norm for scale invariance.

problem can be formulated by the objective function concerning the similarity of transformed images and mapping smoothness.

The first criterion refers to the matching quality of the entire region of the images. That is, the objective function ε_D^2 for the similarity is defined as

$$\varepsilon_D^2 = \int_{p\in\mathbb{D}} \sum_{i=1}^{N_f} \rho_D\Big((F_i * I_{src})(p) - (F_i * I_{dst})(m(p))\Big) \tag{2.2}$$

where the operator $*$ denotes the convolution and ρ_D is the *M-estimator* [HRRS86] which is robust for the perturbations of feature vectors. In our implementation, *Lorentzian function*

$$\rho(x) = \log\Big(1 + \frac{1}{2}\Big(\frac{x}{\sigma}\Big)^2\Big) \tag{2.3}$$

is used as the M-estimator. ρ_D is defined by the sum of Equation (2.3) with the scale $\sigma = \sigma_D$ for all elements of $F_i * I$.

The second criterion is based on the regularity of the mapping. Although the feature vectors defined in Section 2.2.2 may have sufficient information for determining the correspondence uniquely, regularization is essential since it is impossible to determine the mapping where the image intensity is constant. The objective function ε_S^2 reflecting the smoothness of the mapping $m = (m_1, \ldots, m_{dim\mathbb{D}})$ can be defined as

$$\varepsilon_S^2 = \int_{p\in\mathbb{D}} \sum_{d=1}^{dim\mathbb{D}} \rho_S\Big(\nabla m_d(p)\Big) \tag{2.4}$$

where ρ_S is the robust M-estimator that returns the sum of Lorentzian with scale σ_S for all elements of ∇m_d in our implementation.

By combining Equation (2.2) and Equation (2.4) with the weighting parameter α^2, the objective function ε^2 to be minimized is defined as follows.

$$\varepsilon^2 = \varepsilon_D^2 + \alpha^2\varepsilon_S^2 \tag{2.5}$$

The parameter α^2 is used to regularize mapping in the region where the color is nearly constant; hence, α^2 can be small. The parameter σ_S should be sufficiently large since the difference of the features is the only cue to determining the correspondence in our problem. The parameter σ_D is heuristically selected in our experiments, for instance 5% of image width.

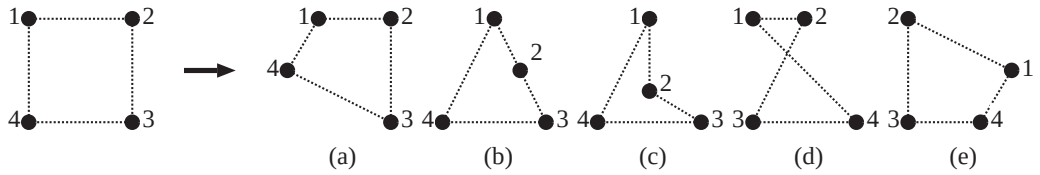


Figure 2.2: Possible configurations of four vertices which form a square in one image and are transformed by the estimated mapping. Only pattern (a) is allowed in our method.

2.2.4 Minimization of Objective Function

In general, it is difficult to find the optimal parameter α^2 since the difference between images can be so large that global matching may cause local distortions such as twists and flips in the mapping which violate our assumption on bijectivity. We prevent the mapping from being distorted by imposing restrictions for the mapping to preserve its local convexity.

When the four vertices of a small square in I_{src} are mapped into I_{dst} by m, the possible configurations of these four vertices are as shown in Figure 2.2 since different points are mapped to different points due to the bijectivity of the mapping. In the determination of mapping m, we ignore patterns (d) and (e) because bijectivity is broken either inside or near the quadrilateral. Note that, even if we ignore the mapping in patterns (d) and (e), it is still feasible to produce the mapping between the images containing occlusions by accounting for the mapping that shrinks or enlarges the squares. We also prohibit the mapping in patterns (b) and (c), because once these mappings are generated, the mapping around the point indicated by 2 is strongly restricted, thereby leading to a local minimum in the optimization. Consequently, we allow only the mapping consisting of the local transformation in pattern (a). This condition is called the *convexity condition*.

In order to determine mapping which satisfies the convexity condition as a whole, each correspondence in the mapping is determined so that the local convexities are preserved. This can be achieved by restricting the area where the correspondence is sought as follows. When a point $p_{x,y}$ in I_{src}, illustrated by a white point on the left in Figure 2.3, is mapped into I_{dst} by a mapping m, the convexity of m is maintained as long as $m(p_{x,y})$ is mapped into the quadrilateral formed by $m(p_{x+\Delta x,y})$, $m(p_{x,y+\Delta y})$, $m(p_{x-\Delta x,y})$ and $m(p_{x,y-\Delta y})$ as illustrated on the right in Figure 2.3. Since $m(p_{x,y})$ affects the convexity of only adjacent quadrilaterals, whole convexity of m is guaranteed by beginning with a convex mapping as an initial estimation and then modifying it so that the local convexity is maintained.

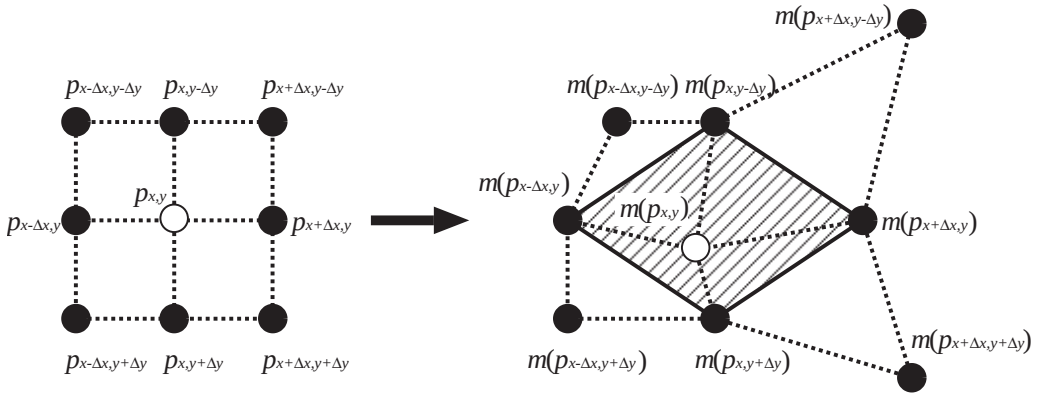


Figure 2.3: Search area for single correspondence. The convexity of mapping is maintained as long as each point $m(p_{x,y})$ in a source image (a white point on the left) is mapped into the quadrilateral formed by $m(p_{x+\Delta x,y})$, $m(p_{x,y+\Delta y})$, $m(p_{x-\Delta x,y})$ and $m(p_{x,y-\Delta y})$ as illustrated by a hutched area on the right.

2.2.5 Implementation

The geometrical constraint introduced by the convexity condition makes it difficult to determine all correspondences simultaneously. Instead, we have adopted a sequential method similar to that used in [SK98b]. For each point p in I_{src}, we calculate the cost functions for all possible correspondences which satisfy the convexity condition, and then choose one of the correspondences whose cost is the minimum. The point p to be modified is chosen in a random order to avoid the bias in obtained mapping.

The minimization of the objective function ε^2 is carried out in multi-resolutional hierarchy to reduce the risk of the mapping's falling into local minima. Before determining the mapping between images, the feature vector fields generated using the filter bank are repeatedly down-sampled into half the size of the original; that generates pyramids of feature vector fields in different resolutions. We start in the lowest resolution and propagate the calculation toward the highest resolution in turn, using an optimized mapping in a resolution as an initial estimation in the next.

In each resolution in the pyramid, the mapping is refined iteratively until the minimization converges. This iteration is essential in our algorithm to correct erroneous correspondences in lower resolution.

Given image colors as inputs, we apply the filter bank defined in Section 2.2.2 to each color channel in the images separately, and then concatenate obtained feature vector fields into a single field. For instance, if the images are RGB color, we generate the feature

vector field composed of 60-dimensional vectors, while gray images yield 20-dimensional vectors.

In our current implementation, the integral in ε^2 is discretized and replaced by a summation for all pixels in source image I_{src}. On the other hand, the domain and range of mapping m is discretized into sub-pixels. In general, the smaller size of sub-pixel yields the better mapping, but leads to the larger computational cost. We set the size of sub-pixels to be 0.1 to 0.5.

2.3 Interactive Rendering for Image Morphing

2.3.1 Warping and Blending

Once the correspondence m between the pair of the images I_{src} and I_{dst} has been obtained, intermediate views between them can be synthesized by image morphing. Image morphing is a technique typically used as an animation tool for smoothly and gradually blending one image with another. The algorithm of morphing consists of two processes: *warping* and *blending*. The basic idea of warping is to gradually distort I_{src} into W_{src} by m and I_{dst} into W_{dst} by m^{-1}, such that

$$W_{src}(p) = I_{src}(m^{-1}(p)) \tag{2.6}$$

$$W_{dst}(p) = I_{dst}(m(p)) \tag{2.7}$$

where $p \in \mathbb{D}$ and $\mathbb{D}$ is the pixel coordinates in the images. Once the warping of the images is defined, the warped images are then blended so that they gradually change their appearance. As the blending proceeds, I_{src} is gradually distorted and is faded out, while I_{dst} starts from W_{dst} and is faded in. Thus, the early images in the sequence look like I_{src}. The middle image of the sequence is the average of I_{src} and I_{dst}. Then the last images in the sequence are similar to I_{dst}. The morphing is based on the observation that, when the middle image looks good, then probably the entire sequence is supposed to look good.

Since the blending of two images is a somewhat straightforward task, the major problem in morphing is how to warp an image. Wolberg [Wol90] formulated the fundamentals of digital image warping and proposed two-pass algorithm which warps images based using spline mapping defined in two dimensions grid coordinates. Beier [BN92] proposed a more general and robust method, field morphing, that warps the image based on sparse correspondences defined using line segments. Field morphing has several advantages over

spline-based warping and is suitable for producing morphing sequence based on user specified correspondence since it can smoothly transform the image with a small number of sparse correspondences.

Another problem with warping is how to synthesize the warped image. The existing method of warping an image can be classified into two groups [Wol90]. The first, called forward mapping, scans through an image pixel by pixel, and copies them to their appropriate place in the warped image. The second group, reverse mapping, goes through the destination image pixel by pixel, and samples the corresponding pixel from the source image. The most important feature of inverse mapping is that every pixel in the destination image gets set to something appropriate. In the forward mapping case, some pixels in the destination might not get painted, and would have to be interpolated. Therefore, the image is deformed by reverse mapping in the proposed system.

In the proposed system for image-based rendering, a simple grid-based bilinear warping with inverse mapping followed by linear blending is adopted. It is unnecessary to warp the images by sophisticated methods such as field morphing since the correspondence determined by the method proposed in previous section is sufficiently dense. Let I_{src} and I_{dst} be the source and destination image, respectively, $m : \mathbb{D} \to \mathbb{D}$ be the mapping from I_{src} to I_{dst}, and $\tau \in [0, 1]$ be the parameter that controls the ratio of warping and blending during morphing; then, the morphed image $M : \mathbb{D} \to \mathbb{D}$ from I_{src} to I_{dst} using the parameter t is synthesized using trilinear interpolation as

$$M(p) = (1 - \tau) \cdot \texttt{bilerp}(I_{src}, p') + \tau \cdot \texttt{bilerp}(I_{dst}, m(p')) \tag{2.8}$$

where $p, p' \in \mathbb{D}$ satisfies

$$p = (1 - \tau) \cdot p' + \tau \cdot m(p') \tag{2.9}$$

and $\texttt{bilerp}(I, p)$ indicates the bilinear sampling of an image I at the position p. The parameter values $\tau = 0$ and $\tau = 1$ correspond, respectively, to the beginning and ending point in the morphing sequence $\{M | t \in [0, 1]\}$ from I_{src} to I_{dst}.

2.3.2 Hardware-accelerated Rendering for Interactive Morphing

The image warping based on reverse mapping requires a high computational cost for each pixel. In order to accelerate the calculation of reverse mapping, the warping is performed with the aid of graphics hardware. In this approach, the image is represented as a set of quadrilateral polygons with texture image. The polygons are tiled so that they cover

the whole image without overlapping. In warping, the corners of the polygons are transformed according to the estimated correspondence. The texture coordinate in the transformed polygon is calculated, and the texture image is smoothly interpolated by graphics hardware.

The warping of both I_{src} and I_{dst} can be effectively implemented based on polygon rendering and texture mapping with the mapping m from I_{src} to I_{dst} without any explicit calculation of the inverse mapping m^{-1}.

input: Image I_{src}
input: Image I_{dst}
input: Mapping M
input: BlendingParamter τ
input: ImageGrid $\{p_{grid}\}$
local: Polygons $\{\triangle_{src}\}$
local: Polygons $\{\triangle_{dst}\}$
1: $\{\triangle_{dst}\} \leftarrow$ GenerateGrid($\{m(p_{grid})\}$)
2: $\{\triangle_{dst}\} \leftarrow$ Warp($\{\triangle_{dst}\}$, τ)
3: $\{\triangle_{dst}\} \leftarrow$ MapTexture($\{\triangle_{dst}\}$, I_{dst})
4: RenderToFrameBuffer($\{\triangle_{dst}\}$)
5: $\{\triangle_{src}\} \leftarrow$ GenerateGrid($\{p_{grid}\}$)
6: $\{\triangle_{src}\} \leftarrow$ Warp($\{\triangle_{src}\}$, τ)
7: $\{\triangle_{src}\} \leftarrow$ MapTexture($\{\triangle_{src}\}$, I_{src})
8: BlendToFrameBuffer($\{\triangle_{src}\}$, τ)

Algorithm 2.2: Hardware-acclerated morphing

The algorithm of the hardware-accelerated morphing is as shown in Algorithm 2.2. At the beginning of the rendering, the image coordinate $\mathbb{D}$ is optionally downsampled into the coarse set $\hat{\mathbb{D}} \subseteq \mathbb{D}$. The images are then rendered by the set of quadrilateral polygons composed of four adjacent grid points $p_{grid} \in \hat{\mathbb{D}}$ using the images I_{src} and I_{dst} as texture images. Let v_{grid} be the warped coordinate of p_{grid}, and $t_{src}, t_{dst} \in \mathbb{D}$ be the texture coordinate of v_{grid} for I_{src}, I_{dst}, respectively; then, the warped images with the parameter τ are rendered by the following parameters.

$$\forall p_{grid} \in \hat{\mathbb{D}}: \quad v_{grid} = (1-\tau) \cdot p_{grid} + \tau \cdot m(p_{grid}) \tag{2.10}$$

$$t_{src} = p_{grid} \tag{2.11}$$

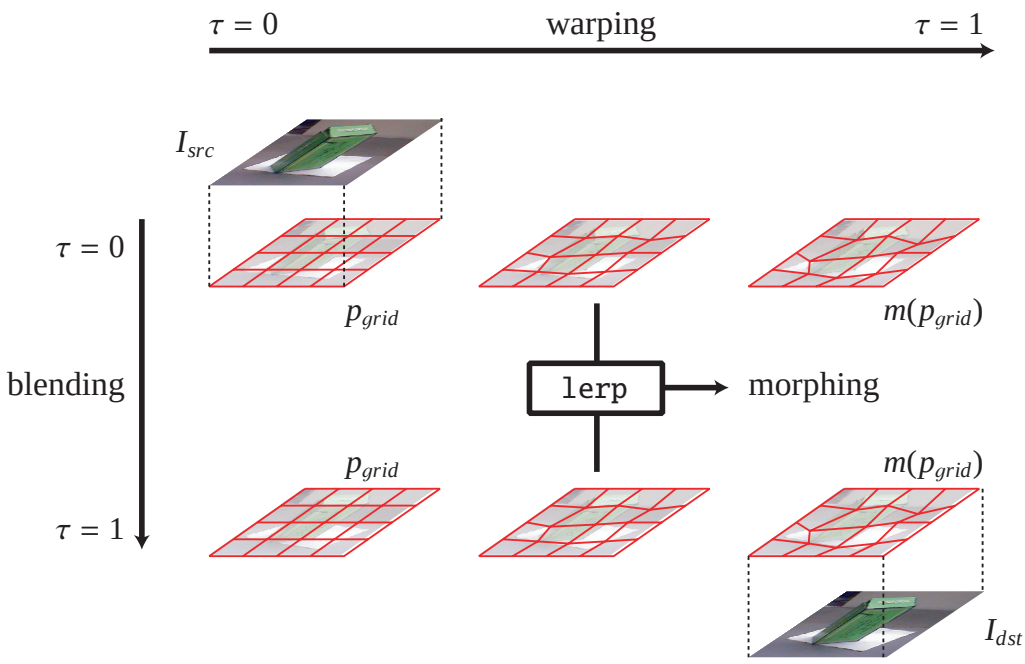


Figure 2.4: Hardware-accelerated image morphing

$$t_{dst} = m(p_{grid}) \tag{2.12}$$

The warped image of I_{src} can be rendered by the polygons composed of $\{v_{grid}\}$ with texture image I_{src} whose corresponding texture coordinate is $\{t_{src}\}$. The warping of I_{dst} can also be rendered similarly using $\{v_{grid}\}$, I_{dst} and t_{dst}.

The blending of these warped images is also accelerated by the graphics hardware. Since the blending is performed by simple linear interpolation in our system, the morphed image M can be rendered by alpha-blending two warped images with the blending value τ on frame buffer. When the system is capable of multi-texture mapping to polygons, it may be possible to accomplish the rendering twice as fast since the number of issued polygons becomes half of that used in the algorithm based on alpha blending.

The warping and blending of I_{src} and I_{dst} based on polygon rendering are illustrated in Figure 2.4.

(a) input (I_{src}) (b) input (I_{dst}) (c) determined mapping

Figure 2.5: Example of mapping determined by our algorithm. Given (a) and (b) as inputs, mapping (c) between them is calculated. The mapping is illustrated by needle diagram.

2.4 Experimental Results

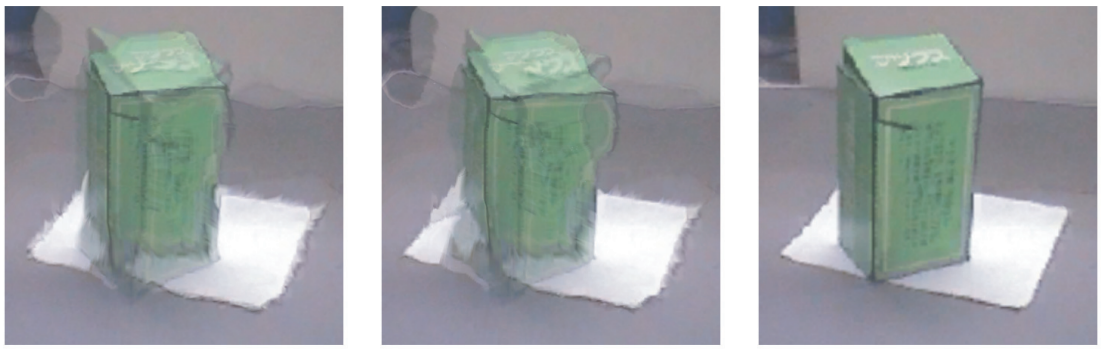
In this section, extensive results of the proposed method for unconstrained image matching and interactive image morphing are presented. The proposed algorithms are implemented on a standard PC with Pentium3 1GHz, 768MB main memory and an ATI Radeon 9800 PRO graphics card.

2.4.1 Results of Unconstrained Image Matching

Figure 2.5 shows the results of matching between two images of an object which is slightly rotated. The resolution of the image is 128^2. Figure 2.5 (a) and (b) shows the input images. Figure 2.5 (c) is the needle diagram for the pixel correspondence determined by the proposed method. Although no geometric information such as camera poses is used in the estimation, the rotation of the object is correctly extracted.

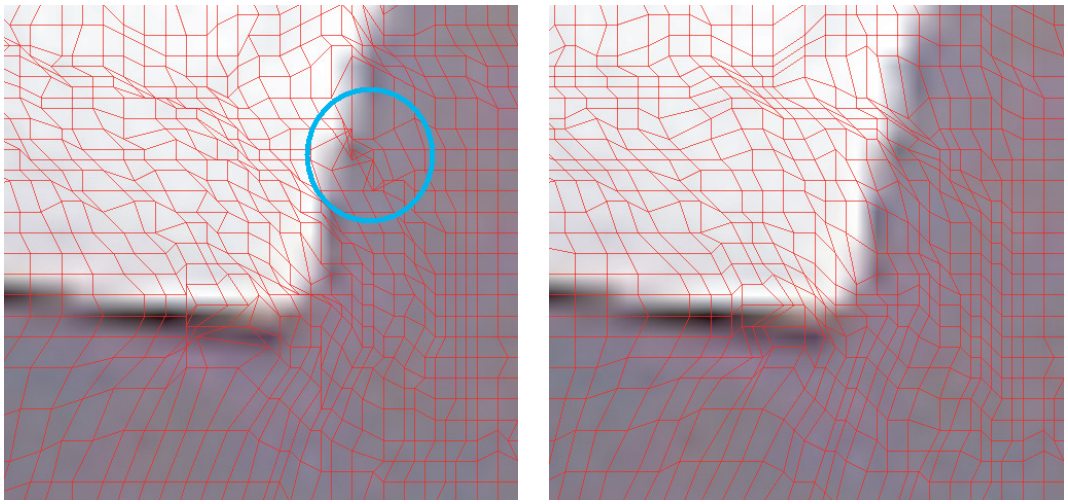
In Figure 2.6, robustness of our method to the perturbation of image intensities is compared. The mapping between Figure 2.5 (a) and 10% brightened (b) was estimated by using mean filter, CPF [SK98b] and our proposed filters; then, intermediate images are generated by obtained mappings. Since mean filter and CPF are directly affected by the change of image intensity, the resultant mappings are totally disrupted, while our method yields a plausible mapping.

We also compared our proposed method with others in terms of quality of mapping generated from the images used in Figure 2.5. A comparison of the smoothness of mapping with [SK98b] is presented in Figure 2.7. Some flips and twists of pixels can be observed



(a) mean (b) CPF (c) ours

Figure 2.6: Comparison of estimation robustness. The figures present intermediate images between Figure 2.5 (a) and 10% brightened (b). Mappings between the images are obtained using (a) mean filters, (b) CPF [SK98b] and (c) our proposed filters.



[SK98b] ours

Figure 2.7: Comparison of the mapping generated by [SK98b] (left) and our proposed method (right). Flipping and twisting of the local correspondence may be observed in the circle on the left.

Table 2.1: Performance comparison of image matching algorithms. The left column in each cell shows the execution time (seconds) and the right shows the sum of the squared differences of intensities.

	[SK98b]		proposed method	
data (size)	time	error	time	error
boxes (128^2)	2.1	6.0×10^6	8.5	5.0×10^6
textures(128^2)	2.0	3.8×10^8	10.7	1.6×10^8
faces (256^2)	8.2	1.3×10^8	51.3	1.1×10^8
brains (128^3)	-	-	8823	1.2×10^{10}

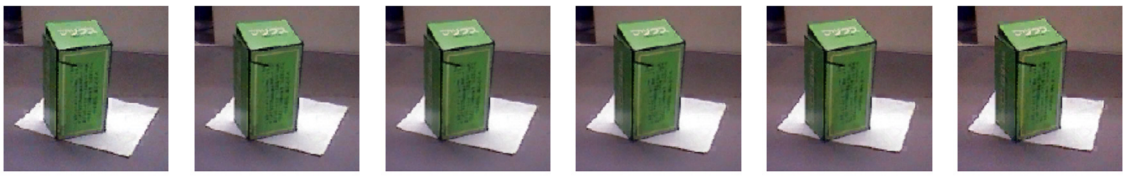


Figure 2.8: Results of view-interpolation for a rotated object. This image sequence is generated by trilinear interpolation of the images (a) and (b) in Figure 2.5 with automatically estimated correspondence.

in the mapping calculated by [SK98b], while our mapping maintains local smoothness.

Table 2.1 contains a comparison of the matching speed (measured in seconds) and matching quality (measured by sum of the squared differences of intensities) with the method in [SK98b]. The result shows that the quality of mapping has improved, while the speed of calculation has increased by about five times, which depends on data owing to iterative minimization.

2.4.2 Results of View-interpolation

Figure 2.8 shows the result of view-interpolation between two images presented in Figure 2.5 (a) and (b). Although the determined mapping may not be the same as an exact rigid transformation, the in-between views are synthesized without conspicuous visual errors.

Figure 2.9 shows the result of view-interpolation between two images. The top two figures indicate the reference images for view-interpolation. The synthesized intermediate view-image is presented on the left in the middle row. The mapping between the reference

(a) input

(b) input

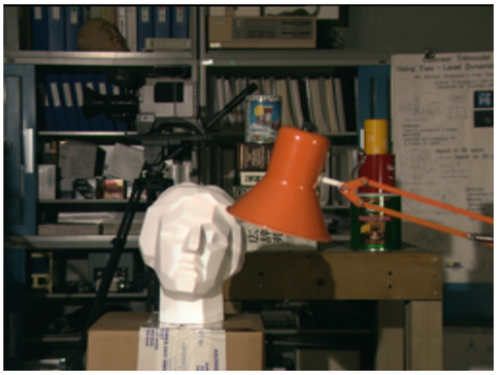
(c) synthesized intermediate view-image

(d) real view-image at the viewpoint of (c)

(e) view-interpolation sequence

Figure 2.9: Result of view-interpolation for Pokeḿons. (a) and (b) are two input images. (c) is the intermediate image synthesized from (a) and (b) by the proposed method of unconstrained view-interpolation. (d) is a real view-image taken at the viewpoints halfway between those of the inputs and unused in the process of the view-interpolation. (e) is the view-interpolation sequence between (a) and (b).

(a) input

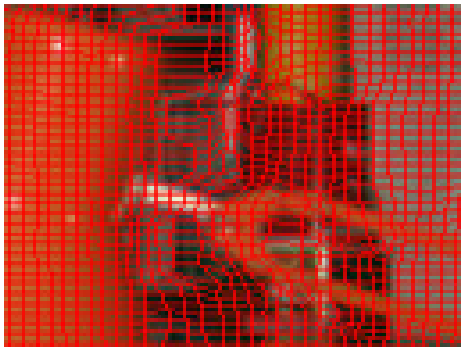


(b) input

(c) synthesized intermediate view-image

(d) real view-image at the viewpoint of (c)

(e) close-up of (c)



(f) close-up of (c) with grids

Figure 2.10: Results of view-interpolation for Tsukuba sequence. (a) and (b) indicate two input images. (c) is the intermediate image synthesized from (a) and (b) by the proposed method of unconstrained view-interpolation. (d) is a real view-image taken at the viewpoints halfway between those of the inputs and unused in the process of the view-interpolation. (e) and (f) are close-up view of the synthesized image. Owing to the occlusion of objects, some parts of mapping is estimated incorrectly.

images is estimated in the resolution of 256×256. The right in the middle is the real view-image taken at the position in the middle of the reference viewpoints. Most parts of the synthesized image look quite similar to those of the real image, but small rubber-band effects are observed between green and red toys in the synthesized image. This is caused by interpolating the points which are visible in one of the reference view and invisible in the other due to occlusion. The bottom row presents the sequence of view-interpolation between the given reference view-images.

