Free Viewpoint Video Synthesis from Uncalibrated Cameras

September 2009

Songkran Jarusirisawad

A Thesis for the Degree of Ph.D. in Engineering

Free Viewpoint Video Synthesis from Uncalibrated Cameras

September 2009

Graduate School of Science and Technology
Keio University

Songkran Jarusirisawad

Acknowledgement

This Ph.D. thesis is the result of works during which I have been accompanied and supported by many people. It is now my great pleasure to take this opportunity to thank them.

First of all, I would like to thank my Master's and Doctoral thesis supervisor, Professor Hideo Saito, who influenced me most during my study at Keio University. He is not only a great professor with deep vision but also and most importantly a kind person. He provided me unflinching encouragement and support in various ways. Without his supervision, this work would not have been completed.

I gratefully thank Professor Masaaki Ikehara, Professor Issei Fujishiro, and Associate Professor Hiroshi Shigeno that in the midst of all their activities, they accepted to be the members of my thesis committees. I have also benefited enormously from their constructive comments.

I also want to express my gratitude to Dr. Vincent Nozick and Dr. Julien Pilet for their advices and tirelessly helps. Their excellent ideas and source code tremendously reduce my time to achieve a good research. I would like to thank Dr. Yuko Uematsu for her assistance with proof-reading and suggestions during this thesis preparation.

I would like to acknowledge a generous financial support from the Japanese Government, Ministry of Education, Culture, Sports, Science and Technology (MEXT) for supporting me during four years of study in Japan.

This work would not have been possible without Chutisant Kerdvibulvech, Anan Jeenanong, Naoyuki Yazawa, Sandy Eggi, Haruka Asai, Hiroaki Kubo, Takahide Hosokawa and Yuki Takaya, colleagues who helped me capturing input videos. I owe deep thanks to Zachary Garrett for proof-reading my paper. In my daily life I

also have been blessed with a friendly and cheerful group of fellows, Tomoaki Teshima, Ryo Hirose (and his family), Yuji Oyamada, Hideaki Uchiyama, and other members in Saito Laboratory.

Last but not least, my deepest thanks go to my parents and my sister for their unconditional support, understanding, and encouragement during my long period away from home.

Abstract

Conventional 2D video provides a fixed viewpoint of the recorded event that viewers can only see a video playback passively. Viewpoint of a video playback is always the same as how the scene was recorded. In contrast, free viewpoint video generates an output video at the virtual views selected by the viewer. This means that each viewer of the same content may be observing from a unique viewpoint. Most of the proposed methods for creating free viewpoint video usually assume that cameras are strongly calibrated, i.e. camera's internal parameters such as optical axis, focal length are known. The contribution of this thesis is the using of uncalibrated cameras for free viewpoint video synthesis based on projective grid space (PGS), a weak calibration framework that provide geometrical relationship between cameras from fundamental matrices and trifocal tensors among views. We proposed an automatic method for weakly calibrating pure rotating and zooming cameras which are commonly used when capturing videos in a large scene. We also proposed a new plane-sweep algorithm in PGS which can reconstruct and render new view images in real-time. These proposed methods are demonstrated in two systems with different scene constraints and cameras setup.

The first system is targeted for a large natural scene where cameras' rotating and zooming to capture the interested part of a scene for a higher resolution of a moving object is necessary. In this case, the cameras must be dynamically calibrated due to the changing in focal length and camera rotation. Automatically extracted 2D-2D corresponding points between the current frame and the initial frame are used for recomputing trifocal tensors of the zoomed and rotated cameras in order to weakly calibrate cameras to PGS. The 3D structure of moving object is reconstructed using

silhouette volume intersection method. Background areas which are approximated as several planes are rendered together with foreground moving object using view interpolation. Four hand-held video cameras are used to capture the input videos. The experimental results show that the proposed method is efficient, even when it is applied to the hand-held cameras with a small movement.

The second system is targeted for a small area where cameras are fixed for the duration of the capturing. We use our proposed plane-sweep algorithm in PGS for reconstruct and rendering free viewpoint videos. Thanks to the implementation of plane-sweep algorithm in Graphics Processing Unit (GPU), the system can achieve real-time reconstruction and rendering. Viewer can choose their own viewpoint between any cameras interactively from live input video cameras or from prerecorded videos. In our experiment, free viewpoint video can be rendered at a real-time frame rates using five web cameras with 320x240 resolutions.