Figure 2.10 shows the result of view-interpolation for another scene. This scene has many objects occluding one another; hence, is difficult to estimate accurate mapping. Using the top two images as reference views, the intermediate view-image is synthesized as shown on the left in the middle row. The mapping used to interpolate the views is in the resolution of 256×256. A view-image taken at the same viewpoint is presented on the right in the middle row for comparison. Owing to the large occlusions and the narrow objects, some parts of the scene correspond to incorrect parts, as shown in the close-up of the synthesized image. The black cable on the orange lamp is doubly observed in the synthetic view as a result of incorrect image correspondences. This effect may be reducible by increasing the resolution of mapping at the cost of the computation time.

The unconstrained view-interpolation between the views of a translucent and specular object is presented in Figure 2.11. Since the view images are rectangular, they were transformed into squares before determining the correspondences. The mapping between the original images were generated by converting the estimated mapping to that in the original size. It is extremely difficult to reconstruct the accurate geometry of the object, because the change of specular reflection causes the mis-estimation of the correct geometric correspondences between the images. Although the translucency and absence of texture also disturb the estimation, the unconstrained view-interpolation of the reference view-images enables the system to yield natural synthetic views without any explicit geometric reconstruction.

Note that neither camera poses nor camera intrinsic parametes was estimated in these experiments.

2.4.3 Applications of Image Morphing

Although the primary goal of this research is to synthesize an images sequence between the different views of a static scene, the proposed method of image matching is applicable to the morphing between the images of different objects.

Figure 2.11: Results of view-interpolation for a translucent and specular object. The left-most and right-most images are the reference views and the center is the intermediate image.

In Figure 2.12, the results of automatic morphing of three different persons are shown. The resolution of the images is 256^3. The mappings between (a), (c) and (c), (e) are calculated separately. Although the middle person wears the glasses, the robust M-estimator successfully ignore the image features which cannot correspond to any parts in the counterparts.

Our algorithm was applied to texture metamorphosis as shown in Figure 2.13. The size of the texture images is 256^3. The result is similar in quality to those generated by the semi-automatic method proposed by Liu et al. [LLSY02], while ours are automatically generated.

Another application is the images registration. We have extended our algorithm to the registration of three dimensional images (volumes). The filters used to extract features are partial derivatives of three dimensional Gaussian functions, and the mapping is calculated so that the small hexahedron shaped by adjacent eight points remains convex. Figure 2.14 shows the result of registration for the brain. In the figure, (a) and (b) are the volumes of human heads in 128^3 resolution obtained by magnetic resonance imaging (MRI). The correspondences between the volumes are calculated by our proposed method, and then the volume (a) is transformed into (c) by trilinear transformation, without blending (a) and (b). The volume is visualized by isosurface rendering with an appropriate isovalue and lighting.

(a) input (b) (a)+(c) (c) input (d) (c)+(e) (e) input

Figure 2.12: Blending of human faces. (a), (c) and (e) are inputs. (b) and (d) are generated from adjacent images.

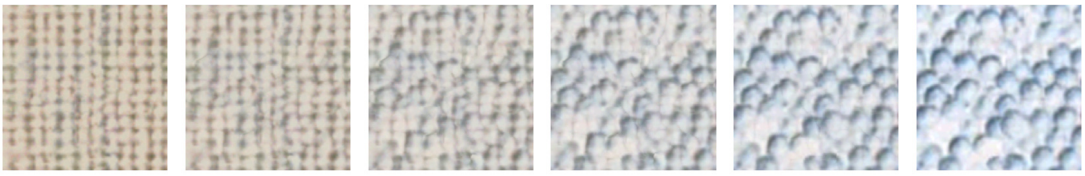
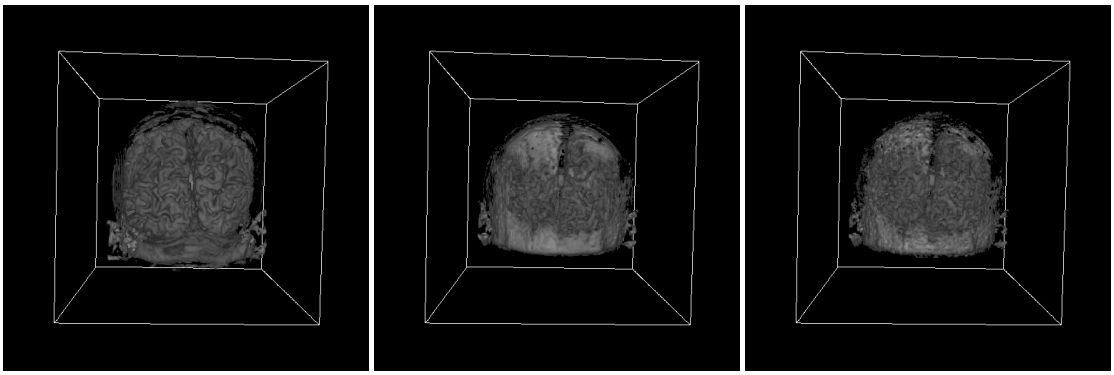


Figure 2.13: Texture metamorphosis sequence is automatically generated from left to right.



(a) input (b) input (c) registered

Figure 2.14: Registration of MRI images of human heads. Volumes are visualized by isosurface rendering. (a) and (b) are the volumes of the heads of two different persons. (c) is the volume (a) registered into (b) by the mapping determined by proposed method.

Chapter 3

Pop-up Light Field

3.1 Introduction

Central to many image-based rendering (IBR) systems is the goal of interpolating accurately between the sample images in order to generate novel views. In IBR, rendering a desired pixel is often equivalent to interpolating intensity values of some input pixels. Such an interpolation, however, depends on the correspondences between the rendered pixel and those pixels from the input sample images. Accurate correspondence between these pixels can be obtained if we have a large number of input images or an accurate geometric model of the scene.

The issue of realistic rendering using dense image samples in IBR has been very well explored in the forms of Light Field [LH96] and Lumigraph [GGSC96]. In light field rendering, a simple focal plane is sufficient to establish accurate correspondence between interpolating pixels. When sampling is sparse, however, conventional light field rendering could cause aliasing because of erroneous correspondence. On the other hand, the Facade system [DTM96] shows that, from only a few photographs, realistic rendering of buildings architecture can be achieved because the reconstructed geometry model provides accurate correspondence for view-dependent texture mapping. A formal analysis of the tradeoff between the number of images and the amount of geometry (in terms of the number of depth layers associated with each pixel) is presented in [CTCS00].

Many IBR systems have proposed different geometric proxies [BBM01b] to alleviate the problem introduced by under-sampled light fields. In practice, however, acquiring an approximate yet "continuous" proxy model is a challenging task. A continuous proxy model is needed for these IBR systems because every desired ray must intersect a point on

the geometric proxy in order to establish the correspondence between interpolating pixels. This explains why most IBR systems today have used a very simple geometric proxy (e.g., a focal plane in a dynamically re-parameterized light field [IMG00]) for a complex scene, but can construct a relatively complex proxy for a single object (e.g., the lion model in Lumigraph [GGSC96] or the car model in unstructured Lumigraph [BBM01b]). The image-based visual hull [MBR+00b] is another geometry proxy that can be constructed and updated in real-time from silhouettes of a single object. If a single global geometric proxy is not sufficient, local geometric proxies can also be used to improve rendering quality, for example, the view-dependent geometry [Rad99] used in impostors [Sch95] and Microfacet Billboarding [YSK+02]. These local geometric proxies are mostly for single objects as well.

The problem to be tackled in this chapter is: Can we use a relatively sparse set of images of a complex scene and still produce photorealistic virtual views free of aliasing? A straightforward approach would be to perform stereo reconstruction or to establish correspondence between all pixels of the input images. The geometric proxy is a depth map for each input image. Unfortunately, state-of-the-art automatic stereo algorithms are inadequate for producing sufficiently accurate depth information for realistic rendering. Typically, the areas around occlusion boundaries [KZ02, KSC01] in the scene give the least desirable results, because it is very hard for stereo algorithms to handle occlusions without prior knowledge of the scene.

3.2 Approach

We approach this problem by suggesting that it is not necessary to reconstruct accurate 3D information for each pixel in the input light field. Our solution is to construct a pop-up light field by segmenting the input sparse light field into multiple coherent layers. The pop-up light field approach differs from other layered modeling and rendering approaches (e.g., [LS97, SGwHS98, BSA98]) in a number of ways. First, the number of layers needed in a pop-up light field is not pre-determined. Rather, it is decided interactively by the user. Second, the user specifies the layer boundaries in key frames. The layer boundaries are then propagated to the remaining frames automatically. Third, our representation is simple. Each layer is represented by a planar surface without the need for per-pixel depth. Fourth, and most importantly, our layers are coherent so that anti-aliased rendering using these coherent layers is achieved. Each coherent layer must have sufficiently small

depth variation so that anti-aliased rendering of the coherent layer itself becomes possible. Moreover, to render each coherent layer with its background layers, not only is accurate layer segmentation required on every image, but also segmentation across all images must be consistent as well.

To segment the layers, the user interface is key. A good user interface can enable the user to intuitively manipulate the structure of a pop-up light field so that virtual views with the desired level of fidelity can be produced. By having a "human-in-the-loop" for pop-up field construction, the user can specify where aliasing occurs in the rendered image. Then, corresponding layers are refined accordingly. More layers are popped up, refined and propagated across all images in the light field until the user is satisfied with the rendering quality (i.e., no aliasing is perceived).

This work was inspired by the real-time 3D model acquisition system of [RHHL02], in which the user specifies areas that need to be modeled depending on the current merged model from multiple scans. Though the goal and methodology are very different, the key concept in our system is similar: our user is in the modeling loop and specifies, through a real-time pop-up light field renderer, where aliasing occurs and how the scene should be further modeled.

Another motivation stems from our frustration of not being able to get accurate per-pixel depth using stereo or other vision techniques. Pop-up light field is an image-based modeling technique that does not rely on accurate 3D depth/surface reconstruction. Rather, it is based on accurate layer extraction/segmentation in the light field images. In a way, we trade a difficult correspondence problem in 3D reconstruction for another equally difficult segmentation problem. However, for a user, it is much easier to specify accurate contours in images than accurate depth for each pixel.

3.3 Pop-up Light Field Representation

3.3.1 An Example

When a light field is under-sampled, conventional light field rendering [LH96] results in aliasing. The top row of Figure 3.1 shows the rendering of a sparse light field with 5x5 Tsukuba images. The top right image is rendered with the 5x5 sparse light field by setting a single focal plane in the scene [IMG00]. Double images can be easily observed on the front objects.

The bottom row of Figure 3.1 shows that anti-aliased rendering can be achieved using

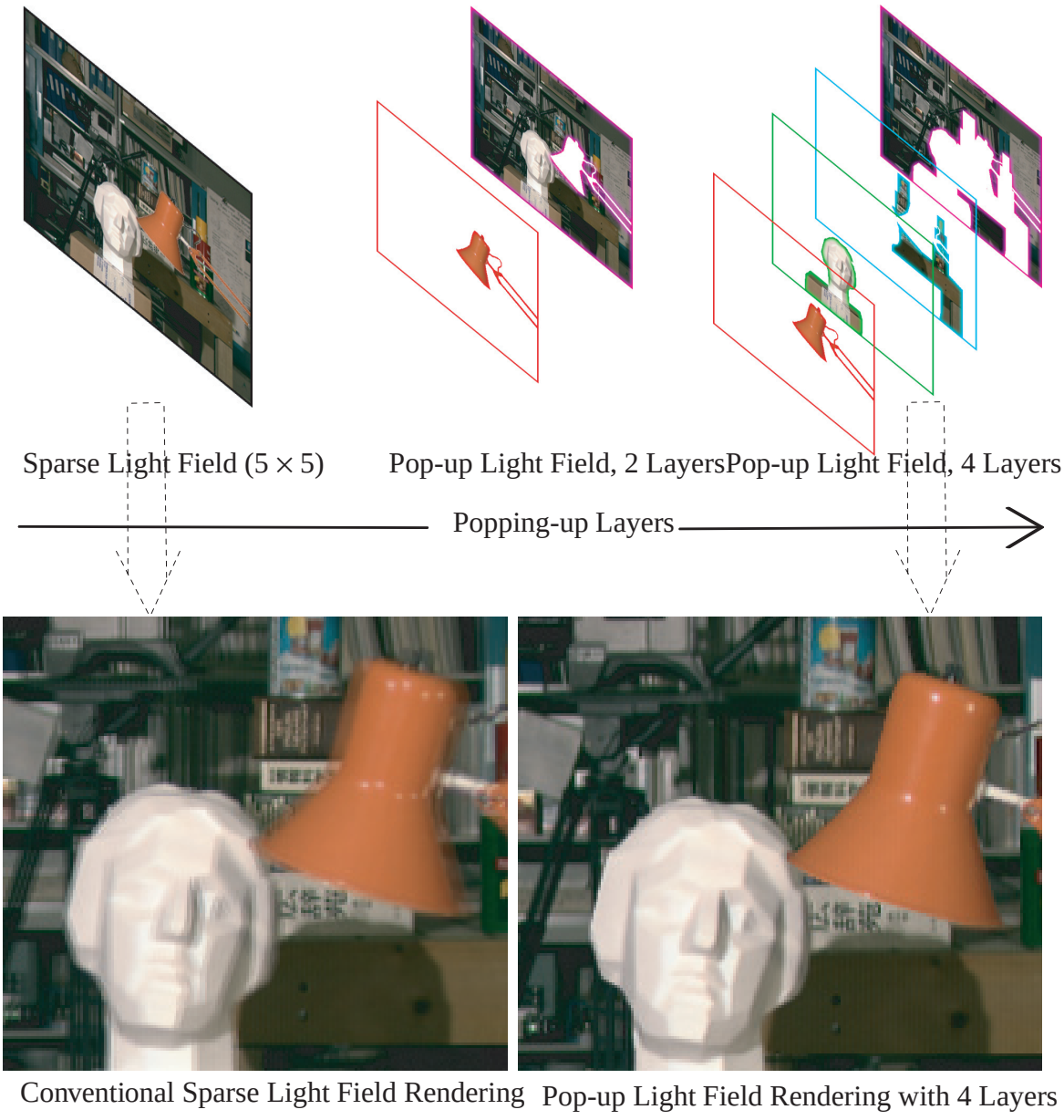


Figure 3.1: An example of rendering with pop-up light fields. Rendering using the 5x5 Tsukuba light field data set is shown in the top left. Aliasing is clearly visible near front objects in the bottom left image because the input light field is sparse. The top row shows that the pop-up light field splits the scene gradually into 4 coherent layers, and achieves anti-aliased rendering as shown in the bottom right image.

four layers, each of which employs a simple planar surface as its geometric proxy. Splitting the scene into multiple layers causes the depth variation in each layer to become much smaller than that in the original sparse light field.

The pop-up light field is represented by a collection of *coherent layers*. A key observation in our pop-up light field representation is that the number of coherent layers that should be modeled or "popped up" depends on the complexity of the scene and how undersampled the input light field is. For a sparser light field, more layers need to be popped up for anti-aliased rendering.

3.3.2 Coherent Layers

We represent a coherent layer L_j by a collection of corresponding layered image regions R_j^i in the light field images I^i. These regions are modeled by a simple geometric proxy without the need for accurate per-pixel depth. For example, a global planar surface (P_j) is used as the geometric proxy for each layer L_j in the example shown in Figure 3.2. To deal with complicated scenes and camera motions, we can also use a local planar surface P_j^i to model the layer in every image i of the light field.

A layer in the pop-up light field is considered to be "coherent" if the layer can be rendered free of aliasing by using a simple planar geometric proxy (global or local). Anti-aliased rendering occurs at two levels when

1. the layer itself is rendered; and

2. the layer is rendered with its background layers.

Therefore, to satisfy the first requirement, the depth variation in each layer must be sufficiently small, as suggested in [CTCS00]. Moreover, the planar surface can be adjusted interactively to achieve the best rendering effect. This effect of moving the focal plane has been shown in [IMG00, CTCS00].

However, to meet the second requirement, accurate layer boundaries must be maintained across all the frames to construct the coherent layers. A natural approach to ensuring segmentation coherence across all frames is to propagate the segmented regions on one or more key frames to all the remaining frames [SGwHS98, ZWGS02]. Sub-pixel precision segmentation may be obtained on the key frames by meticulously zooming in on the images and tracing the boundaries. Propagation from key frames to other frames, however, causes inevitable under-segmentation or over-segmentation of a foreground layer. Typically, over-segmentation of a foreground layer leads to the inclusion of background

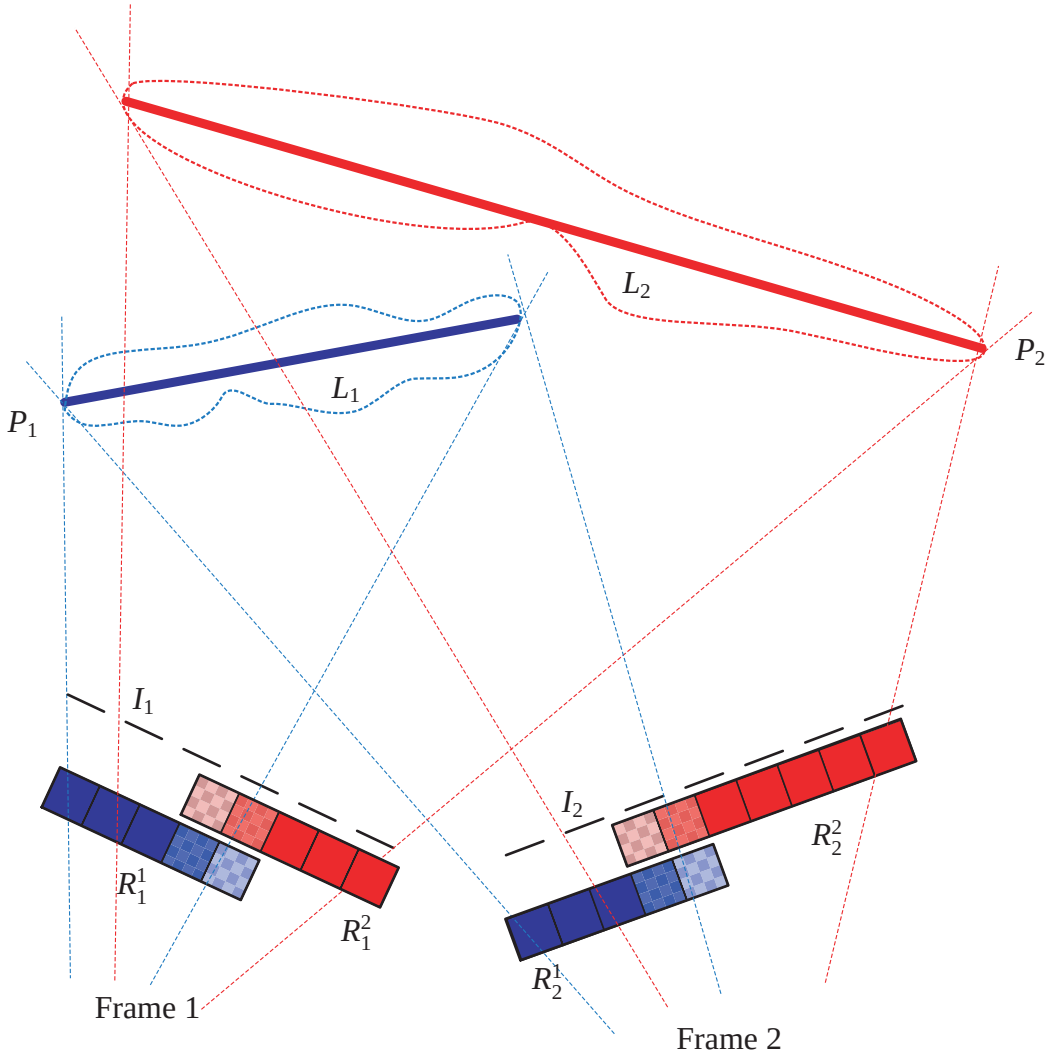


Figure 3.2: A light field (with images I_1 and I_2) can be represented by a set of coherent layers (L_1 and L_2). A coherent layer is a collection of layered images in the light field. For instance, L_1 is represented by a layered image R_1^1 (from I_1) and R_2^1 (from I_2). Each layered image has an alpha matte associated with its boundary. Part of the scene corresponding to each layer (e.g., L_1) is simply modeled as a plane (e.g., P_1).

pixels, thereby introducing ghosting along the occlusion boundaries in the rendered image. A possible example of foreground over-segmentation is exhibited in Figure 4(g) of [SGwHS98] where black pixels on the front object's boundary can be observed. To alleviate the rendering artifacts caused by over-segmentation or under-segmentation of layers, we need to refine the layer boundary with alpha matting [PD84].

Figure 3.2 illustrates coherent layers of a pop-up light field. All the pixels at each coherent layer have consistent depth values (to be exact, within a depth bound), but may have different fractional alpha values along the boundary.

To produce fractional alpha mattes for all the regions in a coherent layer, a straightforward solution is to apply video matting [CCSS01]. The video matting problem is formulated as a maximum a posterior (MAP) estimation as in Bayesian matting [CCSS01],

$$\underset{F,B,\alpha}{\operatorname{argmax}} P(F, B, \alpha|C) \tag{3.1}$$

$$= \underset{F,B,\alpha}{\operatorname{argmax}} L(C|F, B, \alpha) + L(F) + L(B) + L(\alpha) \tag{3.2}$$

where $L(\cdot) = logP(\cdot)$ is log likelihood, C is the observed color for a pixel, and F, B and α are foreground color, background color and alpha value to be estimated, respectively. For color image, C, F and B are vectors in RGB space. In Bayesian matting and video matting, the log likelihood for the alpha $L(\alpha)$ is assumed to be constant so that $L(\alpha)$ is dropped from Equation (3.2).

In video matting, the optical flow is applied to the trimap (the map of foreground, background and uncertain regions), but not to the output matte. The output foreground matte is produced by Bayesian matting on the current frame, based on the propagated trimap. Video matting works well if we simply replay the foreground mattes against a different background. However, these foreground mattes may not have the in-between frame coherence that is needed for generating novel views.

3.3.3 Coherence Matting

We propose a novel approach, called *coherence matting*, to construct the alpha mattes in a coherent layer that have in-between frame coherence. The workflow of our approach is similar to video matting and is illustrated in Figure 3.3. First, the user-specified boundaries are propagated across frames. Second, the uncertain region along the boundary is determined. Third, the under-segmented background regions from multiple images are combined to construct a sufficient background image. Fourth, the alpha matte for the fore-

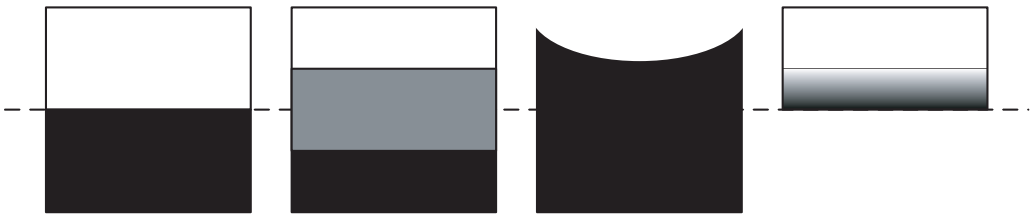


Figure 3.3: Illustration of major steps in coherence matting. (a) The user specifies an approximate segmentation. (b) An uncertain region is added in between foreground and background. (c) A background mosaic is constructed from multiple under-segmented background images. (d) A coherent foreground layer is then constructed using coherent matting.

ground image (in the uncertain region) is estimated. The key to our approach is at the fourth step in Figure 3.3(d) where we introduce a coherent feathering function across the corresponding layer boundaries. Note that, for a given layer, a separate foreground matte is estimated independently for each frame in the light field, and the coherence across frames is maintained by foreground boundary consistency.

$L(B)$ in Equation (3.2) can be dropped since we can explicitly estimate the background (see Section 3.4.4). By incorporating a coherence prior on the alpha channel $L(\alpha)$ across frames, coherence matting can be formulated as

$$L(F, B, \alpha|C) = L(C|F, B, \alpha) + L(F) + L(\alpha) \tag{3.3}$$

where the log likelihood for the alpha $L(\alpha)$ is modelled as:

$$L(\alpha) = -(\alpha - \alpha_0)^2/\sigma_a^2 \tag{3.4}$$

where $\alpha_0 = f(d)$ is a feathering function of d and σ_a^2 is the standard deviation. d is the distance from the pixel to the layer boundary. The feathering function $f(d)$ defines the α value for surrounding pixels of a boundary. In the current implementation, the feathering function is set as $f(d) = (d/w) * 0.5 + 0.5$, where w is feathering width, as illustrated in Figure 3.4.

Assume that the observed color distribution $P(C)$ and sampled foreground color distri-

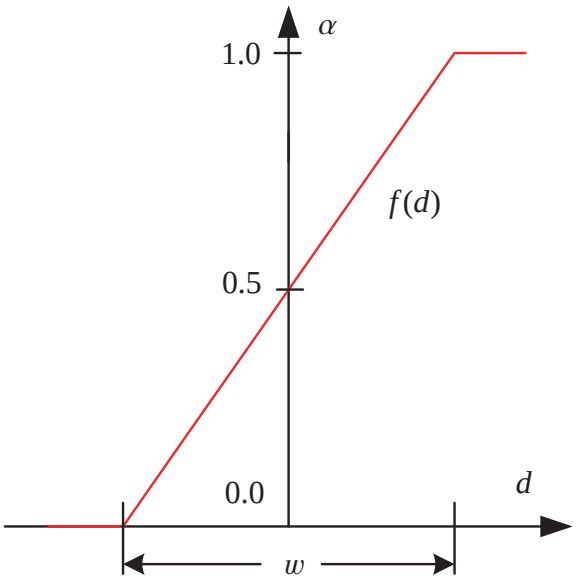


Figure 3.4: Feathering function used in coherence matting

bution $P(F)$ (from a set of neighboring foreground pixels) are all of Gaussian distribution:

$$L(C|F,B,\alpha) = -\|C-\alpha F-(1-\alpha)B\|^2/\sigma_C^2, \quad (3.5)$$

$$L(F) = -(F-\overline{F})^T\Sigma_F^{-1}(F-\overline{F}) \quad (3.6)$$

where σ_C is the standard deviation of the observed color C, $\overline{F}$ is the weighted average of foreground pixels, and Σ_F is the weighted covariance matrix. Taking the partial derivatives of Equation (3.3) with respect to F and α and forcing them equal to zero results in the following equations:

$$F = \frac{\Sigma_F^{-1}\overline{F}+C\alpha/\sigma_C^2-B\alpha(1-\alpha)/\sigma_C^2}{\Sigma_F^{-1}+I\alpha^2/\sigma_C^2} \quad (3.7)$$

$$\alpha = \frac{(C-B)\cdot(F-B)+\alpha_0\cdot\sigma_C^2/\sigma_a^2}{\|F-B\|^2+\sigma_C^2/\sigma_a^2} \quad (3.8)$$

α and F are solved alternatively by using Equation (3.7) and Equation (3.8). Initially, α is set to α_0.

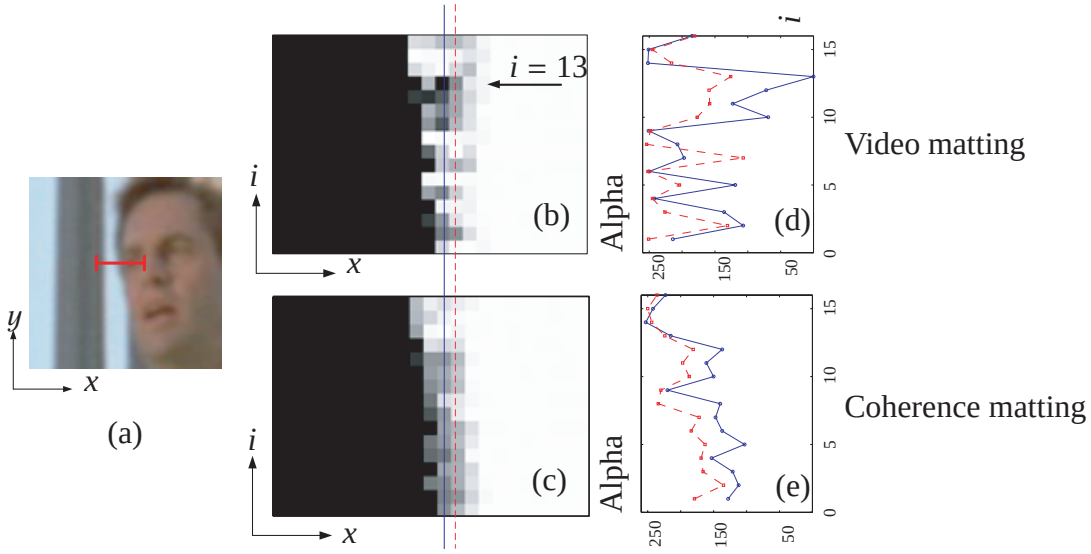


Figure 3.5: Comparison between video matting and coherence matting. (a) is a small window on one frame in the Plaza data (Figure 3.16). (b) and (c) are two alpha epipolar plane images (α-EPI) corresponding to the red line in (a), using the algorithm of video matting and coherence matting, respectively. (d) and (e) are the alpha curves of two adjacent columns, which are marked as blue and red lines in (b) and (c). (d) corresponds to video matting, and shows a large jump at $i = 13$, which causes an accidental transparency within the face. (e) corresponds to coherence matting, which provides a more reasonable result.

3.3.4 Rendering with Coherent Matting

Bayesian matting [CCSS01] and video matting [CAC+02] solve the matting from the equation

$$\alpha = \frac{(C - B) \cdot (F - B)}{||F - B||^2}, \tag{3.9}$$

which works well in general but becomes unstable when $F \approx B$. In comparison, the coherence matting of Equation (3.8) can be solved more stably, because applying the coherence prior on α results in a non-zero denominator. The coherence prior behaves similar to the smoothness constraint commonly used in visual reconstruction (e.g., shape from shading [HB89]).

The spatial inconsistency of the alpha matte from video matting can be observed in Figure 3.5. We plot the alpha epipolar plane image (α-EPI) of a video matting result. Similar to the conventional EPI [BB89], for a short segment of scanline from the Plaza sequence, we stack the alpha values along this segment for all of the 16 frames ((b) and (c)). The alpha values along 2 lines (solid and dotted) in the α-EPI are plotted in (d) and (e). Each line represents the alpha values of the corresponding pixels across 16 frames. A close inspection of (b) around frame $i = 13$(video matting method), shows that the alpha value changes from about 126 to 0, then to 180 (the range of alpha is from 0 to 255), indicating a small part of the face accidently becomes transparent.

The temporal incoherence of the alpha matte from video matting can be more problematic during rendering. The fluctuation of alpha values along both dotted and solid lines will generate incoherent alpha values and thus cause rendering artifacts as we change viewpoints (along axis i). This phenomenon can be more clearly observed in the accompanying videotape. Figure 3.5 (e) shows the same solid and dotted lines with coherent matting results. Both lines have much less fluctuation between neighboring pixels, and appear temporally smoother than their counterparts in Figure 3.5(d).

3.4 Pop-up Light Field Construction

To construct a pop-up light field, we have designed an easy-to-use user interface (UI). The user can easily specify, refine and propagate layer boundaries, and indicate rendering artifacts. More layers can be popped up and refined until the user is satisfied with the rendering quality.

3.4.1 UI Operators

Figure 3.6 summarizes the operations in the pop-up light field construction UI. The key is that a human is in the loop. The user supplies the information needed for layer segmentation, background construction and foreground refinement. By visually inspecting the rendering image from the pop-up light field, the user also indicates where aliasing occurs and thus which layer needs to be further refined. The user input or feedback is automatically propagated across all the frames in the pop-up light field. The four steps of operations in the UI are summarized in the following four sub-sections.

Layer pop-up This step segments layers and specifies their geometries. To start, the user selects a key frame in the input light field, specifies regions that need to be popped up, and assigns the layer's geometry by either a constant depth or a plane equation. This step results in a coarse segmentation represented by a polygon. The polygon region and geometry configuration can be automatically propagated across frames. Layers should be popped up in order of front to back. More details of layer pop-up are shown in Section 3.4.3.

Background construction This step obtains background mosaics that are needed to estimate the alpha mattes of foreground layers. Note that the background mosaic is useful only for the pixels around the foreground boundaries, i.e., in the uncertain region as shown in Figure 3.3. More details of background construction are discussed in Section 3.4.4.

Foreground refinement Based on the constructed background layers, this step refines the alpha matte of the foreground layer by applying the coherence matting algorithm described in Section 3.3.3. Unlike layer pop-up in step 1, foreground refinement in this step should be performed in back-to-front order.

Rendering feedback Any modification to the above steps will update the underlying pop-up light field data. The rendering window will be refreshed with the changes as well. By continuously changing the viewpoint the user can inspect for rendering artifacts. The user can mark any rendering artifacts such as ghosting areas by brushing directly on the rendering window. The corresponding frame and layer will then be selected for further refinement.

3.4.2 UI Design

Figure 3.7 shows the appearance of our UI, including five workspaces where the user interacts with a frame and a layer in the pop-up light field. These workspaces and their functionalities are explained as follows.

Lower middle The user chooses an active frame by clicking on the *frame navigator*, shown at the lower middle part of Figure 3.7. The active frame appears in the *editing frame view*, shown at the upper left in Figure 3.7.

Upper left In the *editing frame view*, the user can create or select an active layer and edit its polygon region. This active layer is displayed in blue polygons with crosses for each editable vertex. The information of the active layer is available in the *layer navigator*, shown at the lower right of Figure 3.7.

Lower right From the *layer navigator*, the user can obtain the active layer's information. The user can select, add, or delete layers in the list. By selecting the layer in the check box, the user can turn on/off a layer's display in the *editing frame view*, *reference frame view* (shown at the upper right of Figure 3.7), and the rendering window. The plane equation of the active layer is displayed and can be modified through keyboard input. Layer equations can also be set through adjusting the rendering quality in the rendering window.

Upper right The *reference frame view* is used to display another frame in the light field. This workspace is useful for a number of operations where correspondences between the reference frame view and the editing frame view need to be considered, such as specifying plane equations.

Lower left To fine tune the polygon location for the active layer, the *boundary monitor* (lower left of Figure 3.7) shows close-up views of multiple frames in the light field. The first row shows the close-up around the moving vertex. The second and third rows show the foreground and background of the active layer composed with a fixed background selected by the user. For instance, using mono fuchsia color in Figure 3.7 as the background makes it easy for the user to observe over-segmentation or under-segmentation of the foreground across multiple frames simultaneously.

Others Not shown in the figure is the rendering window on which the user can render any novel view in real time and can inspect the rendering quality. The user can also

specify the frontal plane's equation for an active layer by sliding the plane depth back and forth until the best rendering quality (i.e., minimum ghosting) is achieved. If the ghosting cannot be completely eliminated at the occlusion boundaries, the layer's polygon must be fine tuned. The user can brush on the ghosting regions, and the system can automatically select the affected frame and layer for modification. The affected layer is front-most and closest to the specified ghosting region.

To specify the slant plane equation for a layer, the user needs to select at least four pairs of corresponding points on the *editing frame view* and the *reference frame view*. The plane equation can be automatically computed and then used for rendering.

Also not shown in the above figure is a dialog box where the user can specify the feathering function. Specifying a feathering curve is useful for the coherence matting algorithm described in Section 3.3.3.

3.4.3 Layer Pop-up

To pop up a layer, the user needs to segment and specify the geometry of the layer for all frames in the light field. In this section, we discuss the operations by which the user interacts with the system and the underlying algorithms.

Layer initialization

We use polygons to represent layer boundaries, since the correspondence between polygons can be maintained well in all frames by the corresponding vertices. The user can specify the layer's boundary with a polygon (e.g., using the polygon lasso tool in Adobe Photoshop) and edit the polygon by dragging the vertices. The editing will be immediately reflected in the *boundary monitor* window and in the rendering window (Section 3.4.2).

First of all, the user needs to inspect the rendering window by changing the viewpoint and decide which region is going to be popped up (usually the front-most non-ghosting object). The user then selects a proper key frame to work with and draws a polygon on the frame.

Then, the user needs to specify the layer's geometry. For a frontal plane, the layer depth is the one that achieves the best rendering quality which can be observed on the rendering window by the user. For a slant plane, the user specifies at least four pairs of corresponding points on at least two frames to estimate the plane equation.

Once the layer geometry is decided, the polygon on the first key frame can be propagated to all other frames by back-projecting its vertices, thereby resulting in a coarse segmentation of the layer on all frames in the light field. All vertices on the key frame are marked as key points. At this stage, the layer has a global geometry which is shared across all the frames. An accurate polygon boundary for layer initialization is not necessary. Because of occlusions and viewpoint changes, propagated polygon boundaries inevitably need to be refined.

Layer refinement

The following aspects need to be considered in layer refinement.

Boundary refinement in a key frame All vertices on any frame can be added, deleted and moved. Once a vertex has been modified, it is marked as a key point. The position of the modified vertex will be propagated across frames at once and the layer region will be updated in several UI workspaces. To adjust a vertex position, the user can observe how well foreground and background colors are separated in the *boundary monitor* window, or how much the ghosting effect is removed in the rendering window.

Boundary propagation across multiple frames For a specific vertex on the layer boundary, if there is a non-key point on frame I_P, we want to interpolate its image coordinate from the corresponding key points in other frames. If there is only one key point in other frames, we compute the coordinate by back projecting the intersection point of layer plane and the viewing ray from key point. Otherwise, we select two or three "neighboring" frames that contain the key points. Then, we compute the coordinate by back projecting the 3D point which has minimal sum of distances to the viewing rays from key points in these frames.

For a 1D camera array, we select the two frames closest to the left and right of the frame I_P that contain key points. For a 2D camera array, we first compute the Delaunay triangulation in the camera plane using all frames containing key points. If frame I_P is in the interior of a triangle, the three frames one the triangle vertices are "neighboring" frames. For example in Figure 3.8, *A*, *B* and *D* are the "neighboring" frames of frame *b*. If frame I_P is in the exterior of all triangles, we select two frames I_0 and I_1 that maximize the angle $\angle I_0 I_P I_1$ in camera plane. For example, *A* and *D* are the "neighboring" frames of frame *a*, as shown in Figure 3.8.

Note that the key points are those that have been modified by the user. They do not necessarily exist on key frames.

Coherence matting It is difficult to accurately describe a layer boundary simply by using polygons. It is hard for the user to manually adjust to sub pixel accuracy a boundary with subtle micro geometry. A pixel is often blended with colors from both foreground and background due to the camera's point spread function. Therefore, we choose not to require the user to specify very accurate sub-pixel boundary positions. Instead, a coherence matting algorithm is applied to further refine the layer boundary. Polygon editing (in a frame and across frames) and coherence matting can be alternatively performed with assistance from the user.

Local geometry When the viewpoint changes significantly, a single planar geometry may not be sufficient to achieve anti-aliased rendering, such as cameras $C1$ and $C2$ in Figure 3.9. Therefore, we introduce a local geometry representation in our system which allows each frame to have its own planar equation $L1$ and $L2$ as illustrated in Figure 3.9.

Using the same UI as in Section 3.4.3, the plane equation can be estimated for each frame. Our system allows the user to specify only a few key frames' geometry, and interpolates the plane equations for frames in between. Similar to the "neighboring" frames selection algorithm in the boundary propagation step, we can select two (for 1D camera array) or three (for 2D camera array) key frames for interpolation. For the frontal plane model, we interpolate the depth of the plane. For the 3D plane model, we interpolate the plane orientation while keeping the intersecting line if using two key frames, or the intersecting point if using three key frames. Once the plane equation has been estimated for each frame, the same rendering algorithm can be applied as in using global geometry.

3.4.4 Constructing the Background

The algorithm of coherence matting in Section 3.3.3 assumes that the background for the uncertain regions (where matting is estimated) is known. A key observation is that, because the uncertain regions are located around foreground boundaries, they can appear only on neighboring frames in the light field where these regions are dis-occluded. Our background reconstruction algorithm fills the disoccluded region using (warped) pixels from neighboring frames.

Once the foreground has been popped up, we obtain the background image by removing the foreground image, as shown in Figure 3.10(a). Moreover, the background boundary is eroded by a few pixels (typically two pixels) before constructing the background mosaic because a possible under-segmentation of the foreground may leave some mixed foreground pixels on the background around boundaries.

An automatic algorithm has been designed to construct the background, by warping the neighboring images to fill the holes using the background layer's geometry. This method works well if the background is well approximated by plane, e.g., in Figure 3.1.

However, when the background contains objects with relatively large depth variation, we need to further subdivide the background layer into sub layers, each of which can be represented as one plane. As shown in Figure 3.10(a), a background layer is segmented manually into four sub layers using polygons. This time, the location of the polygon is not critical. Instead, the criterion here is to group the background boundaries into a better planar approximation.

The sub layers are propagated from the key frame, where the user specifies the division, to all other frames using the existing background layer geometry. This propagation requires less accuracy as long as it covers the same group of boundaries. The relative motion of the sub layer across frames is estimated hierarchically, starting from translation to affine, and from affine to perspective transform [SS97]. Only the pixels visible in both frames are used to estimate parameters. Figure 3.10(b) shows the resulting mosaic. Note that we need not create a hole-free mosaic, as a few pixels surrounding the occlusion boundaries are adequate for coherence matting.

3.5 Real-time Rendering of Pop-up Light Field

An integral part of our UI is the real-time pop-up light field renderer which provides the user instant feedback on the rendering quality. Based on previous light field and Lumigraph rendering systems [GGSC96, IMG00, BBM01b], we have developed a real-time pop-up light field renderer. Our rendering algorithm includes three steps: (1) splitting a light field into layers, (2) rendering layers in back-to-front order, and (3) combining the layers.

3.5.1 Data Structure

The data structure used in our rendering algorithm is shown below.

```
struct PopupLightField {
    Array<CameraParameter> cameras;
    Array<Layer> layers;
};
struct Layer {
    Array<Plane> equations;
    Array<Image> images;
};
struct Image {
    BoundingBox box;
    Array2D<RGBA> pixels;
};
```

The pop-up light field keeps the camera parameters associated with all the input frames. Each layer in the pop-up light field has corresponding layered images, one for each frame. Each layered image has a corresponding plane equation, so as to represent the local geometry. If global geometry is applied to a layer, all equations are the same for images in this layer.

Since these corresponding layered images vary their shapes in different views, they are stored as an array of images on each layer. Layers can be overlapping in the pop-up light field and each layered image is modified independently by mosaicing and coherent matting. Therefore it is necessary to keep both color and opacity of images for each layer separately. Each layered image is stored as an RGBA texture image of the foreground colors with its opacity map, and a bounding box as well. The opacity (alpha value) of the pixel is zero when this pixel is out of the foreground.

3.5.2 Layered Rendering Algorithm

The scene is rendered layer by layer using texture-mapped triangles in back-to-front order. Then the layers are sequentially combined by alpha blending. Our rendering scheme is based on [HKP+99, BBM01b], but is extended to multiple layers. The pseudo code of our rendering algorithm is shown below.

After initializing a frame buffer, we generate a set of triangular polygons on which the original images are blended and drawn. We first project the camera positions onto the image plane and triangulate these projection points together with the image plane's four corner points into a set of triangles.

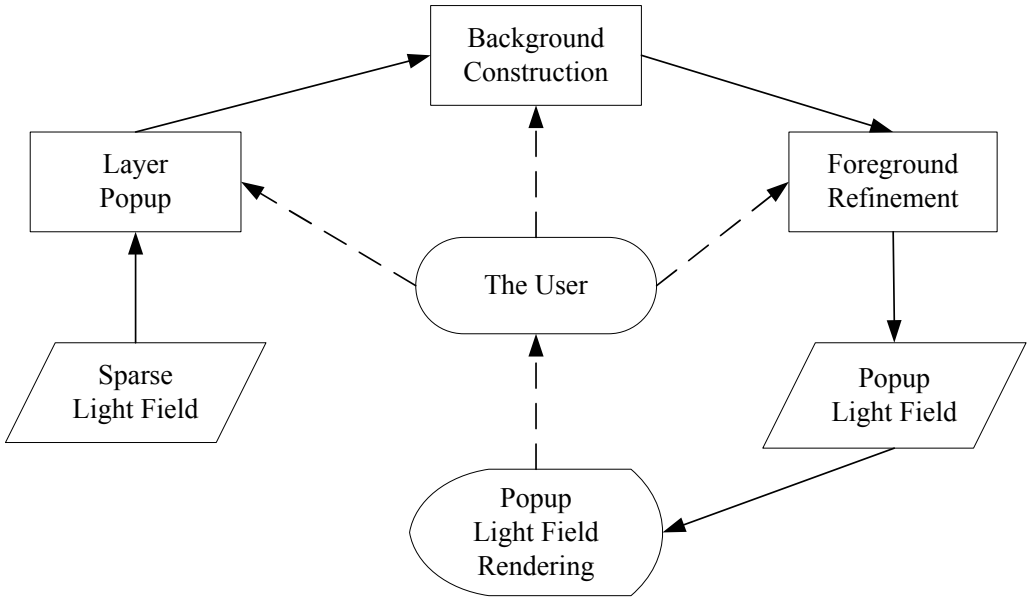


Figure 3.6: Flowchart of pop-up light field construction UI

input: CoherentLayers L
local: RenderingPrimitives T
1: ClearFrameBuffer()
2: $T \leftarrow$ CreateRenderingPrimitives()
3: **for all** layers $l \in L$ from back to front **do**
4: **for all** triangles $\triangle \in T$ **do**
5: SetupProjectiveTextureMapping($\triangle$)
6: Render($\triangle$)
7: BlendToFrameBuffer($\triangle$)
8: **end for**
9: **end for**

Algorithm 3.1: Popup light field rendering

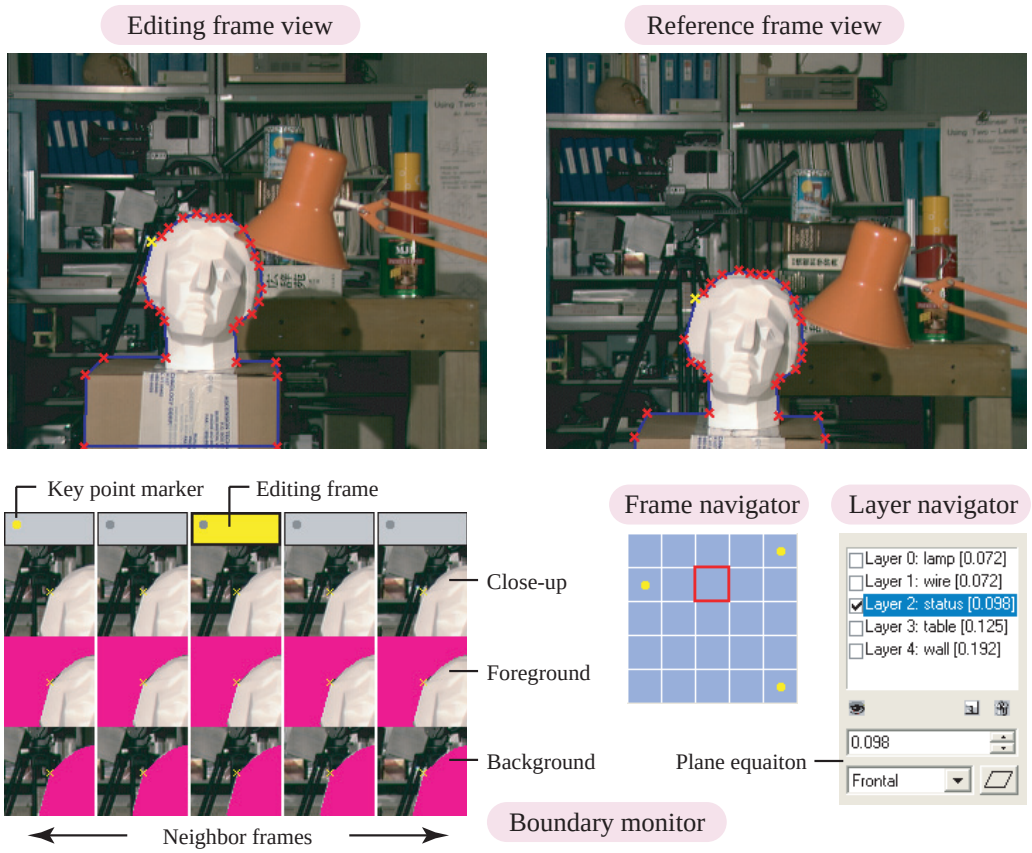


Figure 3.7: User interface for Pop-up light field construction

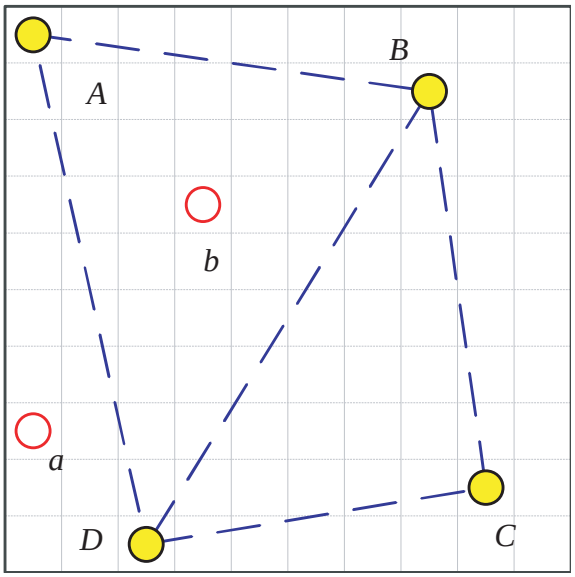


Figure 3.8: "Neighboring" frames selection for 2D camera array. The solid (yellow) dots are frames including key point and hollow (red) dots are frames to be interpolated.

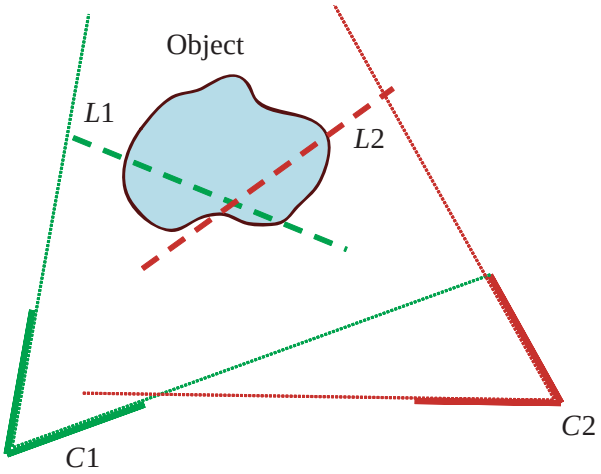
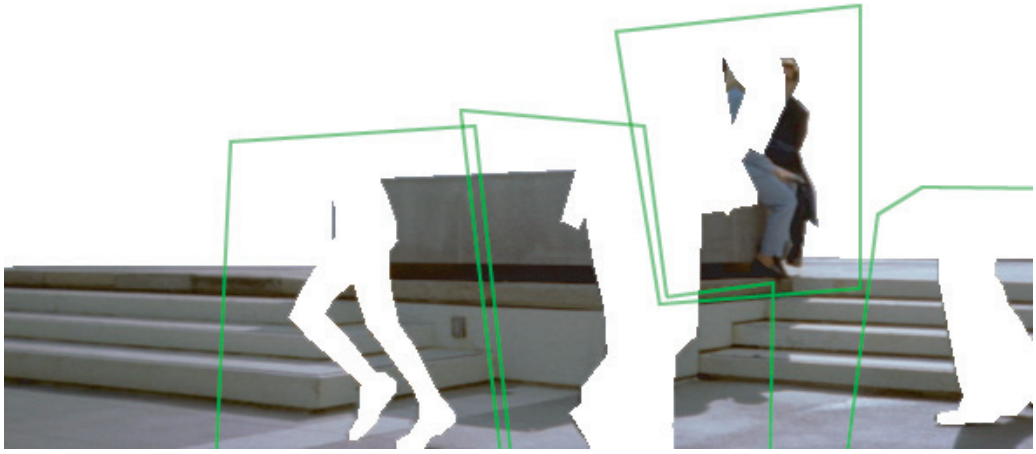


Figure 3.9: Local geometry

(a)



(b)

Figure 3.10: (a) The background mosaic operator uses the polygon lasso operator to segment the layer into regions. (b) The resulting background mosaic fills in many missing pixels in (a). Although (b) still has many missing pixels, it is enough for coherence matting of the foreground.