The advantage of our proposed free viewpoint video synthesis from uncalibrated cameras using PGS comparing to the other methods that use strong calibration is the easiness of cameras calibration. Fundamental matrices and trifocal tensors, which represent the geometry between cameras in PGS, can be estimated from 2D-2D correspondences. In the strong calibration methods, 3D-2D correspondences are necessary for estimating projection matrices. These correspondences are difficult to be measured precisely from a large natural scene or when the cameras are not fixed during video capture. From the experiments, our proposed method is suitable to apply with various scenarios, e.g., with respect to camera setup (small baseline vs. wide baseline), processing performance (on-line vs. off-line) and scene areas (small area vs. large area).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

Conventional 2D video provides a fixed viewpoint of the recorded event whose viewers can only see a video playback passively. The viewpoint of a video playback remains the same as how the scene was recorded. In contrast, free viewpoint video is a system for viewing video allowing the user to control the viewpoint and generate new views of a real-world dynamic scene from the desired 3D position. This means that each viewer of the same content may be observing from a unique viewpoint.

Free viewpoint video effects can be seen in recent films or TV broadcasting. Most people might have seen a famous bullet-time effect from movie "The Matrix" [2] where the camera is moving around the actor at the same time as bullets are shoot. This is impossible with conventional slow-motion, as the physical camera would have to move impossibly fast. In The Matrix, the camera path was pre-designed. Cameras were arranged, behind a green or blue screen, on a designed path, forming a complex curve through space. The cameras were then triggered at extremely close intervals, so the action continued to unfold, in extreme slow-motion, while the viewpoint moved.

Free viewpoint video is also used in sports broadcasting. "EyeVision" [1] system was used for the Super Bowl XXXV broadcast by CBS. Multiple video streams are captured using more than 30 custom-built, robotic pan tilt zoom cameras. These cameras were controlled in concert so that cameras pointed, zoomed and focused synchronously on the same spot on the field. The sequence of video images from different angles are arranged to create virtual camera movements such that viewers can feel revolving around the object at a temporally frozen moment.

Both systems from "The Matrix" and "EyeVision" create free viewpoint videos by simply switching the capture images. Each frame from output video comes from the real camera viewpoints which requires a large number of cameras for generating view change smoothly. The goal of free viewpoint video research is to use computer to generate virtual views to produce the same effect when using a much smaller number of cameras with less effort and cost.

Free viewpoint video is an interdisciplinary research area of computer vision and computer graphics. From multiple input videos at several viewpoints, the geometrical

relation between cameras are estimated and the 3D information of a scene is reconstructed. The 3D reconstructed data and input images provide information to render a realistic image of a scene from the desired viewpoint. This process is so-called image based modeling and image based rendering.

One of the earliest researches of free viewpoint video of a dynamic scene is "Virtualized Reality" [41]. In that research, 51 cameras are placed around hemispherical dome called a 3D Room. The 3D model of an object in a target scene is reconstructed from multiple view images. The colors in real images are used to synthesize the texture of the 3D model. Using conventional rendering techniques, new view images are generated from the color-textured 3D model.

Systems for rendering arbitrary views of natural scenes have become to the interest of computer vision community for a long time. However, the research has reached only in recent years a level worth considering for an end user system, due to advancements in image acquisition hardware, higher capacity drives, and faster PCs. Most of the proposed methods for creating free viewpoint video usually assume that cameras are strongly calibrated, i.e. camera's internal parameters such as optical axis, focal length are known. In this thesis, we propose two novel methods for creating free viewpoint video from multiple uncalibrated cameras based on projective grid space (PGS) framework [75].

In the first system, we propose a method for synthesizing free viewpoint of dynamic events in natural scene from uncalibrated pure rotating and zooming cameras [36, 38, 39]. Using pure rotating and zooming cameras is more flexible in terms of video acquisition. In case that the scene is a large space, cameraman can zoom and capture only some part of a scene to get higher resolution of a moving object, e.g. soccer players in a field, baseball players, etc. In this case, all cameras must be dynamically calibrated at every frame. In the proposed method, we compute projective geometry between cameras from the homography with the initial frame. The 3D structure of moving object is reconstructed using silhouette volume intersection method [47]. Background areas, approximated as several planes, are rendered together with foreground moving object using view interpolation [7, 11, 82]. The proposed scheme is suitable for a large area environment with a wide base-line between cameras.