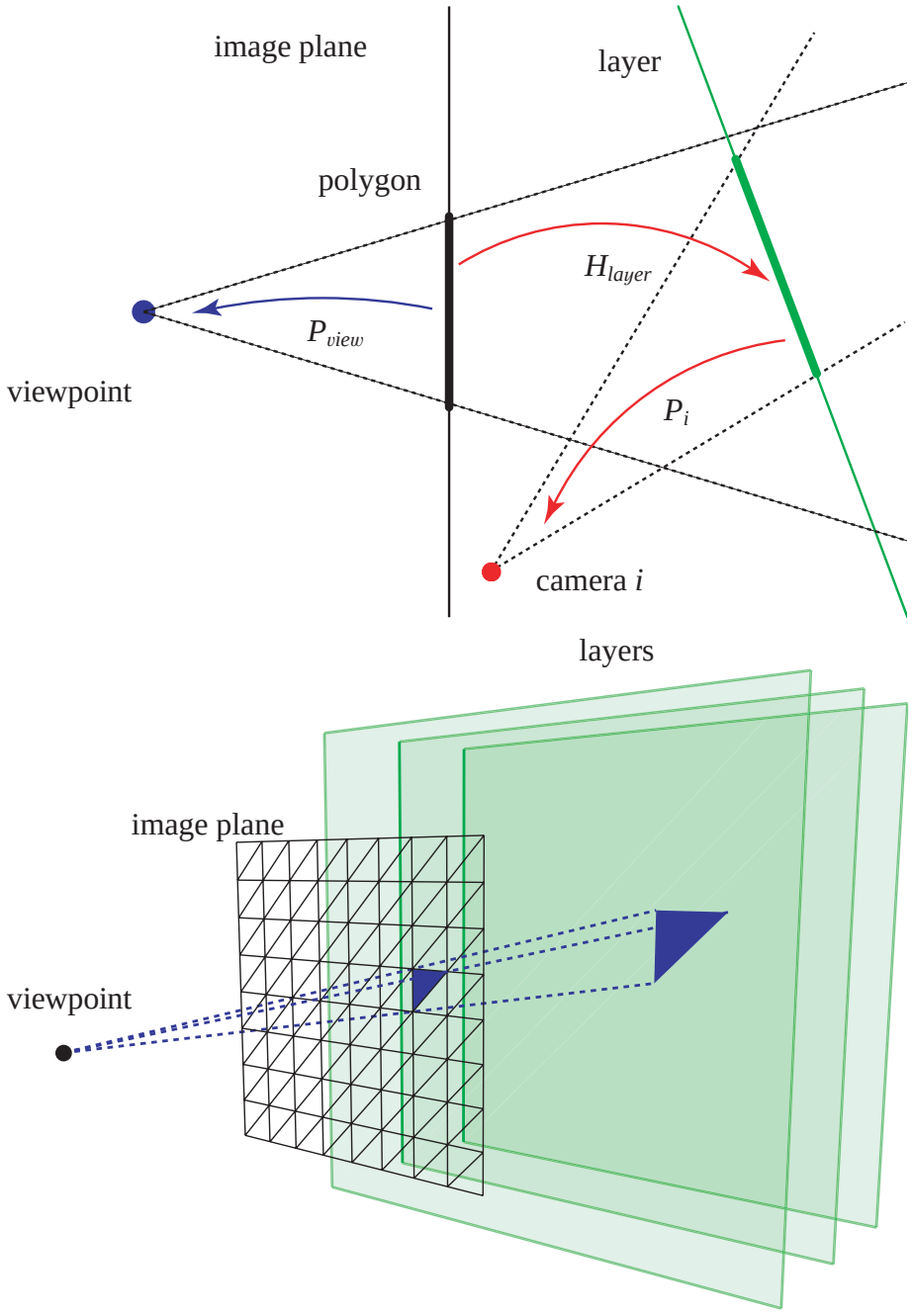


Figure 3.11: Setup of projective texture mapping

Then we assign a trio of texture images $\{I_i\}_{i=1}^3$ to each triangle, which are blended across the triangle when rendering. The blending ratio $\{w_i^k\}_{k=1}^3 (0 \leq w_i^k \leq 1, \sum_{k=1}^3 w_i^k = 1)$ for the three images is also assigned to each of the three vertices, and linearly interpolated across the triangle. The exact blending ratio based on ray angles is not necessarily distributed linearly on the screen. If the size of a triangle is not small enough with respect to the screen size, we subdivide the triangle into four triangles iteratively. On the vertex which is the projection of a I_i's camera, the blending ratio w_i^k is calculated by using the following equation.

$$w_i^k = 1 \quad \text{if camera } i \text{ is projected onto the } k\text{-th vertex} \tag{3.10}$$

$$= 0 \quad \text{otherwise} \tag{3.11}$$

For the vertex which is not the projection of a camera, the weights are calculated using the angle between the ray through the camera and the ray through the vertex [BBM01b].

Then, each layer is rendered by blending texture images $\{I_i\}$ using blending ratios $\{w_i^k\}$. At a point other than the vertices on the triangle, the blending ratios $\{\tilde{v}_i\}$ are calculated by interpolating $\{w_i^k\}_{k=1}^3$. Using $\{I_i\}$ and $\{\tilde{v}_i\}$, the pixels on the triangle are drawn in the color $\sum_{i=1}^3 \tilde{v}_i I_i$.

The texture images are mapped onto each triangle projectively as illustrated in Figure 3.11. Let P_{view} be the projection matrix for the rendering camera (to produce the novel view), P_i be the projection matrix for the camera corresponding to I_i, and H_{layer} be a planar homography from the triangle to the layer plane. Then, the texture image I_i is mapped onto the triangle using a projection matrix $P_i H_{layer}$.

3.5.3 Hardware Implementation

Light field rendering can be accelerated by using graphics hardware. Although several hardware-accelerated light field rendering approaches have been proposed [GGSC96, IMG00, HKP+99, BBM01b], we cannot use them directly for pop-up light field rendering. In these previous approaches, texture images are blended by multi-pass rendering of triangles and alpha-blending them in the frame buffer. In a layered rendering algorithm, we must alpha-blend layers using alpha values assigned in texture images, which means that each layer must be rendered onto the frame buffer in a single pass.

One straightforward way is to copy the frame buffer into memory and composite each layer after rendering. We have implemented this method, and found that it is too slow for

Table 3.1: Performance of rendering

Data	Resolution (w × h × #cameras)	Size of Original LF (MB)	Size of Pop-up LF (MB/#layers)	FPS
Tsukuba	$384 \times 288 \times 25$	8.3	19.4 / 5	62.5
Plaza	$640 \times 486 \times 16$	14.9	38.3 / 16	58.8
Pokemon	$512 \times 384 \times 81$	47.8	117.8 / 5	31.3
Fur	$1136 \times 852 \times 23$	66.8	140.4 / 5	46.9
Statuette	$1136 \times 852 \times 43$	124.8	161.2 / 4	37.1

interactive usage. Instead, we have developed a single pass rendering method that uses multitexture mapping and programmable texture blending which are available on modern graphics hardware.

In order to blend all textures on a single triangle, we first bind three different textures assigned to each triangle, then assign three blending ratios $\{w_1, w_2, w_3\}$ as the primary color { R,G,B } on each vertex. The primary color is smoothly interpolated on the triangle. Hence the interpolated blending ratios $\{\tilde{v}_i\}$ are obtained simply by referring to the primary color at an arbitrary point on the triangle. Then the texture images on the triangle can be blended using the blending equation programmed in the pixel shader in the graphics hardware.

The layers can be composed simply by alpha-blending each triangle on the frame buffer when it is rendered because the triangles are arranged without overlap in a layer and each triangle is drawn in a single pass.

Our rendering system has been implemented using OpenGL and its extensions for multi-texturing and per-pixel shading, and tested on a PC (CPU 660 MHz, memory 768 MB) equipped with an NVidia GeForce4 or ATI Radeon9700 graphics card with 128MB of graphics memory. The performance of rendering is shown in Table 3.1.

3.6 Experimental Results

We have constructed several pop-up light fields from several real scenes. The Tsukuba data set and the Plaza sequence are courtesy of Prof. Ohta of the University of Tsukuba, and Dayton Taylor, respectively. Pokemon data is captured by a computer-controlled vertical X-Y table shown in Figure 3.12. Data sets of Statuette (with unstructured camera motion) and Furry rabbit (with the camera moving along a line) are captured by a Canon G2 Digital

Camera.

As shown in Table 3.1, rendering of all the pop-up light fields can be done in real-time (with a frame rate greater than 30). Table 3.2 summarizes the amount of work required to construct these pop-up light fields. For most scenes in our experiments, it took a couple of hours for a graphics graduate student in our lab to interactively model the pop-up light field. It, however, took the student 5 hours to construct the pop-up light field from the Plaza sequence where 16 layers are segmented.

Tsukuba Some rendering results of the Tsukuba 5×5 light field are shown in Figure 3.1. It is demonstrated that, with 4 layers, anti-aliased rendering can be achieved. In our experiment, we have found that the same rendering quality can be achieved with 7 layers if the light field is down-sampled to 3×3.

Pokemon Figure 3.13 again demonstrates the progressive improvement of visual quality when more layers are popped up. With 5 layers, anti-aliased rendering of pop-up light fields (Pokemon 9×9) is achieved. The four layers that model the three toys and the background use frontal-parallel planes while the table plane is slanted.

Statuette For complicated scenes, instead of using a global planar surface defined in the world coordinate system, local geometry should be used. Figure 3.14 shows the rendering result from a sequence of 42 images taken with unstructured camera motion. If a global planar surface is set as a frontal-parallel plane in the frame (Figure 3.14(a)), rendering at a very different viewpoint will have noticeable artifacts, as shown in Figure 3.14(b). Figure 3.14(c) shows a good rendering result using view-dependent geometry. Specifically, we have changed the plane orientations for different views.

Furry rabbit To show the efficacy of our coherence matting methodology, we use a sparse light field that captures 23 images of a Furry rabbit with the camera path along a line. Figure 3.15 compares the results with video matting and with coherence matting. The zoomed up views of the left ear demonstrate that coherence matting obtains a more consistent matte than video matting does. We refer reviewers to the accompanying video for the rendering results using video matting and coherence matting.

Plaza Figure 3.16 shows an aliasing-free novel view we rendered using the pop-up light field constructed from the Plaza sequence, which is a collection of only 16 images. The

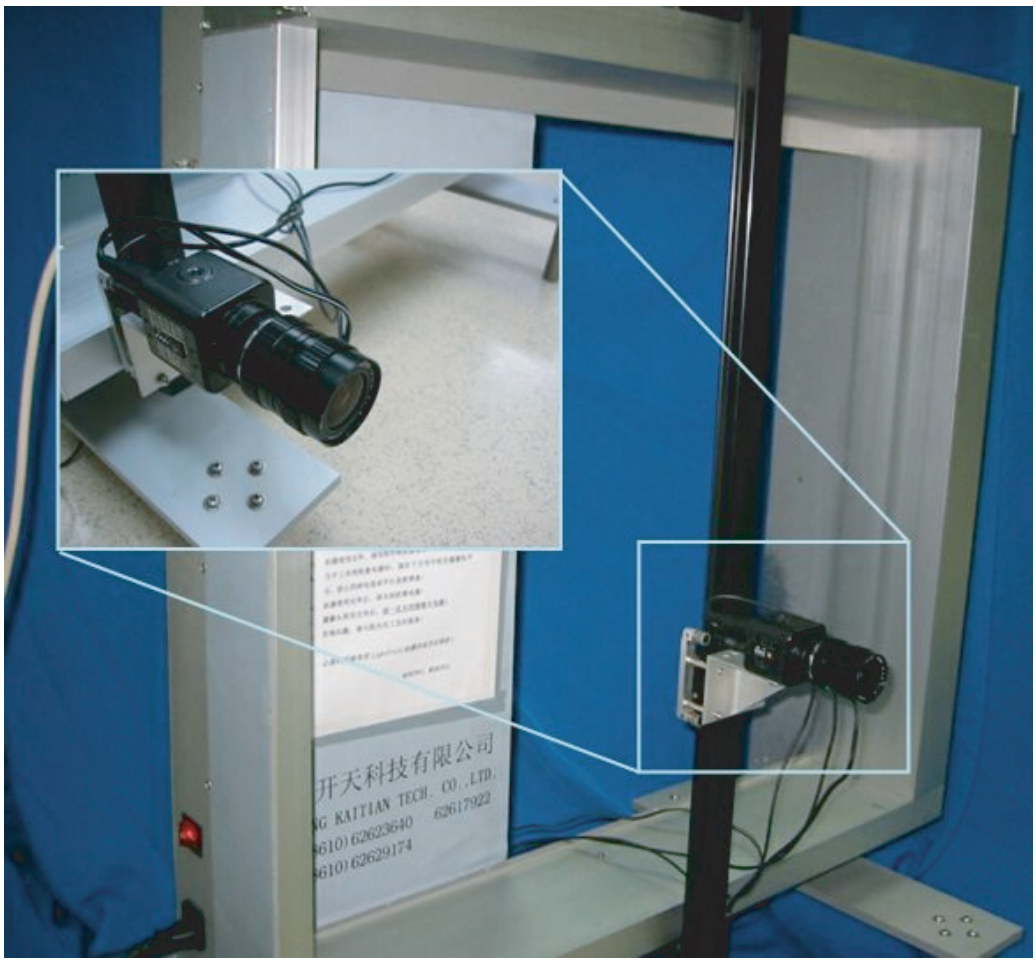


Figure 3.12: System setup for capturing light field. The system is composed of one video camera with 640×480 pixel resolution; the camera is mounted on a computer-controlled support which is enabled to slide on a plane with two degrees of freedom.

one focal plane in the front | one focal plane at the back | 5 layers are popped up

Figure 3.13: Results on Pokemon 9×9: comparison of conventional light field rendering and pop-up light field rendering.

(a) (b) (c)

Figure 3.14: Results on sparse images taken from unstructured camera positions. (a) The global planar surface is set as a frontal-parallel plane in this view, (b) rendering result from another view with the global plane, (c) rendering result from the same view of (b) with local geometry.

Figure 3.15: Comparison of results on a Furry rabbit, with video matting (middle image) and with coherence matting (right image). The alpha matte from coherent matting is smoother that that from video matting in the rendering image. This effect can be more clearly observed in the accompanying video.

Table 3.2: User Interaction

Data	Tsukuba	Plaza	Pokemon	Fur	Statuette
# frame	25	16	81	23	43
# layer	4	16	5	5	4
points/frame	129	379	90	167	47
key points/frame	6.3	62.4	8.6	16.1	12.3
Time (hours)	≈ 0.5	≈ 4	≈ 1	≈ 1	≈ 1.5

sequence was captured by a series of "time-frozen cameras" arranged along a line or curve. Because the scene is very complex, stereo reconstruction is very difficult. Note that nearly perfect matting is achieved for the papers floating in the air. The boundaries for the foreground characters are visually acceptable, made possible mainly by the coherent layers produced by coherence matting.

Figure 3.16: Result of pop-up light field rendering of the Plaza sequence rendered from a novel viewpoint (in the position midway between the 11th and 12th frames). The input consists of only 16 images. We have used 16 layers to model the pop-up light field.

Chapter 4

Microfacet Billboarding

4.1 Introduction

Recently, several researchers [LPC+00, ISN+00] have developed methods for modeling real-world objects based on a measured data set obtained by high-quality optic sensors. However, they are mainly concerned with surface reflectance models and rigid geometry rather than with intricate and soft geometry, such as a cluster of leaves or fur. A natural scene usually contains many objects which have intricate shapes or soft geometry compared with the sensor resolution. It is significantly difficult to render these objects because it is hard to model their complete geometry. As a result, impractically large amounts of view images are required to achieve realistic rendering of real-world scenes.

Consider the case of capturing fur; because the size of individual fur hair is thinner than 0.1mm, no photometric-based scanners such as laser range finders or digital cameras used in light stripe range finders, can capture every hair of the fur. It is also difficult to capture a tree which has clusters of leaves. Since leaves are cluttered, much of their surfaces appear as occluded areas from a scan; thus, many scans are necessary to cover the whole tree. Moreover, it may be almost impossible to capture the interior part of a cluster of leaves. It is not reasonably cost-effective to acquire the complete geometry of a cluttered object.

In this chapter, we deal with the issue of how to synthesize photo-realistic virtual views of objects, especially when their geometry cannot be completely acquired. Hereafter, we mainly consider the scenario in which the geometric model is created from a set of range images acquired through scanning by a laser range finder, while the view-images are taken by digital cameras. We also assume that the geometric model and view-images are aligned in three-dimensional space. Although the use of laser range finders is the best choice

among various methods in terms of the accuracy and the robustness, the obtained range images are supposed to have a considerable amount of noise, our method is designed to synthesize virtual views mainly from reference images with the aid of roughly approximated geometry.

The central idea of our proposed method is based on the following observation: while objects with intricate and soft geometry cannot be acquired accurately by the conventionally used methods of modeling, such as stereo reconstruction [Fau93], visual hull modeling [Lau94, MPN+02] or scanning by laser range finders, they can usually be imaged clearly by using charge-coupled devices (CCD) sensors; such images can be efficiently used to increase the reality of images synthesized by image-based rendering.

The problem to be solved in this rendering framework can be separated into three parts. Among them are: how to integrate and represent the noisy geometric model, how to reduce the visual artifacts caused by the geometrical noise, and how to reduce the visual artifacts caused by limited resolution of concerning optic sensors.

The first difficulty is tackled by introducing a two-level representation of object geometry, that is, "global" and "local" geometry. The global geometry is a volumetric data structure created from acquired geometric models to approximate the object shape consistently between views. The detail of the geometric model is then modeled as a set of local geometry represented as a view-dependent depth map. This view-dependent representation enables the system to solve the second issue by applying a technique of image processing to each reference view-image. In order to overcome the third difficulty, an opacity map is extracted for each reference image and used to increase the photo-reality in the synthetic images.

The remainder of this chapter is organized as follows. In Section 4.2, we first explain the basic framework for the modeling and rendering of objects using view-dependent geometry and view-dependent texture mapping. In Section 4.3, we then propose a novel method for automatically extracting the opacity maps that are required in the rendering. After describing the details of implementation in Section 4.4, the results of rendering for various data sets are presented in Section 4.5

4.2 Modeling and Rendering by Microfacet

In this section, the basic framework for the modeling and rendering of real-world objects is proposed. For modeling, a volumetric data structure is created from given range images

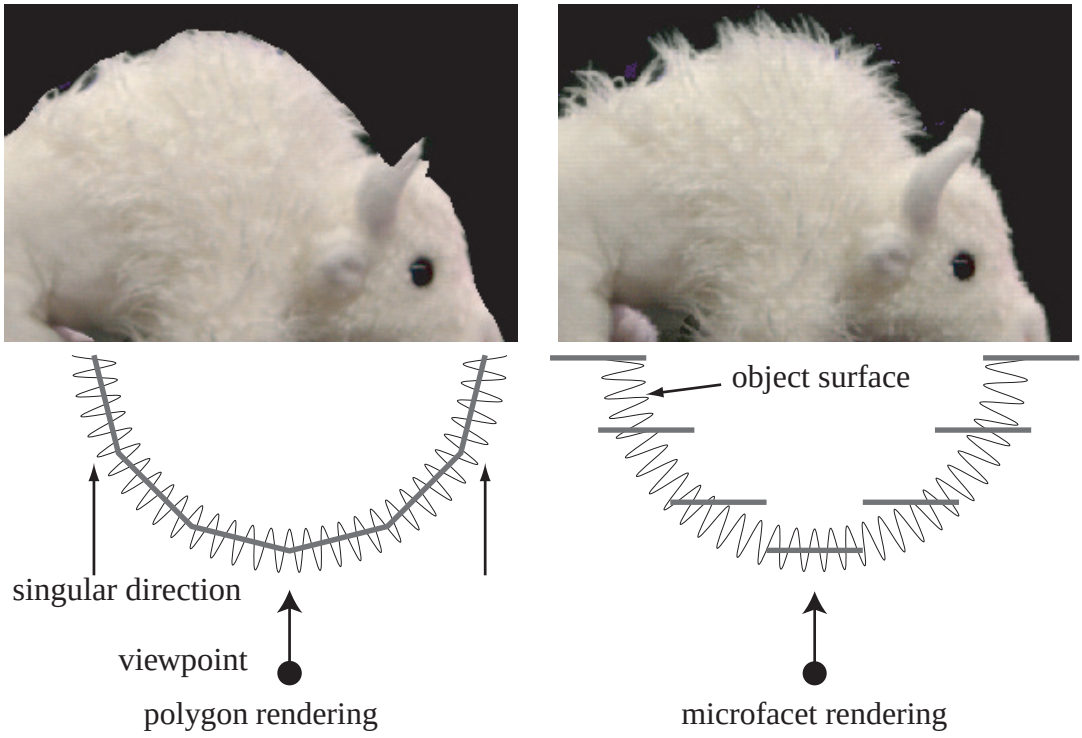


Figure 4.1: Concept of microfacet billboarding. (left) It is difficult to use existing methods which use such fixed geometric primitives as polygons to represent intricate geometry, particularly on occluding boundaries. (right) The facets used in our proposed method are kept perpendicular to the direction of view. When viewed from another point, the facets are again perpendicular.

in order to represent "global" geometry which approximates the object shape consistently between views. The detail of the geometric model is then modeled as a set of "local" geometry in the form of a view-dependent depth map for each view. For rendering, we combine a view-dependent geometry and a view-dependent texture technique to synthesize a realistic image in real time using graphics hardware accelerators. The view-dependent model is represented by a set of discontinuous primitives which we call *microfacet*s, and rendered by the method referred as *Microfacet Billboarding*.

The modeling process may take some time; hence, it is assumed to not be accomplished in real time. On the other hand, once the model has been prepared, our proposed method can render the object in real time by taking advantage of acceleration by graphics hardware. The details of the processes are described in the following sections.

The concept of our method is shown in Figure 4.1. The surface of the object is approximated by a set of *microfacet*s onto which the texture images of the object are mapped. All microfacets are aligned perpendicular to the viewing direction even when the viewpoint or the viewing direction[1] changes. The texture image mapped to each microfacet is generated from acquired view images, estimated alpha matte and view-dependent depth map.

4.2.1 Modeling by Discrete Geometry

output: GlobalGeometricModel M
output: LocalGeometricModels $\{G_v\}$
output: ColorImages $\{C_v\}$
output: AlphaMattes $\{A_v\}$
local: TemporaryGeometricModel M'
1: $M' \leftarrow$ AcquireGeometry()
2: $\{C_v\} \leftarrow$ AcquireColorImage()
3: $M \leftarrow$ GenerateGlobalGeometry(M')
4: **for all** viewpoints v of the images $\{C_v\}$ **do**
5: $G_v \leftarrow$ GenerateLocalGeometry(M, v)
6: $A_v \leftarrow$ EstimateAlphaMatte(G_v, C_v)
7: $G_v \leftarrow$ RefineLocalGeometry(G_v, A_v)
8: **end for**

Algorithm 4.1: Microfacet modeling

[1] In the rest of this thesis, the term *viewpoint* implies both the point and the direction of the viewer.

The outline of the modeling process is shown in Algorithm 4.1. First the geometric model and color images of the object are acquired. The geometric model is represented as a volume in the resolution where the volume is consistent for each view. This volume is referred to as global geometry of the object. The global geometry is then projected to color images so that the depth map is assigned for each color image. Since this initial depth map is supposed to be noisy, it is refined with the aid of the alpha matte extracted from the corresponding color image.

Acquisition of Geometry and Color Images

First, the geometry and color images of the object are acquired. As described in Chapter 1, one of the most important issues in the process of image synthesis is how to model the scene without time-consuming manual operations by the user. In order to create both geometric and photometric models automatically, we designed the system shown in Figure 4.2. The system is composed of five commercial digital cameras in 1024× 768 pixel resolutions; six Halogen lamps with parabolic reflectors which approximate distant lights at infinity; one laser range finder (Minolta VIVID 900); and one rotating turntable placed at the center. The cameras and lamps are mounted on arms with equal spaces along the elevation angle of the hemisphere. The arm mounting light rotates along the axis of the rotating turntable, while the arm mounting cameras is fixed. The digital cameras, Halogen lamps, the arm for lights, rotating table, and laser range finder are all computer-controlled.

For geometric modeling, the camera poses of partial geometric models are determined. This process can be done automatically after the system is calibrated. We used the range images measured by the scanner in this system since our first objective in starting this research was realistic rendering of intricately shaped objects by making the best use of incomplete geometry acquired using a laser range scanner. However, our method is also applicable to models obtained by other methods of modeling, such as stereo- or visual hull modeling. For photometric modeling, we take a set of color images of the object from several viewpoints. It is also assumed that the camera poses are calibrated; hence, the color images are already aligned with the geometric model.

Generation of Global Geometry

Since the obtained geometric model is supposed to have a considerable amount of noise, it is necessary to process the model so that it is consistent between views. Thus, the geometric model is approximated as a single static model, which we call the *global model*.

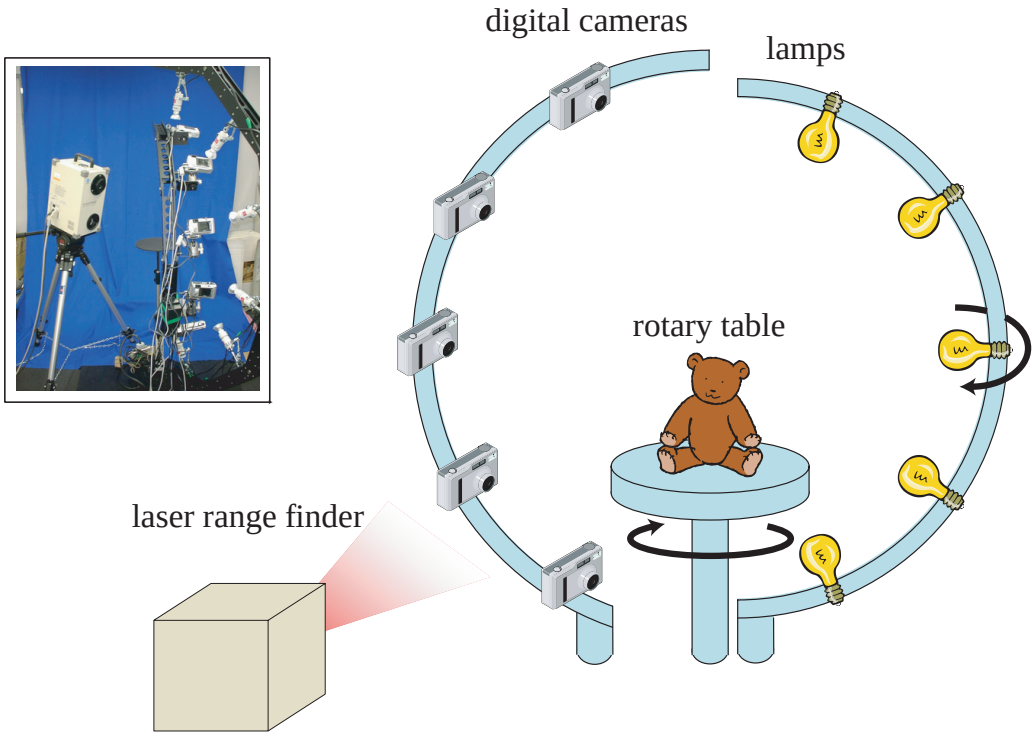


Figure 4.2: System setup for geometric and photometric modeling. The system is composed of five commercial digital cameras in 1024× 768 pixel resolutions, six Halogen lamps with parabolic reflectors, which approximate distant lights at infinity, one laser range finder (Minolta VIVID 900), and one rotating turntable placed at the center. The cameras and lamps are mounted on arms with equal spaces along the elevation angle of the hemisphere. The arm mounting light rotates along the axis of the rotating turntable, while the arm mounting cameras is fixed. The digital cameras, Halogen lamps, the arm for lights, rotating table, laser range finder are all computer-controlled. In this figure, a blue screen is used for robust alpha matting, which is optional.

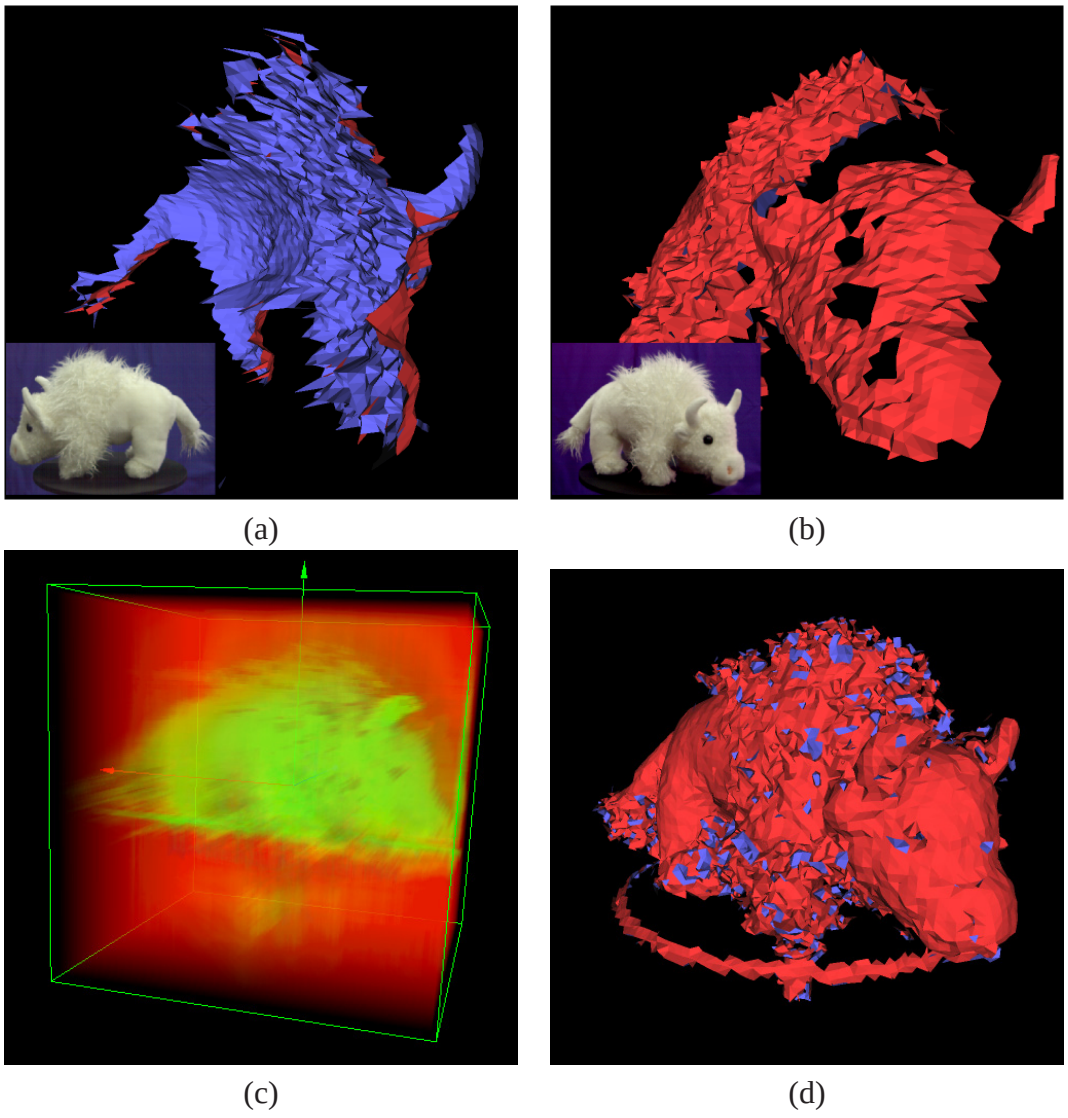


Figure 4.3: Example of the geometric integration for intricately-shaped objects. 36 range images are captured around a stuffed toy using the system shown in Figure 4.2. (a) and (b) are parts of the range images and corresponding view images. The set of range images is converted to a volumetric representation using signed distance transformation and integrated into a single volume by a consensus surface algorithm [WSI98], which is designed for merging noisy range images. (c) is the volume visualized by the technique of volume rending with red and yellow colors for the voxels outside and inside the surface, respectively. By applying a polygonization method such as Marching Cubes [LC87] to the volume, the polygonized model of the object is obtained as shown in (d). In the result of polygon rendering, red and blue polygons indicate front and back surface. Since the object geometry is highly intricate, most of the detailed features on the surface cannot be modeled correctly.

One common method of generating a global model is to integrate range images taking a consensus between the partial geometry [WSI98] with the aid of volumetric representation. This method works effectively to reject outliers in the data set if every part of the object surface is covered by a sufficient number of range images. One of the potential problems with this method is that a large number of range images is required to create a reliable geometric model when the acquired geometry is highly unreliable, since the accuracy of the method depends on the statistical stability of given data. Unfortunately, it is practically impossible to acquire such intricately-shaped geometry as fur into a single model owing to the limited resolution of optic sensors, as shown in Figure 4.3.

input: TemporaryGeometricModel M'
output: GlobalGeometricModel M
local: VolumeResolution L
local: MaxVolumeResolution L_{max} : pre-defined
local: ConsistencyRatio τ : pre-defined

```
 1: L ← L_max
 2: repeat
 3:    M ← SampleToVolume(M, L)
 4:    L ← L − 1
 5:    N_m ← number of voxels in M
 6:    w ← size of voxels in M
 7:    N_i = 0
 8:    for all voxel v in M do
 9:       if not Consistent(v) then
10:          N_i ← N_i + 1
11:       end if
12:    end for
13: until N_i / N_m ≤ τ
```

Algorithm 4.2: Generation of global geometry

Instead of integrating the partial geometric models, we estimate the maximum resolution of the geometric model in which the data is reliable. Similar to the first approach, the acquired geometric models are resampled into a volume using the *signed distance transformation*, that is, each voxel is assigned the distance to the closest surface with a positive sign when the voxel is in front of the surface; otherwise with a negative sign. Then the

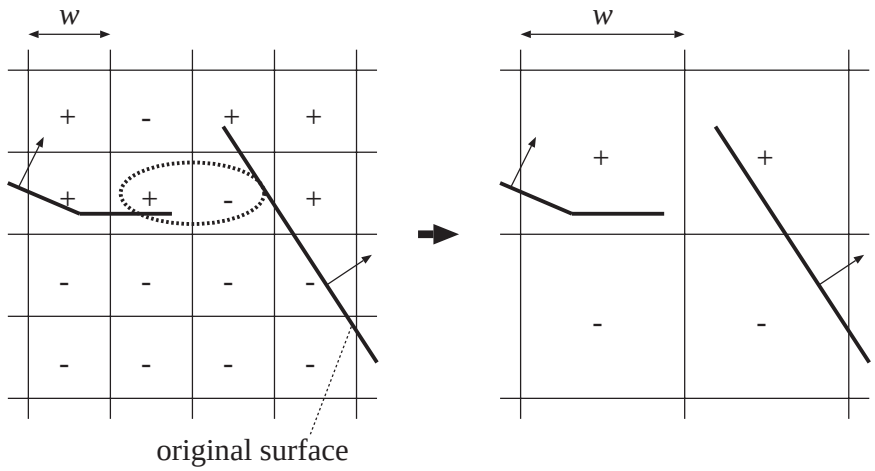


Figure 4.4: Determination of optimal sampling width

maximum resolution in which the geometric model is sufficiently consistent is determined by repeatedly downscaling the volume as shown in Figure 4.4. The algorithm for estimating optimal resolution of the volume is shown in Algorithm 4.2.

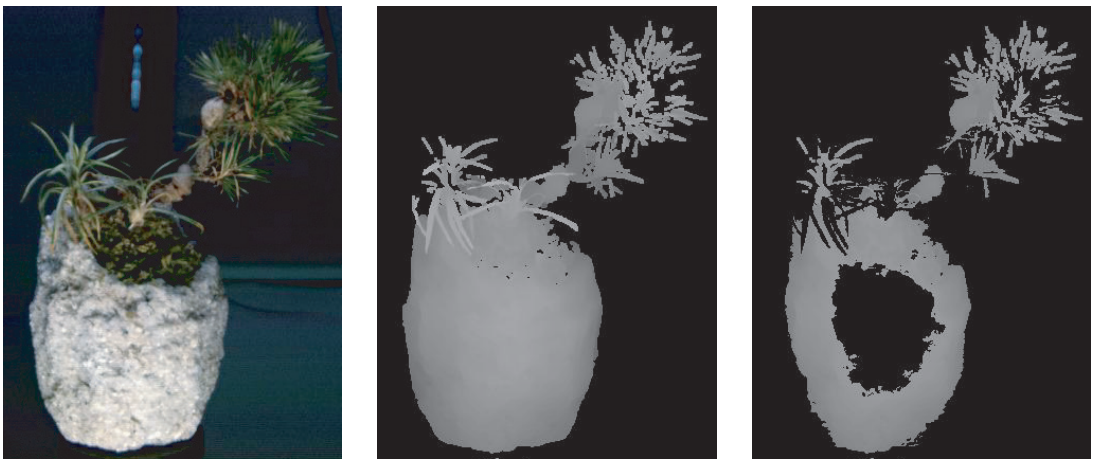
The function Consistent(v) in Algorithm 4.2 tests whether the voxel can be consistent between views. We implement this function such that

$$\text{Consistent}(v) = \begin{cases} \texttt{true} & \text{if} \quad \text{(difference of signed distance)} \leq \alpha w \\ \texttt{false} & \text{otherwise} \end{cases} \tag{4.1}$$

where $\alpha \geq 1$ is the parameter which controls the tolerance for the consistency and the difference of signed distance indicates the maximum absolute difference between the signed distance values assigned to the voxel and the adjacencies. The left side in Equation (4.1) can be regarded as the ratio of the voxels whose values are unreliable due to the sensor noises in acquired geometric model. In the current implementation, L_{max} is set to 256. α and τ are selected for each object wished to be rendered, and set as $\alpha = 1.5$ and $\tau = 0.01$ for the stuffed cow shown in Figure 4.3.

Generation of Local Geomtry

As is mentioned in Section 4.1, the global geometry of the model is approximated by a set of flat primitives which quantizes the depth of the object into the value corresponding to the depth of the facet in the rendering process. If the size of the primitive is sufficiently large, the flatness of the facets becomes apparent, thereby causing visual artifacts when



view image range image clipped range image

Figure 4.5: (left) One of the color images. (center) Range image from the same viewpoint. (right) Clipped result of the range image.

the viewpoint moves.

One approach to removing these artifacts is to clip the texture according to the depth of the object. In order to determine the depth of the object for each texel in the texture, range images with the same resolution as the corresponding color images are generated from the global geometric model for each color image. Since we assume that we have acquired color images from various viewpoints aligned with the global model, we can generate the range image of the model from the viewpoint where each color image was taken. When we render a microfacet of depth D from the viewpoint of a color image, a pixel of the color image which has depth d is rendered by

$$\begin{cases} \text{opaque color} & \text{if} \quad |d - D| < \frac{w}{2} \\ \text{transparent color} & \text{otherwise} \end{cases} \tag{4.2}$$

where w is the width of the space of which the facet takes charge. Figure 4.5 shows an example of a result of clipping. Nearby pixels are bright while ones farther away are dark in the range images. In the clipped image, black pixels are clipped and not mapped to microfacets.

It is worth noting that, although we use range images acquired using a laser range scanner to build the surface model, the generation of range images described in this section is still necessary because the positions where color images are taken are generally not the same as those where range images used to build the surface model are taken.

(a) (b)

Figure 4.6: (a) Originally captured object. (b) Extracted alpha matte.

Generation of Alpha Matte

In order to reduce the visual artifacts caused by limited resolution of concerning optic sensors, the opacity of the pixels in the image is estimated from reference images and used to increase the photo-reality in the synthetic images. The set of opacity values corresponding to the image is referred as *alpha matte*. To automatically and robustly generate alpha matte from a single image, a novel method of alpha estimation using Bayesian framework is proposed in Section 4.3. In this section, the detailed explanation of the method is skipped and the outline is presented.

As is reviewed in Section 4.3.1, most methods need to know the boundary between the foreground and the background to estimate the alpha value. Some of them are done by chromakey and some of them are employed manually. This time, we already have the depth data of the object which is matched to the image, which enables us to detect the boundary region in the color images.

In our algorithm, the SUSAN filter [SB97] is applied to detect the boundary region in a range image. Based on the segmentation, we generate the segmentation, called *trimap*, composed of foreground, background and boundary regions. Then the alpha matte of a color image is extracted by the proposed method of automatic alpha estimation with the trimap as an initial segmentation. Figure 4.6 shows an example of the alpha matte automatically generated from the corresponding range image.

Refinement of Local Geometry

The local geometry generated from the global geometry is supposed to have a considerable amount of errors since the global geometry is an approximation of the actual geometric model and is generated from highly unreliable range images. As a result of noise in range images, the range images generated by projecting the obtained surface model can have holes and missing parts which cause undesired deletion of the texture by texture clipping. Therefore, we apply a morphological filter to the obtained range image in order to remove the holes.

The detection of missing depth is detected based on the comparison between the range image and the corresponding alpha matte estimated in Section 4.2.1. First, the foreground pixels which should have their depth in the range image are detected by thresholding alpha matte. When the depth is not defined in the range image at the foreground pixel, a morphological filter is applied to fill the depth of the pixel from the surrounding depth values.

The filter is based on a fixed size of the window. For each pixel in the image, we assign a certain size of window whose center is the pixel. Then the value of each pixel is iteratively modified according to the values of the pixels within the window. In our adopted filter, the value of the pixel is replaced by the median of the values of the non-hole pixels within the window if the modified pixel is detected as a hole.

In our experiments, we use a 3×3 pixel window and the threshold is set to 5 pixels. An example of filtering range images is shown in Figure 4.7.

4.2.2 Rendering by Microfacet Billboarding

The outline of the rendering process is shown in Algorithm 4.3. The object is rendered by a set of microfacets with color texture. First, view-dependent microfacets are generated, and then the color image, alpha matte and depth map are generated and mapped onto each of them. The texture image mapped onto the facet is dynamically clipped according to the distance to the object in the rendering pipeline of graphics hardware.

Generation of Microfacets

A microfacet is defined as a slice which intersects the voxel and is aligned perpendicular to the viewing direction with a constant interval. Each microfacet represents the approximated surface inside the voxel occupied by an object. The interval is a tuning parameter

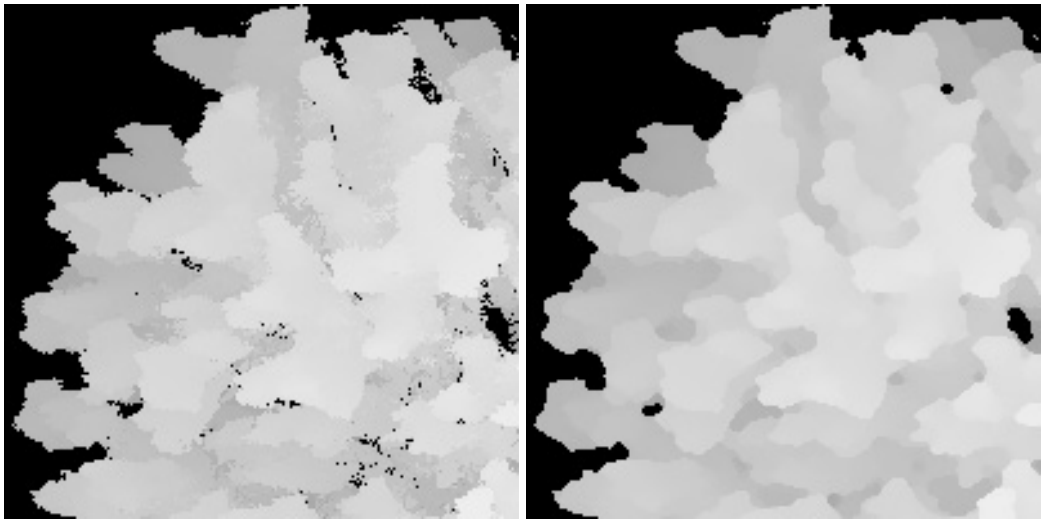


Figure 4.7: The holes in a range image can be removed by applying a morphological filter. (left) Original range image. (right) Filtered range image.

input: GlobalGeometricModel M
input: LocalGeometricModels $\{G_v\}$
input: ColorImages $\{C_v\}$
input: AlphaMattes $\{A_v\}$
input: Viewpoint V
local: Microfacets F
local: RGBATexture T
local: DepthTexture G
1: $F \leftarrow$ GenerateMicrofacets(M, V)
2: ClearFrameBuffer()
3: **for all** microfacet f in F **do**
4: $T \leftarrow$ GenerateRGBATexture($\{C_v\}$, $\{A_v\}$, m, V)
5: $G \leftarrow$ GenerateDepthTexture($\{G_v\}$, m, V)
6: RenderMicrofacet(m, T, G, V)
7: **end for**

Algorithm 4.3: Microfacet rendering

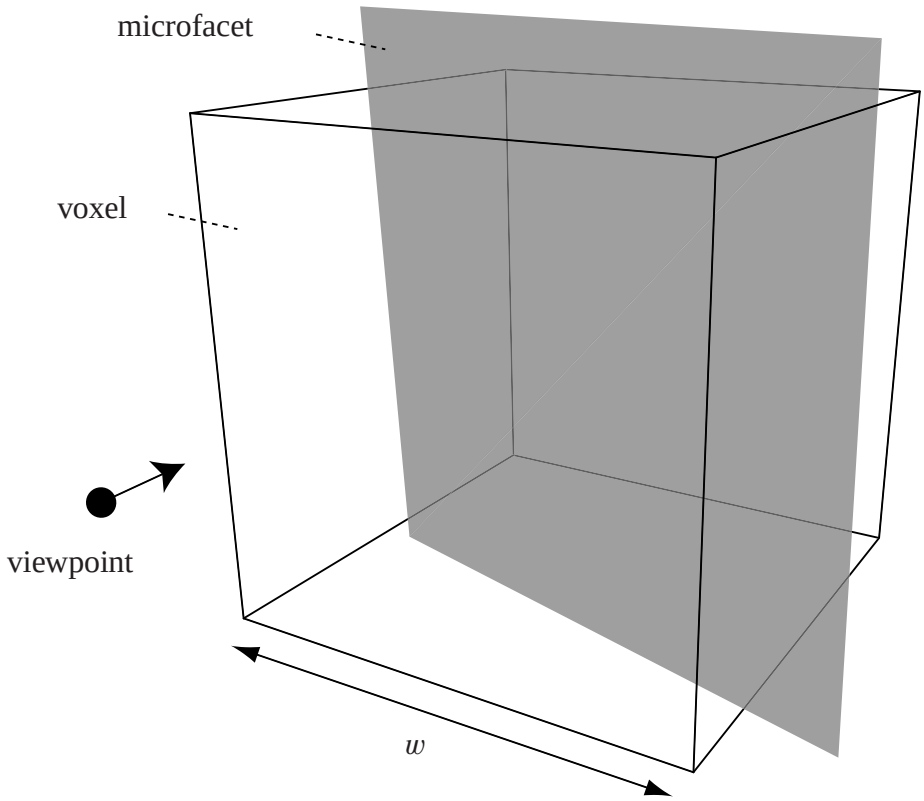


Figure 4.8: A microfacet is defined as a slice which intersects the voxel and is aligned perpendicular to the viewing direction with a constant interval.

that controls the speed and quality of rendering. The optimal interval is evaluated in Section 4.2.3.

The choice of the shape used to represent a microfacet can have a significant effect on the quality of the final image. With regard to the analogy of splatting techniques [RL00b], a square or an ellipse is possible. In our method, a square is chosen because a quadrilateral polygon is simple enough to render and to map a texture efficiently using standard graphics hardware. The width of the square is defined by the size of the voxel (see Figure 4.8). Namely, if the size of a voxel is w, the microfacet of $\sqrt{3}w$ will cover the voxel.

The quadrilateral facet is usually rendered as a group of small triangles since a small region of the texture image is mapped onto them. It, however, can be rendered as a point with a single color if the size of the facet is sufficiently small compared with the width of texture sampling. We implemented the renderer using triangles and that using points, and we compare their performance in Section 4.5.3.

Selection of Texture

The texture mapped onto a microfacet is selected or generated from input images according to the angle formed by the viewing direction for rendering and the camera direction of input images. The simplest way to generate texture is to select the image whose camera direction is "nearest" to the current viewing direction. The distance between directions is defined by the angles they form. Let a unit vector parallel to the current viewing direction be $\mathbf{v}$ and a unit vector parallel to the i-th camera direction be $\mathbf{c}_i$; then the distance d_i between the directions is defined as

$$d_i = cos^{-1}(\mathbf{v} \cdot \mathbf{c}_i). \tag{4.3}$$

With every change of viewing direction, distance d_i for each i is calculated; then, the camera position i which has minimum d_i is selected and the i-th camera image is mapped to the facet.

A better way to generate texture is interpolation. For every change of viewpoint, the distance between viewing direction and each camera direction is calculated. The parameters used for interpolating images are determined by the distances for all images; then, some or all of the images are blended according to the values. For smooth rendering, blending parameters should change continuously between 0 and 1. Let d_i be the angle formed by the viewing direction and the direction of i-th camera and n be the number of the selected cameras, then the blending ratio w_i for the i-th camera image can be determined as

$$w_i = 1 - \frac{d_i}{\max_{i=1,\ldots,n}\{d_i\}}. \tag{4.4}$$

For the simplicity of calculation, the weights are then normalized into $\tilde{w}_i$ so that they sum up to one.

$$\tilde{w}_i = \frac{w_i}{\sum_{i=1,\ldots,n} w_i} \tag{4.5}$$

If the camera positions are distributed spherically around the object, blending the three nearest camera images can accomplish smooth rendering. If the cameras are distributed uniformly in three-dimensional space, selecting the four nearest camera images can enable smooth blending. Generally, since it is not necessarily feasible to position the cameras uniformly in space, optimal selection of the camera images can involve complicated problems, as discussed in [BBM+01c].

The method of interpolation takes account of only blending, and not warping; there-

fore, coarse input images can result in such artifacts as ghost images. Adopting a more sophisticated interpolation technique such as view morphing [SD96] or the method proposed in Chapter 2 may produce a smoother and more accurate result.

Mapping of Texture Images

The selected camera images are mapped onto the facets by perspective projection. When an image is mapped, we post-process it by clipping the area which should be on the facet, using the range image described in Section 4.2.1. For every texel in the texture image, the depth d to the object is fetched from the corresponding range image; then, d is compared with the distance D to the point to which the texel is mapped on the microfacet. If Equation (4.2) is satisfied, the texel is mapped with an opaque color; otherwise, the point on the facet is rendered as transparent.

Assume that the viewpoint is on the line of view of a selected camera; then, the distance to the microfacet is equal for every point on the facet. In this case, texture can be clipped by comparing range data with fixed D for each facet. The distance D for a single facet, however, can vary since microfacets rotate according to the viewpoints. Consequently D must be changed for every point, even on a facet. In addition, d is defined in the camera direction, and not in the viewing direction; therefore, D must be calculated as the distance from the camera to the facet, instead of that from the viewpoint to the facet.

4.2.3 Analyses of Visual Artifacts

Microfacets are rendered with projective texture mapping; hence, the rendered view is photo-realistic in the sense that it is exactly the same as one of the input photograph when the scene is rendered at the position of the camera. As the viewpoint for rendering moves and goes off from the camera position, the synthetic view tends to be distorted due to the error in the underlying geometric model. This is the fundamental nature in the methods of image-based rendering, and is quantatively analyzed in the plenoptic sampling theory [CTCS00].

Apart from the inevitable error divined by the sampling theory, another type of visual artifacts, discontinuity or simply holes in synthetic images, can be observed due to the discontinuity of geometry representation by microfacets. The degree of possible discontinuity depends on the viewpoint for rendering since the geometric model changes view-dependently. In this section, the possible discontinuity observed in virtual views synthesized by the proposed method is estimated, and then the optimal interval between

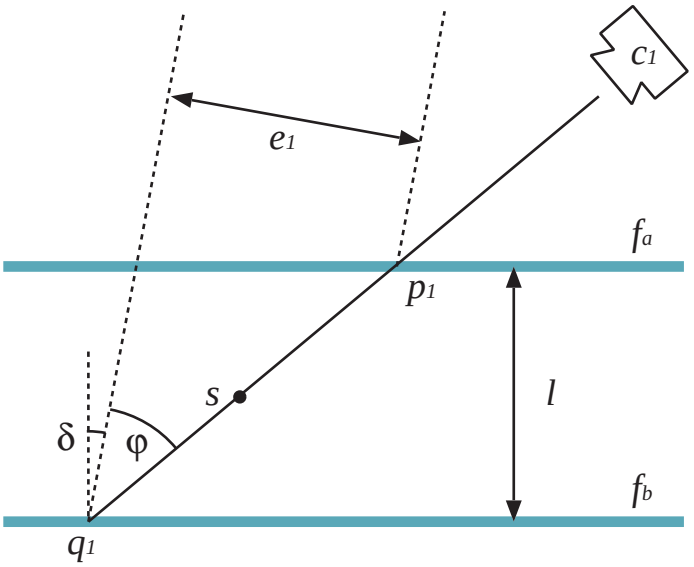


Figure 4.9: Visual discontinuity observed when a single camera image is projected onto two separate microfacets

microfacets mentioned in Section 4.2.2 is determined to accomplish the rendering of the highest quality with the lowest computational cost.