In the second system, we proposed a real-time method for generating free viewpoint video from uncalibrated cameras. We use PGS as a weak calibration framework in both systems. However, in this real-time system, we use our proposed plane-sweep algorithm in PGS for 3D reconstruction and rendering free viewpoint videos. Thanks to the implementation of our algorithm on Graphics Processing Unit (GPU), our system can achieve real-time rendering. In this system, viewers can choose their own viewpoint between any cameras interactively. This system is suitable for a smaller area environment with a small base-line between cameras.

1.2 Overview of the Approach

This thesis proposes novel methods of virtual view synthesis for dynamic events from multiple uncalibrated cameras. Our contribution is the use of uncalibrated cameras for free viewpoint video synthesis based on using projective grid space (PGS) [75] framework. We propose two systems as listed below, using different 3D reconstruction and rendering algorithms targeted for different scene constraints.

- Free Viewpoint Video from Pure Rotating and Zooming Cameras (Chapter 4)
- Realtime Free Viewpoint Video using Plane-Sweep Algorithm (Chapter 5)

The first system is targeted for a large environment where cameras should be rotated and zoomed to get a higher resolution image of the interested part of a scene. For example, in the soccer field or baseball field, we can zoom cameras and capture only the part where the ball is. In this case, cameras are not fixed, geometrical relation between multiple cameras are changed every frame. We propose a novel method for weakly calibrating these pure rotating and zooming cameras for virtual view synthesis. The background scene are approximated as several planes are rendered together with a moving object in the output virtual view video.

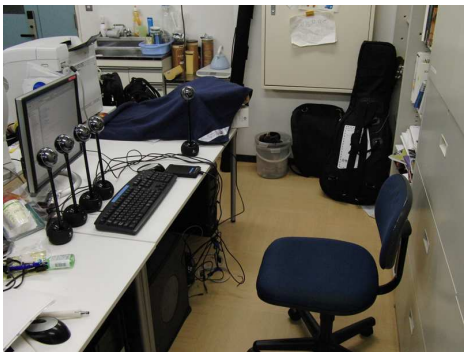
The second system is targeted for a smaller environment where base-line between cameras is small. We assume that all cameras in this system are static. Geometrical relations among cameras are estimated only once before the video acquisition starts. We propose a novel plane-sweep algorithm in PGS for generating virtual views in real-time. Our plane-sweep algorithm is implemented on GPU which can reduce a computation time so that it become an online system.

We summarize the differences between two proposed systems in Table 1.1. Figure 1.1 and 1.2 show cameras configuration and targeted scenes in both systems.

(a) Camera configuration

(b) Example targeted scene

Figure 1.1: Cameras configuration and the example of targeted scenes for viewpoint video system using uncalibrated rotating and zooming cameras (Chapter 4)



(a) Camera configuration

(b) Example targeted scene

Figure 1.2: Cameras configuration and the example of targeted scenes for real-time free viewpoint video system using plane-sweep algorithm (Chapter 5)

	Free viewpoint video system	
	Rotating and zooming cameras (Chapter 4)	Real-time plane-sweep (Chapter 5)
Processing time	Off-line	Online
Targeted scene	Large area	Small area
Camera movement	Pure rotating and zoom	Fixed
3D Reconstruction	Visual hull	Plane-sweep
Background scene	Approximated as planes	Any scene

Table 1.1: Summary of the differences between two proposed systems.

1.3 Thesis Outline

The outline of this thesis is as follows. In Chapter 2, related works are discussed. We firstly survey on image based modeling and rendering techniques which are categorized according to how much geometric information is used. Then, we survey on video based modeling and rendering techniques which are off-line or online systems targeted for multiple videos input.

Chapter 3 presents a projective grid space (PGS) [75] which is a weak calibration framework we used in our proposed systems. We extend originally proposed PGS by using trifocal tensors for relating non-basis cameras instead of fundamental matrices. This extension increases stability of calibration and allows more general camera configuration. We describe both original proposed scheme and our extension in this chapter.