Let two adjacent microfacets be f_a and f_b, the interval between the microfacets be l, the index of currently selected camera be c_1, the angle formed by viewing direction and camera direction be ϕ, and the angle formed by viewing direction and the normal of microfacets be δ, as shown in Figure 4.2.3. It is also assumed that $\phi, \delta \in (-\pi/2, \pi/2)$ holds. For simplification, the size of microfacets is supposed to be sufficiently smaller than the distance between microfacets and viewpoint, hence the orthographic camera projection is assumed.

First of all, the interval of adjacent microfacets measured in the camera direction must be less than the voxel width w in order to generate at least one microfacet for each voxel.

$$\frac{l}{cos(\phi + \delta)} \leq w \tag{4.6}$$

Chosen one of the pixels observed in the image indicated c_1 whose associated depth is at the middle point s between f_a and f_b, its projection to f_a and f_b are denoted by p_1 and q_1 respectively. Then, the interval $\overline{p_1 q_1}$ is observed on the rendering screen in the size of

$$e_1 = l\frac{sin\phi}{cos(\phi + \delta)}. \tag{4.7}$$

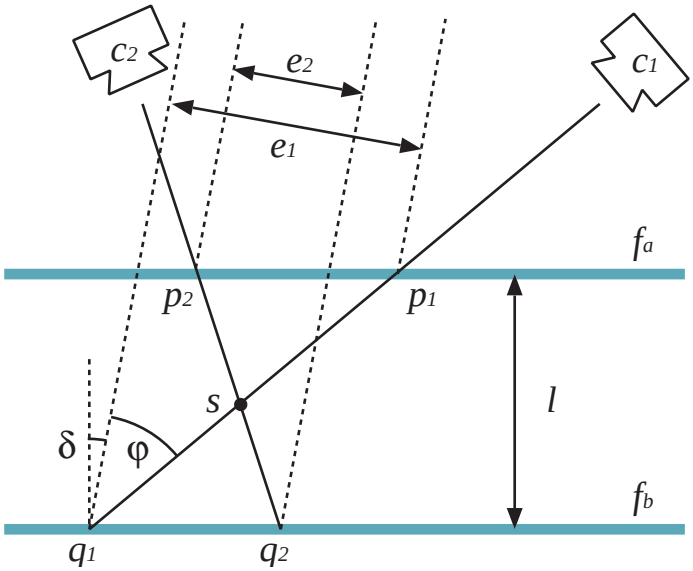


Figure 4.10: Comparison of visual discontinuities observed when two divergent camera images are projected onto two microfacets

This e_1 can be regarded as the discontinuity observed on the rendering screen as the side effect of approximating continuous geometry by a set of discrete microfacets.

As long as only the camera denoted by c_1 is concerned, the observable error e_1 is minimized when $cos(\phi + \delta) = 1$ holds, that is, $\delta = -\phi$ is satisfied from Equation (4.7). In practice, however, it is necessary to synthesize the texture images using multiple camera images, where a set of chosen cameras depends on the viewpoint for rendering. In order to keep the observable error small at all viewing points, it is desirable that the discontinuity in rendering is distributed among the views so that the largest discontinuity in all possible views is set to the smallest.

Consider the situation where the index of the camera whose direction is the nearest to current viewing direction changes from c_1 to c_2, as shown in Figure 4.2.3. The symbols in the figure are the same as those defined in Figure 4.2.3, and the point s projected to f_a and f_b in the direction of camera c_2 are denoted as p_2 and q_2 respectively. The angles formed by the viewing direction and the direction of camera c_1, c_2 are both equal to ϕ; hence, the interval $\overline{p_2q_2}$ is observed on the rendering screen in the size of

$$e_2 = l\frac{sin\phi}{cos(\phi - \delta)}. \tag{4.8}$$

Then the absolute difference between the discontinuities observed in the camera images is

$$\begin{aligned} |e_1 - e_2| &= \left| l\frac{sin\phi}{cos(\phi+\delta)} - l\frac{sin\phi}{cos(\phi-\delta)} \right| \\ &= 2lsin^2\phi \left| \frac{sin\delta}{cos(\phi+\delta)cos(\phi-\delta)} \right| \end{aligned} \tag{4.9}$$

Equation (4.9) takes the minimum value 0 for all ϕ when

$$\delta = 0 \tag{4.10}$$

holds. This means that the microfacets perpendicular to the viewing direction yield the smallest errors concerning the discontinuity when the viewpoint moves. The discontinuity e on the rendering screen observed when the a set of selected camera changes is calculated as

$$e = ltan\phi, \tag{4.11}$$

where the error depends on the viewpoint. Let the largest allowable error of discontinuity be E; then, the observed error e can be kept smaller than E by satisfying

$$ltan\phi \leq E. \tag{4.12}$$

Combining Equation (4.6), Equation (4.10) and Equation (4.12), the condition $e \leq E$ can be satisfied for all viewpoints by setting the interval l of microfacets so that

$$l \leq min(\frac{E}{tan\phi}, wcos\phi) \tag{4.13}$$

holds.

4.2.4 Controlling Visual Artifacts

The dominant factor that decides the speed of rendering is the number of rendering primitives, that is, microfacets. It is possible to increase the speed of drawing by reducing the number, which will decrease the quality in the synthetic views. To deal with the trade-off relationship, so-called the *level of details* (LOD) is introduced to control the quality of rendering according to the quality required at resolution. That is, the number of microfacets is changed, depending on the size of the object on the rendering screen.

The number of rendered microfacet can be changed by changing the level of volume

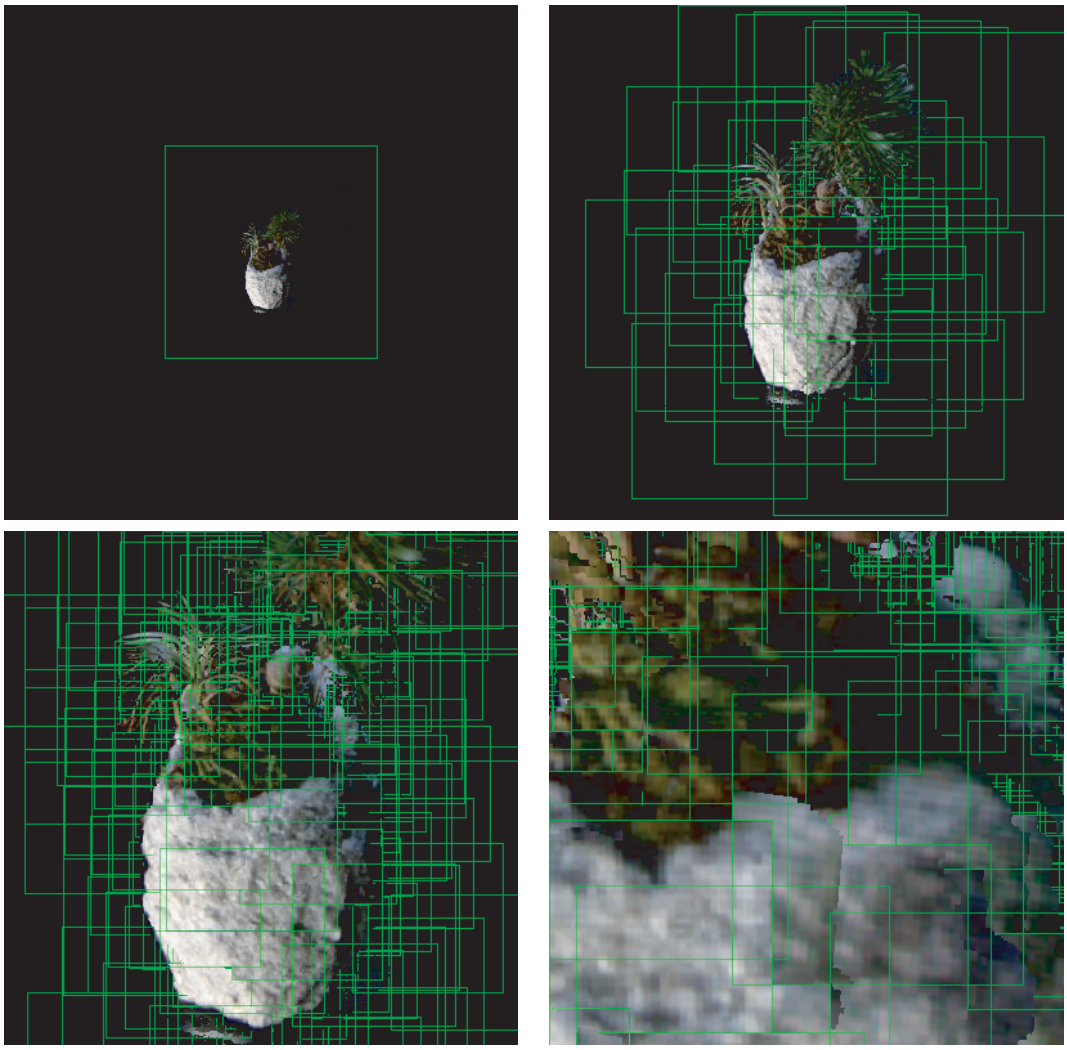


Figure 4.11: Level-of-detail control of microfacet generation

subdivision or by controlling the interval between microfacets. According to the observation on the interval, the interval larger than the maximum value in Equation (4.13) leads to visual artifacts observed as discontinuity of texture or holes in the synthesized images.

The LOD control of microfacet billboarding can be applied by changing the interval of microfacets together with the volume resolution, depending on the required resolution for the rendering without discontinuity. Let the voxel width be w_i when the size of volume by which microfacets are generated is $i \times i \times i$; then, the rendering without discontinuity is achieved by voxel size w_i and by setting the interval l of microfacet so that

$$l \in \left(\frac{1}{2} w_i cos\phi, w_i cos\phi\right] \tag{4.14}$$

holds. The volumes in all possible resolutions less than the maximum value L_{opt} are generated before rendering, and used one of them according to the change of the interval of microfacets.

When a virtual view of an object is synthesized with a certain viewpoint, the level of detail required at the resolution can be determined by Equation (4.11). Once the viewpoint and the size of rendering screen is fixed, it is desirable to set l to the maximum value such that the error e observed on rendering screen is within a predefined tolerance. Figure 4.2.4 shows an example of LOD control according to the change of rendering viewpoint. The predefined tolerance is set to eight pixels in the synthetic images.

4.3 Automatic Alpha Estimation

4.3.1 Introduction

The texture images used to render the object are taken using an ordinary camera; the images include background which should be removed before or when rendering. If the object has rigid geometry such as a smooth surface, it is unnecessary to remove the background because it is removed by texture clipping using depth information. However, since we focus on rendering intricately shaped objects, background can be removed not by clipping but rather, by alpha estimation.

Alpha estimation is the process of extracting a foreground element from a background in an image by estimating foreground and background colors and opacity of the foreground at each pixel. The opacity value at each pixel is called *alpha*, and a set of opacity values corresponding to the whole image is referred to as the *alpha matte*. Once the alpha matte

of an image is obtained, the foreground element in the image can be overlaid into other background scenes seamlessly. This operation is essential in photo-realistic image syntheses, such as 3D photography of intricately-shaped objects [MPN+02] or sparse light field rendering using rough geometric proxy [SSY+ar], as well as in such fields as publishing, television and film production.

The basic concept of alpha was first introduced by Porter and Duff [PD84] for purposes of image synthesis in computer graphics. The fundamental equation between pixel colors and alpha value can be described by the following *compositing equation*

$$C = \alpha F + (1 - \alpha)B \tag{4.15}$$

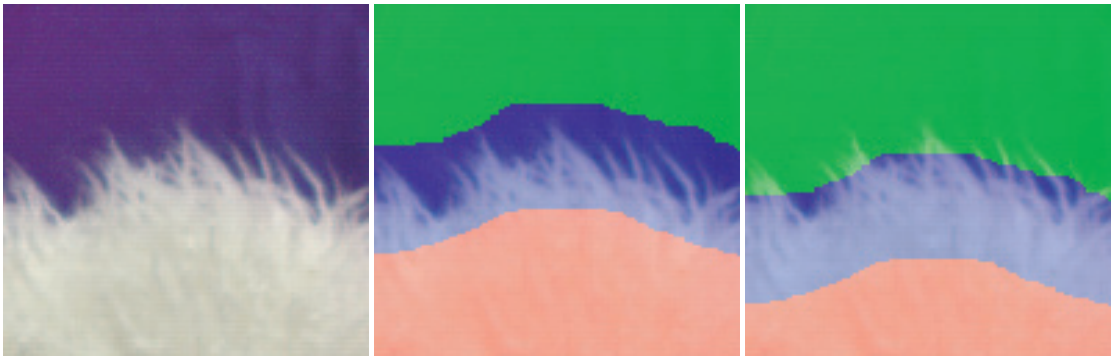
where C is the composite color observed in an image, F and B are respectively the colors of the foreground element and background scene at the pixel, and α is the opacity of the foreground. In this thesis, F, B and α are referred to as *matting parameters*.

Smith and Blinn [SB96] demonstrated that estimating matting parameters from given images, called the *matting problem*, provides a unique solution if two images with different background colors are given. Consequently, existing methods for the matting problem with a single image allow such assumptions on an input as follows, implicitly or explicitly.

1. **A priori segmentation:** Matting parameters in a part of a given image are a priori.
2. **Locality:** Nearby pixels are supposed to have similar colors.
3. **Distinguishability:** Foreground and background colors for each pixel are fairly separated in color space.

As for the first assumption, existing methods require the user to indicate the image areas which are definitely foreground, definitely background or to be alpha-estimated unless the algorithm can assume such an additional constraint as constant color background. The segmentation is called *trimap*, which is composed of foreground, background and unknown regions, as shown in Figure 4.3.1. Given an image and corresponding trimap, the system estimates potential matting parameters for all pixels in the unknown regions.

Based on the above-mentioned assumptions, the Corel's Knockout software [Cor] estimates foreground and background by a weighted sum of nearby pixels to solve underconstrained problems. Ruzon and Tomasi [RT00] introduced a statistical view to the matting problem and modeled observed colors by *spherical Gaussian* distribution where the covariance matrix is a scalar multiple of the identity matrix. As the approximation by



input image accurate trimap inaccurate trimap

Figure 4.12: Color image and corresponding trimap given as input. Foreground, background and unknown regions are painted in red, green and blue respectively.

spherical Gaussian is inaccurate for natural images, Hillman et al. [HHR01] applied principal component analysis to find the centers of color distribution. On the other hand, Chuang et al. [CCSS01] represented the distribution by *oriented Gaussians* and proposed the Bayesian estimation where the most likely parameters are estimated by maximizing a posterior of foreground, background and alpha values.

One of the practical problems in all these techniques is that they assume that the trimap is sufficiently accurate; otherwise, the estimation goes wrong on account of errors in statistics calculated according to the trimap. Since existing methods for automatic image segmentation are not always reliable, especially in the case where the image includes such complex effects as color blending and motion blur, it is necessary for the user to specify the trimap carefully before the process. This limitation is crucial in such applications as some image-based rendering systems (for example, opacity hull [MPN+02] or microfacet billboarding [YSK+02]) or video matting [CCSS01], where the number of images to be processed tends to be huge. If the algorithm is sufficiently robust to errors in a given trimap, it is potentially possible to automatically extract an alpha matte from a given image by using the rough trimap generated by certain appropriate methods like color-based segmentation.

In this thesis, we propose a novel method of natural image matting that is robust to errors in an initial trimap. Given an input image and corresponding trimap, foreground color, background color and alpha value are simultaneously estimated by maximizing a posterior for matting parameters like the Bayesian matting [CCSS01]. Our method, however, takes into account that the initial trimap may have errors, by penalizing the pixels that violate

the distinguishability assumption. The penalty is naturally formulated in the framework of Bayesian estimation using joint probability concerning matting parameters and their constraints. In addition, the distribution of pixel colors is modeled by the *Gaussian Mixture Models* (GMM) which can represent such complex effects as blending of colors. When an error in a trimap is detected, our method refines the trimap, and then solves the matting problem for each pixel iteratively until all unknown pixels have been processed.

4.3.2 Bayesian Matting

Our method is based on the Bayesian matting proposed by Chuang et al. [CCSS01], which appears to yield the best estimation among existing methods described in Section 4.3.1. In Bayesian matting, the matting parameters F, B and α are determined through the *maximization of a posterior* (MAP)

$$\begin{aligned} & \underset{F,B,\alpha}{\operatorname{argmax}} P(F, B, \alpha|C) && (4.16)\\ = & \underset{F,B,\alpha}{\operatorname{argmax}} P(C|F, B, \alpha)P(F)P(B)P(\alpha)/P(C) && (4.17)\\ = & \underset{F,B,\alpha}{\operatorname{argmax}} P(C|F, B, \alpha)P(F)P(B), && (4.18) \end{aligned}$$

where P indicates probability regarded as likelihood in the estimation. In Equation (4.17), $P(C)$ is eliminated since C does not affect maximization, and $P(\alpha)$ is dropped because no information on α is available without additional information on the object in the image.

Owing to the locality assumption on input, $P(F)$ and $P(B)$ in Equation (4.18) can be calculated as the weighted sum of colors of nearby pixels which are already known. Suppose each probability distribution P can be modeled by a Gaussian, the logarithm of the likelihood in Equation (4.18) comes down to second order polynomials composed of F, B and α, which can be maximized by finding such parameters that make the first derivative zero. The MAP estimation is applied for all unknown pixels sequentially from the inner and outer boundaries of unknown regions until all unknown pixels are processed.

4.3.3 Constrained Maximization of A Posterior

Thanks to the assumption on local coherence of pixel colors, the distribution of pixel colors around the pixel to be alpha-estimated is supposed to form clusters. When a trimap is sufficiently accurate that the colors of the true foreground F_{true} and true background B_{true} lie only in the foreground and background regions respectively, the color distribution

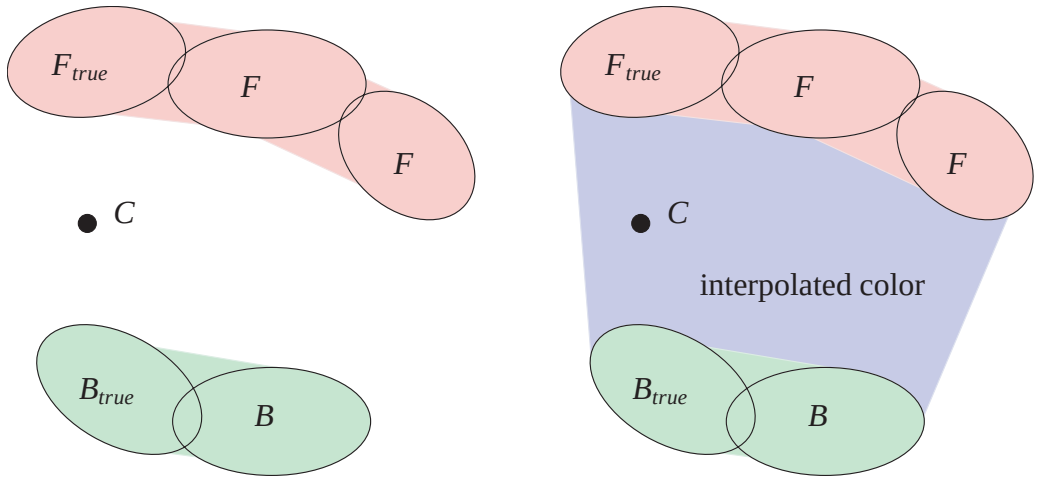


Figure 4.13: Color distribution of pixels in (left) accurate and (right) inaccurate trimap to estimate matting parameters at pixel

is expected to form two separated sets of clusters, namely foreground clusters and background clusters (Figure 4.3.3 left). However, when the trimap is inaccurate, the foreground clusters may include background colors, and vice versa. Moreover, both clusters include a considerable number of interpolated colors, thus making it difficult to estimate F_{true} and B_{true} through the MAP estimation because all Fs and Bs along the line between F_{true} and B_{true} provide great likelihood in the estimation of Equation (4.18) (Figure 4.3.3 right).

In order to avoid misestimating parameters owing to the mixture of color clusters, we introduce another constraint concerning the distance between F_{true} and B_{true} by regarding the distinguishability assumption described in Section 4.3.1 as a constraint to be satisfied. Let the additional condition be denoted by $F \neq B$; then, the *Constrained MAP* (CMAP) estimation can be formulated as the maximization of a joint posterior probability of matting parameters and the additional condition as follows.

$$\underset{F,B,\alpha}{\operatorname{argmax}} P(F, B, \alpha, F \neq B|C) \tag{4.19}$$

$$= \underset{F,B,\alpha}{\operatorname{argmax}} P(C|F, B, \alpha)P(F \neq B|F, B)P(F)P(B) \tag{4.20}$$

In Equation (4.20), $P(C)$ and $P(F \neq B)$ are assumed to be independent of $F \neq B$ and α, respectively.

The desirable property of the probability density function $P(F \neq B)$ is that the density should be small if $F = B$, and it should asymptotically approach its maximum as $|F - B|$

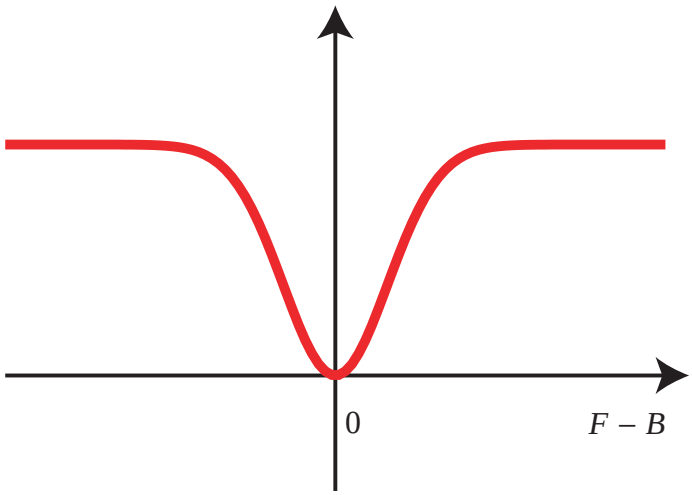


Figure 4.14: Probability density function of $P(F \neq B)$

becomes larger. Based on the observation, $P(F \neq B)$ can be modeled heuristically as

$$P(F \neq B|F,B) \propto \begin{cases} 1 - \exp(-\eta|F-B|^2), & \text{if} \quad |F-B| < \rho_B \\ 0, & \text{otherwise} \end{cases} \tag{4.21}$$

where $\rho_B > 0$ is an upper bound of $|F-B|$. $\eta > 0$ is a parameter that controls the acceptable distance between F_{true} and B_{true}. Note that the lower case in Equation (4.21) is considered so that the infinite integral satisfies $\iint P(F \neq B|F,B)dFdB = 1$, but has no effect on maximization. The probability density function $P(F \neq B)$ is illustrated in Figure 4.3.3.

4.3.4 Likelihood Models

While Ruzon and Tomasi [RT00] and Chuang et al. [CCSS01] approximated color distributions by the Gaussian model, it is inaccurate for natural images. As pointed out by Hillman et al. [HHR01], the shape of color clusters tends to be elongated due to such effects as illumination, even if the distribution forms a single cluster. With an erroneous trimap, however, the distribution can be composed of many clusters overlapping one another, owing to such effects as anti-aliasing, color blending and other noises in measurement as well as illumination.

In order to approximate complex shape of the distribution, we adopted the *Gaussian*

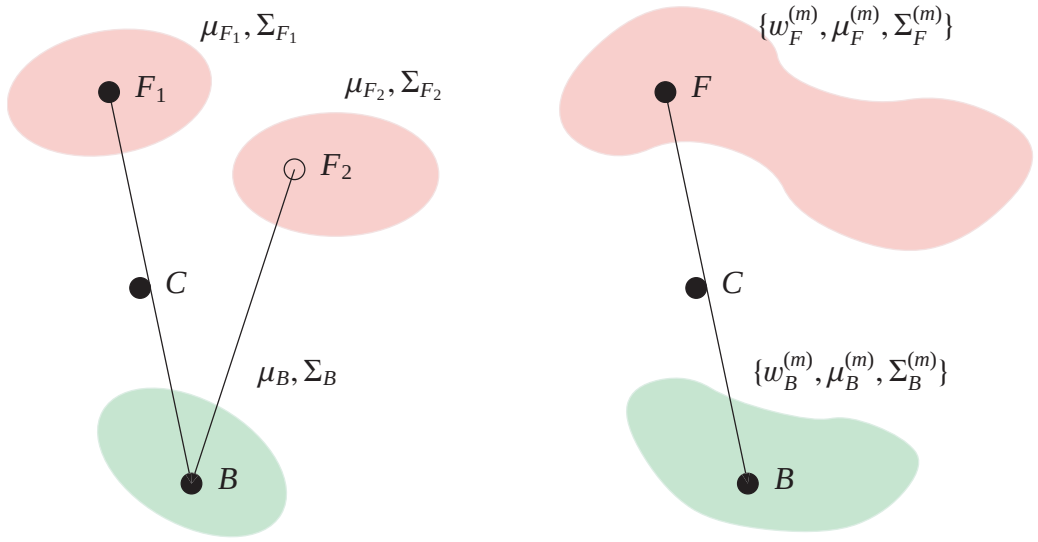


Figure 4.15: Maximizing a posterior estimation using Gaussian models (left) and Gaussian mixture models (right).

Mixture Models (GMM), which is represented as

$$GMM(x) = \sum_{m=1}^{M} w^{(m)} \, \mathrm{N}(\mu^{(m)}, \Sigma^{(m)})(x) \tag{4.22}$$

where M is the number of Gaussian components, $w^{(m)}$ is the weight for m-th component, and $\mathrm{N}(\mu^{(m)}, \Sigma^{(m)})$ is a Gaussian distribution centered at $\mu^{(m)}$ with covariance matrix $\Sigma^{(m)}$. The weight $w^{(m)}$ satisfies $\sum_{m=1}^{M} w^{(m)} = 1$. The comparison between Gaussian models and Gaussian mixture model is illustrated in Figure 4.3.4.

If the number of Gaussian components is known, the parameters $\{w^{(m)}, \mu^{(m)}, \Sigma^{(m)}\}$ in the GMM which most likely reproduce a given data set can be estimated by *Expectation Maximization* (EM) method [Moo96]. The number of components, however, is unknown in our case, hence we adopted two step estimation; First, M is determined by *subtractive clustering* method [Chi94], and $\{w^{(m)}, \mu^{(m)}, \Sigma^{(m)}\}$ are then estimated by EM method. The EM estimation in this algorithm converges very quickly, typically within a few iterations, since the centers of clusters estimated in the first step can be used as initial $\{\mu^{(m)}\}$ in the second step.

Using GMM, the probability density function for $P(F)$ and $P(B)$ can be written as

follows.

$$P(F) = \sum_{m=1}^{M_F} w_F^{(m)} \mathrm{N}(\mu_F^{(m)}, \Sigma_F^{(m)})(F) \tag{4.23}$$

$$P(B) = \sum_{m=1}^{M_B} w_B^{(m)} \mathrm{N}(\mu_B^{(m)}, \Sigma_B^{(m)})(B) \tag{4.24}$$

As for the probability density function for $P(C|F, B, \alpha)$, we used the same function as the Bayesian matting, that is,

$$P(C|F, B, \alpha) = \mathrm{N}(\alpha F + (1 - \alpha)B, \sigma_C^2 \mathbf{I})(C) \tag{4.25}$$

where σ_C^2 is the variance of observed colors.

4.3.5 Numerical Maximization

Since the function to be maximized in Equation (4.20) is supposed to be non-monotonic, special effort must be made toward stable calculation from the view of both algorithm and implementation.

First, we adopted iterative methods to maximize the objective function using its gradient, since it is difficult to solve the problem in a closed form. In order to accelerate the convergence, we used *conjugate gradient method*, which guarantees that the calculation falls into a local maximum within the same number of iterations as the dimension of parameters if the function is in quadratic form. Although our function is not exactly quadratic, this method works very well and converges to a maximum within two iterations on average.

Second, the maximization algorithm must be designed so that the algorithm can find the global optimum. Broadly speaking, each distribution of GMM has local maxima less than, or equal to, the number of Gaussian components. In addition, it is supposed that at least one of the mean $\mu^{(m)}$ in a GMM is reachable to the global maximum monotonically. This observation implies that the iterative method can reach the global maximum by locally maximizing the objective function starting from all combinations of mean values in $P(F)$ and $P(B)$. In our algorithm, the CMAP estimation is repeated in $M_F \times M_B$ times with every pair of $\mu_F^{(m)}$ and $\mu_B^{(m)}$ as the initial estimation for F and B, respectively. Initial α is set to be the weighted mean of α at neighbor pixels as with the original Bayesian matting.

Third, parameters estimated by iterative optimization can be outside their domain due

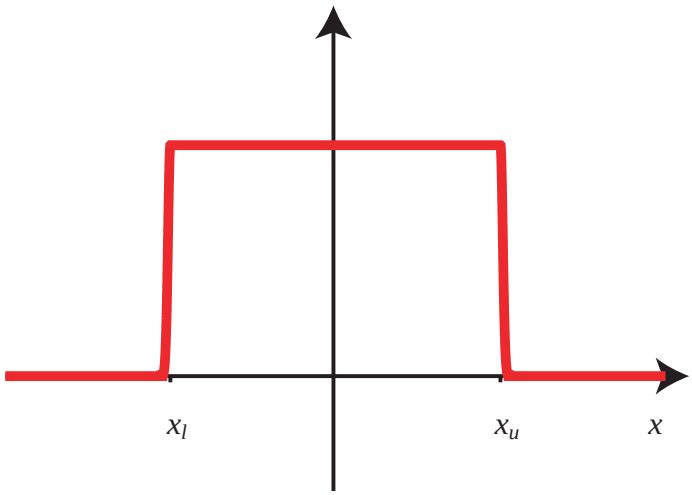


Figure 4.16: Probability density function of $P(x \in \mathcal{D}_x)$

to noise or incorrect assumption on the locality of colors; in particular, it often occurs that estimated α is less than zero or more than one, which is unacceptable from the definition of opacity. In order to avoid such invalid values, the constraints on the range of valid parameters are also taken into account in CMAP estimation. Let these conditions be $F \in \mathcal{D}_F$, $B \in \mathcal{D}_B$ and $\alpha \in \mathcal{D}_\alpha$, then CMAP estimation in Equation (4.20) is modified to

$$\begin{aligned} & \underset{F,B,\alpha}{\operatorname{argmax}} P(F, B, \alpha, F \neq B, F \in \mathcal{D}_F, B \in \mathcal{D}_B, \alpha \in \mathcal{D}_\alpha | C) \\ = \quad & \underset{F,B,\alpha}{\operatorname{argmax}} \begin{array}{l} P(C|F, B, \alpha) P(F \neq B|F, B) P(F) P(B) \times \\ \qquad P(F \in \mathcal{D}_F) P(B \in \mathcal{D}_B) P(\alpha \in \mathcal{D}_\alpha) \end{array}, \end{aligned} \tag{4.26}$$

where $\mathcal{D}_x = [x_l, x_u]$ is the domain of the parameter x. The likelihood of the additional constraints can be modeled as

$$P(x \in \mathcal{D}_x) \propto \begin{cases} \exp(-\rho_D (x - x_l)^2) & \text{if} \quad x < x_l \\ 1 & \text{if} \quad x_l \leq x \leq x_u \\ \exp(-\rho_D (x - x_u)^2) & \text{if} \quad x > x_u \end{cases} \tag{4.27}$$

where $\rho_D > 0$ is a huge number. x denotes either a color channel in F or B, or α. The probability density function $P(x \in \mathcal{D}_x)$ is illustrated in Figure 4.3.3.

Fourth, we also devoted considerable care to the implementation of function evaluation. Since the exponential functions can easily overflow or underflow the IEEE 64-bit floating point number (namely, `double` in C language), it is necessary to take very small

or large numbers into account in the evaluation of exponentials and logarithms. We dealt with the problem in two ways; one is by incorporating arbitrary precision arithmetic, for example, by using GMP library [GMP]. The other is simply by ignoring values smaller than a certain threshold. The former is better in quality and slow, while the latter may suffer from severe round-off errors in compensation for fast calculation achieved by floating point arithmetic hardware. In our experiments, the latter method was used since the difference in quality was very small.

Last, we took the logarithm of the function in Equation (4.20) for computational stability before maximization, though it cannot be decomposed into such simple forms as polynomial.

4.3.6 Trimap Correction

While the matting parameters in the area which is specified as an unknown region in initial trimap can be determined by the proposed robust estimation, the parameters for the pixels are incorrectly specified as either foreground or background even though they have medium alpha values. In order to obtain an accurate alpha matte as a whole, it is desirable to estimate correct alpha values in such region.

If the trimap is accurate, the distribution of foreground and background colors provides two sets of clusters, while clusters overlap one another when the trimap is inaccurate as described in Section 4.3.3. Based on the observation, the wrongly classified pixels are likely to be positioned in either foreground color in background clusters, background color in foreground clusters, or interpolated color of foreground and background colors. Since it is difficult to determine which color cluster is classified into foreground or background, or which area can be regarded as interpolated colors, we adopted the following conservative method.

After each iteration in the CMAP estimation, pixels satisfying the following conditions are regarded as unreliable pixels which need to be relabeled as unknown and alpha-estimated later.

- Foreground color in the intermediate region and not in the cluster of F_{true}
- Background color in the intermediate region and not in the cluster of B_{true}

The unreliable pixels are determined using the Mahalanobis metric with covariance matrix $\sigma_C^2\mathbf{I}$ in color space. The distance d_{line} between a pixel and an intermediate region is measured by the *Mahalanobis distance* between the color and the line of $\overline{F_{true}B_{true}}$. The

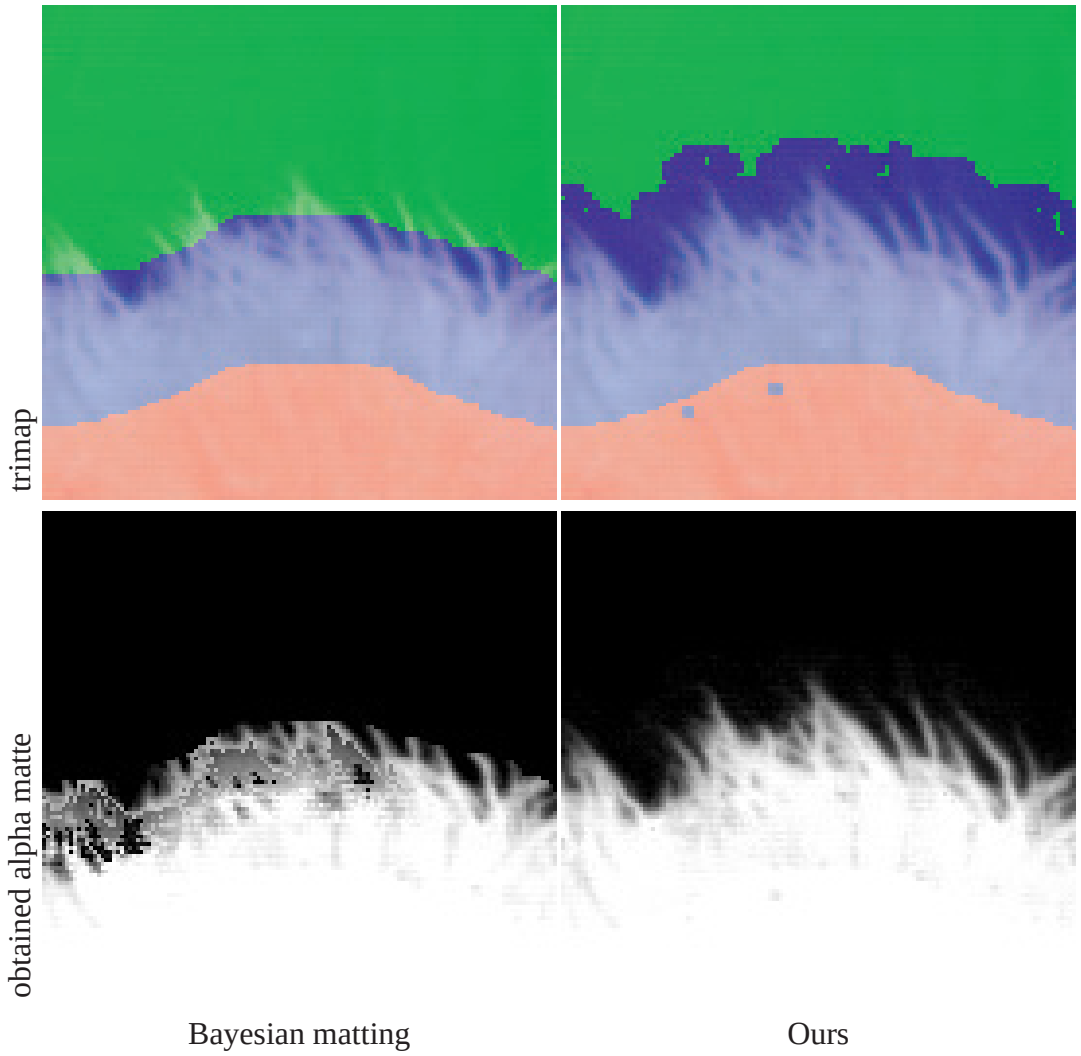


Figure 4.17: Trimap correction and estimated alpha matte. While both algorithms are initialized with the trimap shown on top-left, our method yielded better alpha matte by correcting errors in the initial trimap.

distances $d_{cluster}$ to the clusters of F_{true} or B_{true} are calculated in the same manner. In our experiments, the thresholds for d_{line} and $d_{cluster}$ are set to 3.0 and 1.0, respectively. The result of trimap correction is shown in Figure 4.3.6.

4.4 Implementation Details

The rendering by microfacet billboarding can be effectively implemented by commonly used graphics library such the OpenGL and the Microsoft Direct 3D library. Microfacets are rendered by polygons with multitextures, taking advantage of the fast and flexible texture synthesizer called *pixel shader*. In this section, we explain the detail of the hardware-accelerated implementation for a texel-wise operation which is necessary to perform the texture clipping described in Section 4.2.2.

In order to map the obtained images to microfacets during the run time, all color images, view-dependent depth maps, and alpha mattes are loaded into the graphics memory before rendering. In rendering a polygon representing a single microfacet, some trios of the color image, depth map and alpha matte are selected according to the camera selection described in Section 4.2.2. To avoid the color blending between foreground and background, the alpha mattes are first applied to the corresponding color images. The textures are then blended according to the weights $\tilde{w}_i$ determined by the angles formed by the current viewing direction and the direction of cameras. Once a polygon representing a single microfacet has been rasterized into a set of pixels and the corresponding texels have been generated, the interpolated depth of the pixel is then compared with the depth of the microfacet, and then passed to the following process if the pixel is within the voxel represented by the microfacet, otherwise it is discarded. If the pixel passes the texture-based depth test, the pixel is rendered into a frame buffer using alpha blending.

This texture-based depth test and alpha blending is implemented as a pixel shading program and performed within the rendering pipeline of graphics hardware using a pixel shader when microfacets are rendered. In our current implementation, the High Level Shader Language (HLSL) for the Microsoft Direct 3D version 9 is used to program the functions, although it can also be implemented using the OpenGL library with the aid of other programming languages such as NVidia Cg. The color images, view-dependent depth maps and alpha mattes are loaded as 2D textures, and the depth of microfacets and other parameters related to the viewing direction are loaded into the registers in the pixel shader.

Table 4.1: Performance comparison. Figures indicated are seconds elapsed in the alpha estimation for the image and trimap presented in Figure 4.3.1.

	Bayesian	Ours
Accurate trimap	44.893	44.979
Inaccurate trimap	45.350	44.861
Trimap correction	118.228	118.535

4.5 Experimental Results

We implemented a microfacet billboarding renderer on a standard PC equipped with ATI Radeon 9800 Pro GPU. Microfacets were rendered as squares of texture-mapped polygons using the Microsoft Direct 3D version 9 graphics library and its pixel shading functions, taking advantage of the rendering acceleration of graphics hardware.

4.5.1 Automatic Alpha Estimation

The performance of the methods is presented in Table 4.1. For all constraints on color distribution and range of variables, our algorithm is as fast as the original Bayesian matting. Although the conjugate gradient method for CMAP estimation in our algorithm is more complicated than the alternative minimization used in the Bayesian matting, the number of iterations is considerably smaller in our method. The average number of iterations in conjugate gradient method for each single pixel is about two, while the alternative minimization requires 10 to 20 iterations on average. When trimap correction is enabled, the number of pixels to be alpha-estimated increases and the algorithm slows down.

The examples of automatic digital matting are shown in Figure 4.5.1. Robust alpha estimation enables the user to utilize rough segmentation obtained by any method such as initial trimap. The alpha images shown at the bottom are generated without any manual segmentation. We generated initial trimaps by applying the SUSAN filter [SB97] to a range image captured by a laser range finder, a disparity map obtained by binocular stereo, and a simple color-based segmentation. The width of unknown regions in the trimap is determined by manually adjusting the filter parameters. The a priori parameters like σ_C or η are also determined empirically, while similar scenes can be processed using similar parameters.

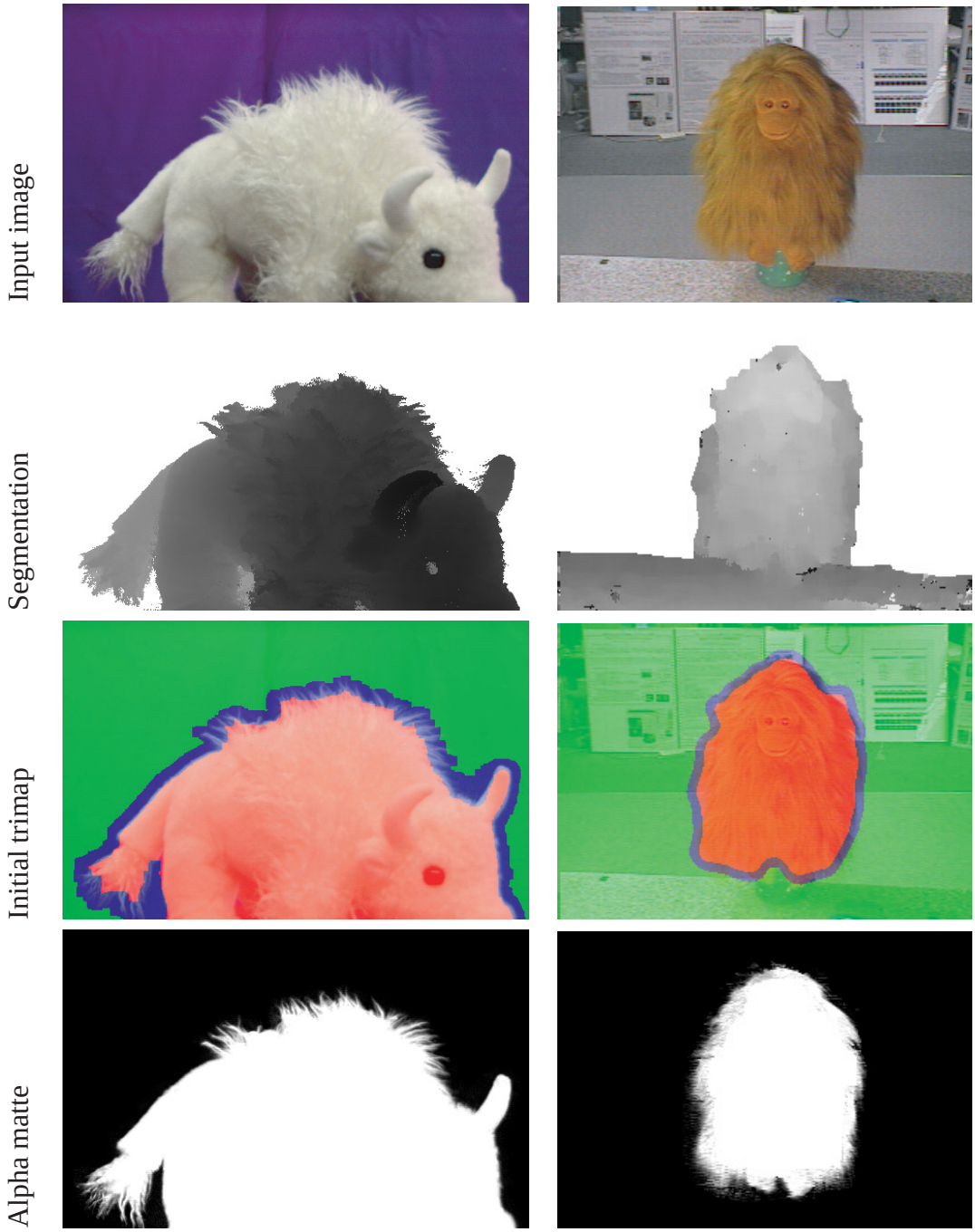


Figure 4.18: Example of automatic alpha estimation. Each row shows input images, rough segmentation used to generate initial trimaps, generated trimaps, and obtained alpha matte from top to bottom. The trimaps are generated by applying the SUSAN filter to a range image taken by a laser range finder in the left column, and a disparity map obtained by a stereo method in the right column.

4.5.2 Rendering Objects by Microfacet Billboarding

A stuffed cow with soft and intricate geometries was modeled and then rendered, as shown in Figure 4.19. The model is furry over a large part of its surface and it is difficult to measure and to represent its geometry completely.

In our experiment, the geometry and texture of the object were acquired at the same time, in the form of a sequence of range images and texture images, using the VIVID 900 laser range scanner. The number of images captured for the model was 36. Although our method has no limitation on the position of cameras, we assume that the cameras were placed along a circle because a turning lathe was used to create the model in this experiment. This circular camera position confined the area in which the viewpoint could be positioned to a two-dimensional plane which contained the circle of the camera positions. However, the information on the geometry of the object expanded this two-dimensional plane to three-dimensional space with the distortion estimated in Section 4.2.3.

Figure 4.19 shows the intermediate and final results of microfacet billboarding rendering. The objects to be rendered are the stuffed toys of a bear and a cow. The acquired geometry and texture were composed of 36 range images and color images taken around the objects. The resolution of the range image and color image is 640×480. The resolution of global geometry is $64 \times 64 \times 64$. The top row in Figure 4.19 shows the global geometries rendered by the technique of volume rendering with red and green colors for the voxels outside and inside surfaces, respectively. The middle row shows the result of microfacet rendering with texture clipping, and the bottom row displays the final result of the rendering.

The result of background substitution is demonstrated in Figure 4.20. The synthetic view image rendered by microfacet billboarding is shown on the left. The right image is an unused reference view taken at the same viewpoint. Compared with the reference, synthetic image is a little blurred. This can be regarded as the artifacts caused by the error in the geometric approximation.

4.5.3 Performance Analysis

The speed of rendering is mainly dependent on the number of microfacets to be generated in the resolution. An object was rendered in various resolutions by using (a) quadrilateral polygons with view-dependent textures, (b) points with view-dependent colors, and (c) quadrilateral polygons with accelerated view-independent textures, as shown in Table 4.2.

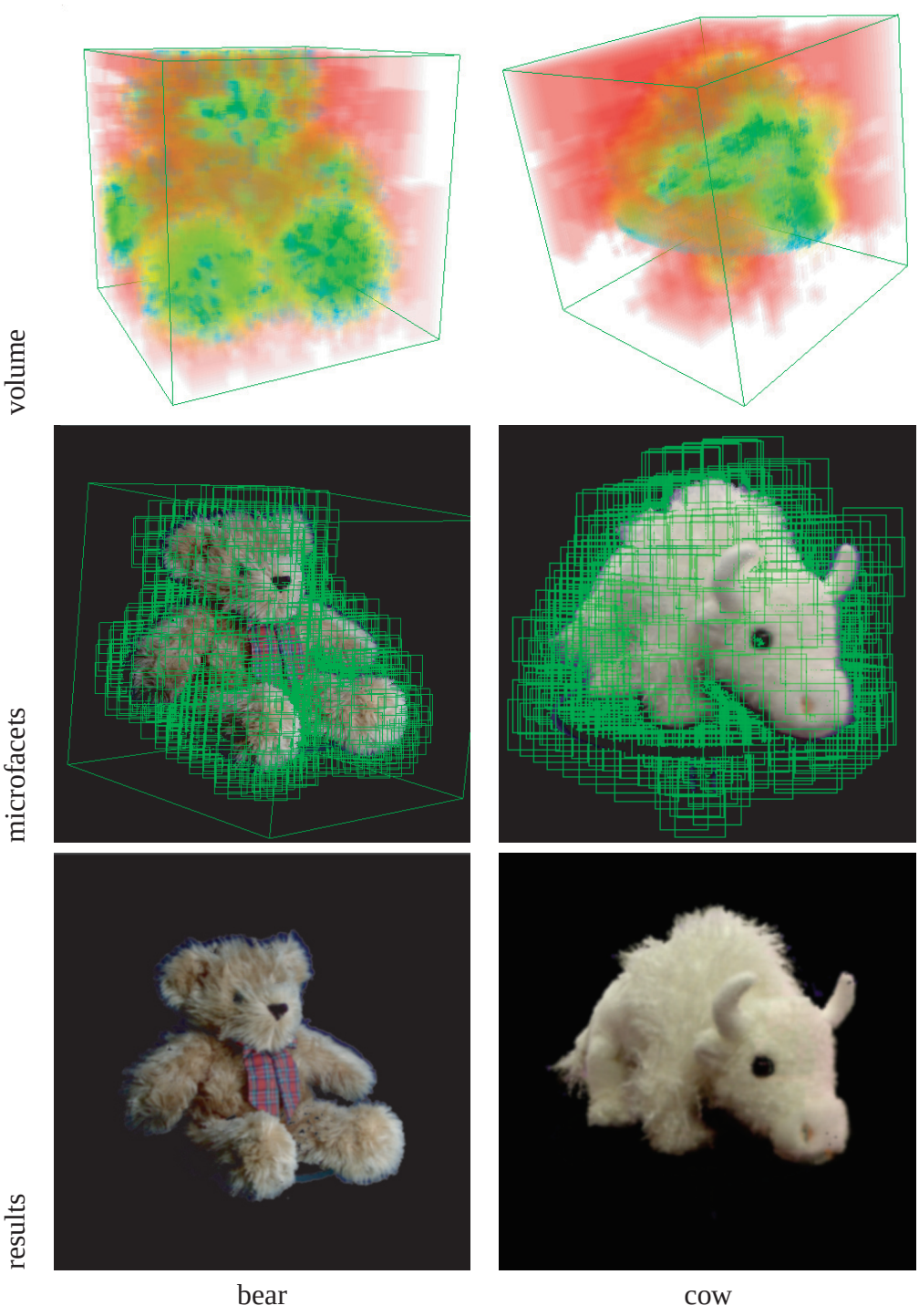


Figure 4.19: Images of stuffed toys rendered by microfacet billboarding rendering. (top) Voxelized geometry visualized by volume rendering. Green cells indicate the voxels placed at the position of object surface. (middle) Virtual views synthesizsed by microfacet billboarding. (bottom) Results of rendering without facet border.

synthetic view-image

unused reference view-image

Figure 4.20: Results of alpha composition

Table 4.2: Example of rendering performance. The top two rows indicate the size of the data. The bottom rows show the number of frames rendered per second (FPS) when microfacets are represented by (a) quadrilateral polygons with view-dependent textures, (b) points with view-dependent colors, and (c) quadrilateral polygons with accelerated view-dependent textures.

Size of volume	32^3	64^3	128^3	256^3
# of facets	2460	11207	50560	208289
(a)	100.1	69.8	18.2	5.2
(b)	113.3	72.4	20.5	6.4
(c)	187.6	162.3	46.3	20.1

Contrary to our expectation, a comparison of (a) and (b) in Table 4.2 indicates that the shape of microfacets has little effect on the rendering performance. The reason is that the bottleneck of the rendering process is not the rasterization of microfacets in graphics hardware but rather, the camera selection for view-dependent texture. The calculation for camera selection can be accelerated by limiting the number of candidates for the selection. With every change of viewpoint, we generate a cone which includes all possible directions from the viewpoint to microfacets; then, some texture images are selected as the candidates for camera selection using the cone. This simple technique can increase the performance of rendering, as shown in row (c) in Table 4.2. Since our current implementation has not been optimized for performance, the adoption of more sophisticated techniques of camera selection or other processes may accelerate the performance. Further improvement of performance is not discussed in this thesis.