Chapter 4 presents our proposed method for virtual view synthesis of dynamic scenes using pure rotating and zooming cameras. The method to dynamically calibrate rotated and zoomed cameras to PGS is described. Experimental results show realistic scenes at virtual viewpoints generated by our system. The effectiveness of the proposed method is evaluated both qualitatively and quantitatively.

Chapter 5 introduces our online system for generating free viewpoint videos from uncalibrated cameras in real-time. Our novel plane-sweep algorithm is described in detail first. Then, we explain the implementation of this algorithm on GPU for achieving real-time rendering. Qualitative and quantitative evaluation are presented in the end of this chapter.

Chapter 6 summarizes contributions of this work. The possible extensions are also discussed.

Chapter 2

Related Works

2.1 Image Based Rendering

The traditional approach of computer graphics for rendering an image has been to manually create a geometric model in 3D and reproject it onto a 2D image. The modeling stage is a tedious and time consuming process which could require days or even a month in case of a complex scene. Image-based modeling and rendering is a computer vision technique to analyze multiple images and automatically build 3D model for generating novel 2D images.

Over the last decade, various image-based modeling and image-based rendering (IBR) techniques have been developed [87, 85, 106]. The early works on IBR focused on static scenes, mostly due to hardware limitations in image capture, processing and storage. IBR techniques can be categorized into three categories based on how geometric-centric the scene representation is, which are rendering without geometry, rendering with explicit geometry (either with approximate or accurate geometry), and rendering with implicit geometry (i.e., from correspondences) . The second one, rendering with explicit geometry, is commonly termed as model-based approach while the third one, rendering with implicit geometry, is termed as transfer-based approach. Figure 2.1 depicts categories of these techniques.

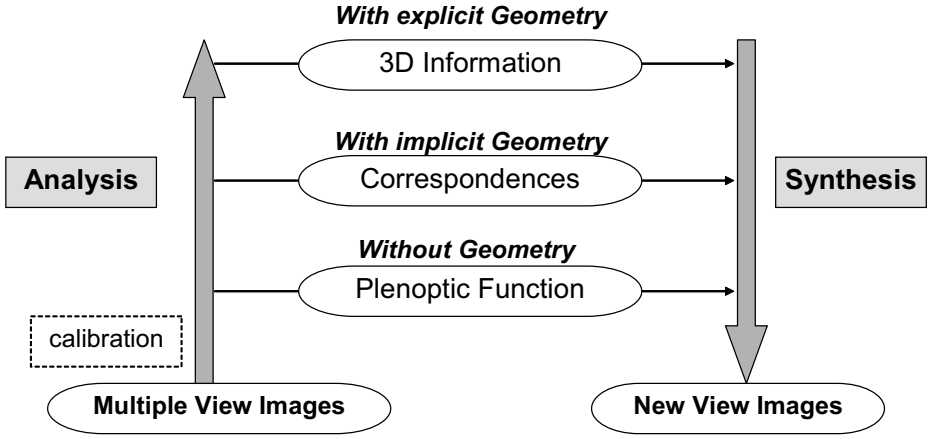


Figure 2.1: Image-based modeling and image-based rendering

Our proposed method in chapter 4 [39] is a combination between model-based approach and transfer-based approach. The reconstructed 3D model in projective grid

space (PGS) is used for making dense correspondences between two reference images. Then image interpolation can be performed as conventional transfer-based approach. The benefit of reconstructing a 3D model in our method is to make correspondence automatically and so handle the occluded areas.

The method proposed in chapter 5 can be a categorized as a model-based approach. In this system, we use our proposed plane-sweep algorithm in PGS [35, 40]. The plane-sweep algorithm recover the depth information of a scene and renders a new view at the same time.

2.1.1 Rendering without Geometry

The techniques for rendering without geometry rely on the characterization of the plenoptic function, which describes all the radiant energy perceived by an observer at any point in space and time. These techniques do not require any geometric information or correspondences to create novel views but they do require a large amount of input images compared to other techniques.