The depth in the range images is quantized as integer in $[0, 255]$ due to the limitation of the graphics hardware. Therefore, microfacets finer than those generated from a volume of more than 256^3 have the same quality of texture. This limitation can be overcome by using floating point texture map available in the latest hardware.

The required memory size for microfacet billboarding depends on both the number of facets and the number of cameras. For geometry, one microfacet requires three 32-bit floating points to represent its position in space. The normal and rotation of a microfacet are determined dynamically when it is rendered. Therefore, the required memory size for geometry is roughly estimated as $24 \times \{\text{number of facets}\}$ for each resolution of the volumes. For texture, one texture image requires $w \times h$ pixels which have 5 channels (RGB, depth and alpha) in 8-bit precision. Therefore, the required memory size for texture is roughly estimated to be $5 \times w \times h \times \{\text{number of images}\}$.

4.5.4 Comparison with the Surface Model

It is common for models with clear and static surfaces to be rendered by methods using polygons with texture; hence, we should compare our method with such methods. One of the most suitable techniques for rendering objects with intricate geometry is the combination of visual hull modeling and view-dependent/independent texture mapping. The visual hull algorithm can generate a model of the approximate geometry of the object independently of the intricacy of the surface of the object, because this algorithm utilizes only the silhouette line of the object. However, the detail of an object which has an intricate silhouette cannot be modeled correctly.

unused image | polygon + texture | microfacet billboarding

Figure 4.21: The texture mapped to a microfacet is interpolated when viewed from a location between camera positions. (left) An image which is not used for rendering. (center) The image synthesized by texture-mapped surface rendering at the position of the image on left. (right) The image synthesized by microfacet billboarding, which successfully reproduced intricately shaped needles of the plant.

Figure 4.21 shows a result of the comparison. Several images of a potted plant were captured, and one of them is shown on the left. First, the geometry of the object was generated using a consensus surface algorithm [WSI98], and represented by a polygonal mesh (center) and microfacets (right). Then, both models were rendered using view-dependent texture mapping. Although the quality of both models is poor, microfacet billboarding reproduced the real view much better than did the surface-based algorithm.

Chapter 5

Conclusion

5.1 Summary

In this dissertation, three different techniques for photorealistic image synthesis based on measured data sets have been proposed. The motivation of this research is that, owing to some fundamental or practical reasons concerning measurement, it is often difficult to model real-world scenes accurately enough to achieve photo-realistic rendering.

Image-based Rendering from Sparse Images

When the knowledge on the scene geometry is completely unavailable and it is difficult to extract geometric constraints from given reference images, the existing methods cannot yield photo-realistic views without a huge amount of reference view images as inputs. In this dissertation, two different solutions to this issue are proposed. One is a fully automatic method based on automatic image morphing. The other solution is an interactive system for image-based modeling and rendering.

In the automatic method for photo-realistic rendering based only on images, the virtual view of the scene is synthesized by interpolating the images using the possibly non-physically based correspondence between the images. We have proposed a method of automatically determining dense and smooth mapping between two images without a priori information on either the camera pose or the objects. In order to determine plausible correspondences even between different images, features of images are extracted by linear filters similar to those used in early vision. We have applied our method to various data sets and showed that our method works better than existing methods when intensity

changes or the difference between images is large.

Interactive system for image-based modeling and rendering

The development of an interactive system for modeling and rendering, which we call *Pop-up Light Field*, is another solution to the problem of photo-realistic rendering from a sparse set of images. As a basic model of the scene, we introduced coherent layers representation which is a collection of corresponding planar regions in the light field images. A coherent layer can be rendered free of aliasing, all by itself or against other background layers. To construct coherent layers, we introduce a Bayesian approach, *coherence matting*, to estimate alpha matting around segmented layer boundaries by incorporating a coherence prior to maintain coherence across images. In this system, the user specifies how many coherent layers should be modeled or popped up according to the scene complexity. We have developed an intuitive and easy-to-use user interface (UI) to facilitate pop-up light field construction. The key to our UI is the concept of human-in-the-loop, in which the user specifies where aliasing occurs in the rendered image. The user input is reflected in the input light field images where pop-up layers can be modified. The user feedback is instantaneous through a hardware-accelerated real-time pop-up light field renderer. Experimental results demonstrate that our system is capable of rendering anti-aliased novel views from a sparse light field.

Image-based Rendering based on Incomplete Geometry

When the objects that we wish to render have such elaborate details as hairs and pine needles, acquired geometry may suffer from considerable noise and view-images are wrongly affected due to the limited resolution of optic sensors. With the goal of synthesizing the photo-realistic views of such intricately-shaped objects, we have proposed a novel method, called microfacet billboarding, that approximates intricate geometry by using view-dependent geometric primitive, and renders the objects by using view-dependent texture mapping. We described the basic algorithm and an effective implementation on commodity graphics hardware. We also told how we estimate artifacts generated by the use of discrete facets, and how we analyze the optimal parameter settings that yield the best rendering performance without losing the quality. The comparison with the view synthesized by conventional surface-based method proves that the proposed method is highly advantageous for rendering intricately shaped geometry.

Within the algorithm of microfacet rendering, a robust method for estimating alpha matte from a single image is also presented. Different from prior work, the proposed method enables us to correct the error in an initial segmentation called trimap; hence, automatic estimation of alpha matte is achieved based on a rough initial segmentation generated from rough geometric model of the scene. Based on the Bayesian matting technique, the robust estimation is formulated as the maximization of a joint posterior probability of matting parameters and the additional condition, which can be solved by the conjugate gradient method without conspicuous loss in computational efficiency. In addition, we introduced the Gaussian Mixture Model (GMM) for representing complicated distribution of image colors especially when a given initial trimap has considerable errors, and demonstrated that the proposed method enables automatic digital matting which is essential in applications requiring many images to be alpha-estimated.

5.2 Contribution

The contributions of this dissertation are summarized as follows.

- Proposed is a new framework to achieve view-interpolation from sparse images; this framework is based on the techniques of automatic image matching and hardware-accelerated image morphing. As opposed to prior methods of automatic image matching, the proposed method does not rely on any assumption of what appears in the images. To avoid the difficulty in the geometric reconstruction of the scene wished to be rendered, the problem is reduced to finding plausible mapping between the images. To solve the problem, the correspondence between given images is determined by matching the image features extracted by the filters similar to those used in human vision system.

- Also proposed is a new framework for interactive modeling and rendering using sparse light field. The scene wished to be rendered is decomposed into a set of *coherent layers* to achieve alias-free rendering. To construct coherent layers, the technique which we call *coherence matting* is introduced to estimate alpha matting around segmented layer boundaries by incorporating a coherence prior to maintain coherence across images. To facilitate the layer construction, an intuitive and easy-to-use UI was designed so that the rendering quality can be improved as much as the user wants with little manual operation.

- Also proposed is a new framework to create the geometric model of the objects whose surfaces have such elaborate features as needles of pine trees, hairs, and fur. The key idea is to introduce view-dependency in both modeling and rendering. Due to the limited resolution of optic sensors, it is difficult to consistently create a static model from an acquired data set between views. Instead of resolving the inconsistency, the geometric model is represented as a combination of static and view-dependent data structures, and the texture images are generated using the corresponding alpha matte.

- In addition to these three fundamental methods of photo-realistic rendering, also proposed is a robust method of alpha matting based on the Bayesian framework. By extending the MAP estimation to consider the constraints on the color distribution, the method makes it possible to extract an alpha matte from a single natural image by the use of a rough trimap. Although this work is a by-product developed in the course of designing the microfacet billboarding, it helps the alpha matting performed in the conventional context, such as image editing in publishing.

5.3 Discussion and Future Work

We conclude with a discussion of open problems and future improvements which we ourselves are interested in pursuing and would like to see pursued by other researchers.

Combination of view-dependent rendering and image morphing

The view-dependent modeling of an object allows the system to deal with a globally inconsistent representation of the geometry. In this fuzzy approach, we can effectively avoid such problems as that a part of acquired data set is excessively eliminated due to some "robust" modeling, which occurs in the visual hull modeling [Lau94, MPN+02]. But the view-dependency is useless for reducing such visual artifacts as double imaging caused by the deficiency of sampling [CTCS00] since it is the fundamental limitation of image-based rendering as explained in Chapter 1.

One possible approach to this problem is to incorporate the method with the rendering techniques based on a sparse images, such as that proposed in Chapter 2 and 3.

In the algorithm of the microfacet billboarding, the process of synthesizing view-dependent textures by linear interpolation can be replaced with image morphing based

on automatic image matching. To accomplish the combination, the matching algorithm proposed in Section 2.2 has to be extended in two ways. First, it is desirable for the matching algorithm to use the geometric model as the initial estimation of the image correspondence, even if it is unreliable. Second, it is necessary to extend the matching and morphing algorithms so that they can appropriately handle the correspondence problem between more than two images.

On the other hand, in the system of the pop-up light field system, the knowledge of the scene geometry can be used to aid in the initialization and verification of the coherent layer construction. For example, when the rough depth map for the scene to be rendered is obtained by applying stereo reconstruction to the light field images, the image segmentation based on the depth map can be utilized as the initial boundary of the coherent layers. It is likely easier to improve the initial boundaries rather than to create them from scratch. Once a layer has been popped up, it can be determined whether the layered object can be approximated using simple planar geometry by verifying that the variance of the pixel depths within the layer is sufficiently small.

Compression of created model

One of the problems in modeling and rendering from a measured data set is that the total size of the created model tends to become huge. Although all of the proposed methods aim to render scenes from relatively sparse or under-sampled data sets, the size of data can easily explode when additional dimensions in data space, such as motions and illumination changes, are taken into account.

From the analogies to the methods for the appearance compression [NSI01, WAA+00], it may also be possible to compress the size of texture images in microfacet billboarding, since the texture images mapped on a microfacet are supposed to have coherency between views to some extent. In contrast to above referred methods, our geometric model can be fairly inaccurate, which makes it difficult to reduce the dimension of texture images by applying the principal component analysis (PCA) in texture space. Thus, it is interesting to analyze the trade-off relationship between the accuracy of the geometric model and the compressibility of photometric model.

It is also interesting to utilize the image correspondence estimated by the method proposed in Chapter 2 for compressing the size of reference images in the same manner as is done in the block matching technique for compressing video sequences. A part of the pixels in reference images can be reduced by propagating the pixels in the nearby view-

images using the per-pixel correspondence. In addition, it is also interesting to develop a technique that would reduce the resolution of the mapping with the least loss of visual quality in the interpolated sequences, since the per-pixel correspondence between images may be unnecessary for rendering purposes and inappropriate for compression,

Various extension of the Pop-up Light Field

Since our pop-up light field takes on a sparse light field as input, the application has several limitations, which are as follows. It cannot handle specular highlights and other significant appearance changes. It also assumes that each layer itself can be rendered correctly by the sparse light field. Our method cannot be guaranteed alias-free if the layer boundaries are too complicated for the user to segment accurately, or a single object violates plenoptic sampling criterion. Our layer-based approach does not handle topological changes properly. Moreover, coherent matting does not work well for semi-transparent surfaces and long hair because the prior $L(\alpha)$ used in our formulation is approximately modeled as a point spread function. The solutions to all these problems remain as future work.

BIBLIOGRAPHY

[AB91] Edward H. Adelson and James R. Bergen. *Computational Models of Visual Processing*, chapter The plenoptic function and the elements of early vision. MIT Press, 1991.

[BB89] H.H. Baker and R.C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *International Journal of Computer Vision*, 3(1):33 – 39, 1989.

[BB95] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM Computing Survey*, 27(3):433–467, 1995.

[BBM+01a] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Proc. SIGGRAPH 2001*, pages 425–432, 2001.

[BBM01b] Chris Buehler, Michael Bosse, and Leonard McMillan. Non-metric image-based rendering for video stabilization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2001.

[BBM+01c] Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen. Unstructured lumigraph rendering. In *Proc. SIGGRAPH '01*, pages 425–432, 2001.

[BG01] Samuel Boivin and Andre Gagalowicz. Image-based rendering of diffuse, specular and glossy surfaces from a single image. In *Proc. SIGGRAPH 2001*, pages 107–116, 2001.

[BN92] Thaddeus Beir and Shawn Neely. Feature-based image metamorphosis. In *SIGGRAPH 92 Conference Proceedings*, page 35. ACM, July 1992.

[BSA98] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *Proc. CVPR'98*, pages 434–441, 1998.

[BWK02] Marlo Botsch, Andreas Whatanaya, and Leif Kobbelt. Efficient high quality rendering of point sampled geometry. In *Proc. 13th Eurographics Workshop on Rendering*, pages 53–64, 2002.

[CAC+02] Yung-Yu Chuang, Aseem Agarwala, Brian Curless, David H. Salesin, and Richard Szeliski. Video matting of complex scenes. In *Proc. SIGGRAPH 2002*, pages 243–248, 2002.

[CBCG02] Wei-Chao Chen, Jean-Yves Bouguet, Michael H. Chu, and Radek Grzeszczuk. Light field mapping: Efficient representation and hardware rendering of surface light fields. *ACM Transactions on Graphics*, 21(3):447–456, 2002.

[CBL99] Chun-Fa Chang, Gary Bishop, and Anselmo Lastra. LDI tree: A hierarchical representation for image-based rendering. In *Siggraph 1999, Computer Graphics Proceedings*, pages 291–298, 1999.

[CCSS01] Yung-Yu Chuang, Brian Curless, David H. Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *Proc. CVPR 2001*, volume 2, pages 264–271, 2001.

[CET98] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. In *Proc. ECCV '98*, pages 484–498, 1998.

[Che95] Shenchang Eric Chen. QuickTime VR — an image-based approach to virtual environment navigation. In *Proc. SIGGRAPH '95*, pages 29–38, 1995.

[Chi94] Stephen L. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2(3):267–278, September 1994.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH '96*, pages 303–312, 1996.

[CLF02] Umberto Castellani, Salvatore Livatino, and Robert B. Fisher. Improving environment modelling by edge occlusion surface completion. In *Proc. International Symposium on 3D Data Processing Visualization and Transmission*, pages 672–675, 2002.

[Cor] Corel Corporation. http://www.corel.com.

[CRZ99] A. Criminisi, I.D. Reid, and A. Zisserman. Single view metrology. In *Proc. International Conference on Computer Vision '99*, pages 434–441, 1999.

[CT81] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. In *Proc. SIGGRAPH '81*, pages 307–316, 1981.

[CTCS00] Jinxiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. In *Proc. SIGGRAPH 2000*, pages 307–318, 2000.

[CW93] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proc. SIGGRAPH '93*, pages 279–288, 1993.

[DMGL02] James Davis, Steven R. Marschner, Matt Garr, and Marc Levoy. Filling holes in complex surfaces using volumetric diffusion. In *Proc. International Symposium on 3D Data Processing Visualization and Transmission*, pages 428–438, 2002.

[DSTT00] Frank Dellaert, Steven Seitz, Charles Thorpe, and Sebastian Thrun. Structure from motion without correspondence. In *Proc. CVPR 2000*, pages 557–564, 2000.

[DT96] Fred W. DePiero and Mohan M. Trivedi. 3-d computer vision using structured light: Design, calibration, and implementation issues. *Advances in Computers*, 43:243–278, 1996.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proc. SIGGRAPH '96*, pages 11–20, 1996.

[DVS03] Carsten Dachsbacher, Christian Vogelgsang, and Marc Stamminger. Sequential point trees. *ACM Trans. Graph.*, 22(3):657–662, 2003.

[Fau93] O. Faugeras. *Three-dimensional computer vision: A geometric viewpoint*. The MIT press,, 1993.

[FK98] Olivier Faugeras and Renaud Keriven. Complete dense stereovision using level set methods. In *Proc. ECCV '98*, pages 379–393, 1998.

[GD98] J. P. Grossman and William J. Dally. Point sample rendering. In *9th Eurographics Workshop on Rendering*, pages 181–192, 1998.

[GGSC96] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Proc. SIGGRAPH '96*, pages 43–54, 1996.

[GH97] R. Gupta and R. I. Hartley. Linear pushbroom cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9), 1997.

[GMP] GMP, GNU Multiple Precision Arithmetic Library. http://www.swox.com/gmp/.

[HAA97] Youichi Horry, Ken-Ichi Anjyo, and Kiyoshi Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *Proc. SIGGRAPH 1997*, pages 225–232, 1997.

[HB89] Berthold K. P. Horn and M. J. Brooks. *Shape From Shading.* MIT Press, 1989.

[HHR01] Peter Hillman, John Hannah, and David Renshaw. Alpha channel estimation in high resolution image and image sequences. In *Proc. CVPR 2001*, volume 1, pages 1063–1068, 2001.

[HKP+99] B. Heigl, R. Koch, M. Pollefeys, J. Denzler, and L. Van Gool. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In *Symposium fur Mustererkennung*, pages 94–101, 1999.

[HRRS86] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley, 1986.

[HSIW96] Adrian Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *Proc. European Conference on Computer Vision*, pages 117–126, 1996.

[IMG00] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. In *Siggraph 2000, Computer Graphics Proceedings*, pages 297–306, 2000.

[IS01] Katsushi Ikeuchi and Yoichi Sato, editors. *Modeling from Reality*. Kluwer, 2001.

[ISN+00] Katsushi Ikeuchi, Yoichi Sato, Ko Nishino, Ryusuke Sagawa, Takuji Nishikawa, Takeshi Oishi, Imari Sato, Jun Takamatsu, and Daisuke Miyazaki. Modeling cultural heritage through observation. In *Proc. IEEE First Pacific-Rim Conference on Multimedia*, 2000.

[JM92] David Jones and Jitendra Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *Proc. ECCV '92*, pages 395–410, 1992.

[KS99] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. In *Proc. International Conference on Computer Vision '99*, pages 307–314, September 1999.

[KSC01] Sing Bing Kang, Richard Szeliski, and Jinxiang Chai. Handling occlusions in dense multi-view stereo. In *Proc. CVPR 2001*, pages I:103–110, 2001.

[KZ02] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *Proc. European Conference on Computer Vision*, pages 82–97, 2002.

[Lau94] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.

[LC87] William Lorensen and Harvey Cline. Marching cubes: a high resolution 3D surface reconstruction algorithm. In *Proc. SIGGRAPH '87*, pages 163–169, 1987.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proc. SIGGRAPH '96*, pages 31–42, 1996.

[LLSY02] Ziqian Liu, Ce Liu, Heung-Yeung Shum, and Yizhou Yu. Pattern-based texture metamorphosis. In *Proc. Pacific Graphics 2002*, pages 184–191, 2002.

[LPC+00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3D scanning of large statues. In *Proc. SIGGRAPH 2000*, pages 131–144, 2000.

[LS97] Jed Lengyel and John Snyder. Rendering with coherent layers. In *Proc. SIGGRAPH 1997*, pages 233 – 242, 1997.

[LW85] Marc Levoy and Turner Whitted. The use of points as a display primitive. Technical Report 85-022, Computer Science Department, University of North Carolina at Chapel Hill, 1985.

[MB95a] Leonard McMillan and Gary Bishop. Head-tracked stereoscopic display using image warping. In *Proc. SPIE*, pages 21–30, 1995.

[MB95b] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Proc. SIGGRAPH '95*, pages 39–46, 1995.

[MBR+00a] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *Proc SIGGRAPH '00*, pages 369–374, 2000.

[MBR+00b] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *Proc. SIGGRAPH 2000*, pages 369–374, 2000.

[MBSL99] Jitendra Malik, Serge Belongie, Jianbo Shi, and Thomas Leung. Textons, contours and regions: Cue integration in image segmentation. In *Proc. International Conference on Computer Vision '99*, pages 918–925, 1999.

[MCDD01] Byong Mok, Max Chen, Julie Dorsey, and Frédo Durand. Image-based modeling and photo editing. In *Proc. SIGGRAPH 2001*, pages 433–442, 2001.

[Mil75] David L. Milgram. Computer methods for creating photomosaics. *IEEE Transactions on Computer*, 24:1113–1119, 1975.

[Moo96] Todd K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, pages 47–60, November 1996.

[MPN+02] Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. Image-based 3D photography using opacity hulls. In *Proc. SIGGRAPH 2002*, pages 427–437, 2002.

[MV98] J. B. Antoine Maintz and Max A. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36, 1998.

[NI02] Ko Nishino and Katsushi Ikeuchi. Robust simultaneous registration of multiple range images. In *Proc. ACCV '02*, pages 454–461, 2002.

[NMP96] Chahab Nastar, Baback Moghaddam, and Alex Pentland. Generalized image matching: Statistical learning of physically-based deformations. In *Proc. ECCV '96*, pages 589–598, 1996.

[NSI99] Ko Nishino, Yoichi Sato, and Katsushi Ikeuchi. Eigen-texture method: Appearance compression based on 3D model. In *Proc. CVPR '99*, volume 1, pages 618–624, June 1999.

[NSI01] Ko Nishino, Yoichi Sato, and Katsushi Ikeuchi. Eigen-texture method: Appearance compression and synthesis based on a 3D model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1257–1265, 2001.

[OBM00] Manuel M. Oliveira, Gary Bishop, and David McAllister. Relief texture mapping. In *Proc. SIGGRAPH 2000*, pages 359–368, 2000.

[PD84] Thomas Porter and Tom Duff. Compositing digital images. In *Proc. SIGGRAPH '84*, pages 253–259, 1984.

[PZvBG00] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proc. SIGGRAPH 2000*, pages 335–342, 2000.

[Rad99] Paul Rademacher. View-dependent geometry. In *Proc. SIGGRAPH 1999*, pages 439–446, 1999.

[RH01] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *Proc. SIGGRAPH 2001*, pages 117–128, 2001.

[RHHL02] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3D model acquisition. In *Proc. SIGGRAPH 2002*, pages 438–446, 2002.

[RL00a] Szymon Rusinkiewicz and Marc Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proc. SIGGRAPH 2000*, pages 343–352, 2000.

[RL00b] Szymon Rusinkiewicz and Marc Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proc. SIGGRAPH 2000*, pages 343–352, 2000.

[RT00] Mark A. Ruzon and Carlo Tomasi. Alpha estimation in natural images. In *Proc. CVPR 2000*, pages 24–31, 2000.

[SB96] Alvy Ray Smith and James F. Blinn. Blue screen matting. In *Proc. SIGGRAPH 1996*, pages 259–268, 1996.

[SB97] Stephen M. Smith and J. Michael Brady. SUSAN - A new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[Sch95] Gernot Schaufler. Dynamically generated impostors. In *Proc. GI Workshop on Modeling, Virtual Worlds, Distributed Graphics*, pages 129–135, 1995.

[Sch98a] Gernot Schaufler. Image-based object representation by layered impostors. In *Proc. Symposium on Virtual Reality Software and Technology*, pages 99–104, 1998.

[Sch98b] Gernot Schaufler. Per-object image warping with layered impostors. In *Proc. the 9th Eurographics Workshop on Rendering*, pages 145–156, 1998.

[SD96] Steven M. Seitz and Charles R. Dyer. View morphing. In *Proc. SIGGRAPH '96*, pages 21–30, 1996.

[SGwHS98] Jonathan Shade, Steven Gortler, Li wei He, and Richard Szeliski. Layered depth images. In *Proc. SIGGRAPH 1998*, pages 231–242, 1998.

[SH99] Heung-Yeung Shum and Li-Wei He. Rendering with concentric mosaics. In *Proc. SIGGRAPH'99*, pages 299–306, 1999.

[SK98a] Steven M. Seitz and Kiriakos N. Kutula. Plenoptic image editing. In *Proc. International Conference on Computer Vision '98*, pages 17–24, 1998.

[SK98b] Yoshihisa Shinagawa and Tosiyasu L. Kunii. Unconstrained automatic image matching using multiresolutional critical-point filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9):994–1010, 1998.

[SS97] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proc. SIGGRAPH 1997*, pages 251–258, 1997.

[SSY+ar] Heung-Yeung Shum, Jian Sun, Shuntaro Yamazaki, Li Yin, and Chi-Keung Tang. Pop-up light field. *ACM Transactions on Graphics*, To appear.

[SWCT02] Heung-Yeung Shum, Lifeng Wang, Jin-Xiang Chai, and Xin Tong. Rendering with manifold hopping. *International Journal of Computer Vision,*, 50(2):185–201, 2002.

[SWI97] Yoichi Sato, Mark D. Wheeler, and Katsushi Ikeuchi. Object shape and reflectance modeling from observation. In *Proc. SIGGRAPH '97*, pages 379–387, 1997.

[TK92] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: A factorization method. *Int. Journal of Computer Vision*, 9(2):137–154, 1992.

[TS67] Kenneth E. Torrance and Ephraim M. Sparrow. Theory for off-specular reflection from roughened surfaces. *Optical Society of America*, pages 1105–1114, 1967.

[WAA+00] Daniel Wood, Daniel Azuma, Wyvern Aldinger, Brian Curless, Tom Duchamp, David Salesin, and Werner Steutzle. Surface light fields for 3D photography. pages 287–296, 2000.

[Wol90] George Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 1990.

[Woo80] Robert J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.

[Woo89] Robert J. Woodham. Determining surface curvature with photometric stereo. In *Proc. Robotics and Automation*, pages 36–42, 1989.

[WSI98] Mark D. Wheeler, Yoichi Sato, and Katsushi Ikeuchi. Consensus surfaces for modeling 3D objects from multiple range images. In *Proc. International Conference on Computer Vision '98*, pages 917–924, 1998.

[YDMH99] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: recovering reflectance models of real scenes from photographs. In *Proc. SIGGRAPH '99*, pages 215–224, 1999.

[You85] Richard A. Young. The gaussian derivative theory of spatial vision: analysis of cortical cell receptive field weighting profiles. Technical Report GMR 4920, General Motors Research, 1985.

[YSK+02] Shuntaro Yamazaki, Ryusuke Sagawa, Hiroshi Kawasaki, Katsushi Ikeuchi, and Masao Sakauchi. Microfacet billboarding. In *Proc. the 13th Eurographics Workshop on Rendering*, pages 175–186, 2002.

[ZC03] Cha Zhang and Tsuhan Chen. A survey on image-based rendering. Technical Report AMP 03-03, Carnegie Mellon University, 2003.

[ZCar] Cha Zhang and Tsuhan Chen. Spectral analysis for sampling image-based rendering data. *IEEE Transactions on CSVT*, to appear.

[ZWGS02] Zhunping Zhang, Lifeng Wang, Baining Guo, and Heung-Yeung Shum. Feature-based light field morphing. In *Proc. SIGGRAPH*, pages 457–464, 2002.

Index

R

S

T

U

V

W