In the most general forms, the *plenoptic function* [4] (from *plenus*, complete or full, and *optic*) is defined as the intensity of light rays passing through the camera center at every 3D location (V_x, V_y, V_z) at every possible angle (θ,ϕ) for every wavelength λ, at every time t, i.e., $P_7(V_x, V_y, V_z, \theta, \phi, \lambda, t)$.

McMillan and Bishop proposed an image-based rendering approach for a static scene named *plenoptic modeling*. Their method is based on the idea of *plenoptic function* but removing two variables, time t and light wavelength λ.

The light field rendering of Levoy and Hanrahan [50] and the Lumigraph of Gortler et al. [23] simplified the plenoptic function to four dimensions. Both works are based on the idea that as long as novel views are render outside the convex hull of an object and the medium is non-dispersive, light rays can be described by their intersections with two planes in arbitrary position. By convention, the coordinate system on the first plane is (u, v) and on the second plane is (s, t). The plenoptic function in these works are simplified to a 4D light field function $P_4(u, v, s, t)$, where (u, v) and (s, t) are parameters of two planes used for describing the light ray.

The concentric mosaics (CMs) proposed by Shum and He [86] reduces the amount of data by limiting capturing positions of input images along a circle path. They define a 3D plenoptic function using three parameters for rotation angle, radius and vertical elevation. Capturing CMs is easy by simply spinning the camera on a rig which takes only several minutes. This makes CMs attractive for many virtual reality applications.

A complete plenoptic function at a fixed viewpoint can be constructed from incomplete samples. Specifically, a panoramic mosaic is constructed by registering multiple regular images. Many systems have been built to construct cylindrical and spherical panoramas by stitching multiple images together [10, 60, 91, 92]. Szeliski and Shum [92] presented a complete system for constructing panoramic image mosaics from sequence of images. Their mosaic representation associates a transformation matrix with each input image rather than explicitly projecting all of the images onto a common surface, such as a cylinder.

Table 2.1 compares the mentioned methods. Those for rendering without geometry provide a much better image quality and lower computation cost for rendering virtual views compared to the model-based approaches described in the next section. However, a large amount of input images taken at slightly different positions are necessary. This main drawback makes these approaches not suitable for large scale environments.

Name	View space	Dimension
Plenoptic function	any position	7
Plenoptic modeling	any position	5
Light field / Lumigraph	bounding box	4
Concentric Mosaics	bounding circle	3
Cylindrical / Spherical panorama	fixed point	2

Table 2.1: Summary of the rendering techniques based on Plenoptic function.

2.1.2 Rendering with Explicit Geometry

The techniques in this category, so-called model based, use direct 3D information reconstructed from the input images. 3D information of a scene is in either the form of 3D coordinates or depth along lines-of-sight. The key technology is shape reconstruction of the objects from multiple view images.

The straightforward technique in this category is using a traditional 3D model with a single texture mapping [67, 98]. After 3D model is reconstructed, conventional graphics rendering pipeline can be used to render new view images directly.

Using single texture-mapped model can not capture visual effects such as highlights, reflections, and transparency. To obtain these visual effects of a reconstructed architectural environment, Debevec et al. [15, 14] used view-dependent texture mapping to render new views by warping and compositing several input images of an environment. Advantage over conventional texture mapping is that multiple textures from different sampled viewpoints are warped to the same surface and combined with weights computed based on proximity of the rendered viewpoint to the sampled viewpoints.

Buehler et al. [8] apply a more principled approach of blending textures based on relative angular position, resolution, and field of view. Kang and Szeliski [43] used view-dependent depth maps, which capture not only view-dependent textures but also view-dependent geometries, from a collection of images to render new views.

Billboards is a rendering technique commonly used in games for representing a complex objects such as plants or trees and also used for image-based rendering of a natural scene. Billboards are either single texture-mapped rectangles that remain parallel to the image plane, or sets of two rectangles arranged in a cross. The advantage of this technique is the ease of rendering using traditional graphic pipeline but they typically give a realistic appearance only when the object is almost a planar or when being viewed from a far distance.

Hayashi and Saito [29] proposed a free viewpoint system for soccer scene which represents the soccer players as billboards. They also introduced the choice of optimal camera for using as a billboard texture. Decoret et al. [16] proposed *billboard clouds* which is a set of textured, partially transparent polygons (billboard), with

independent size, orientation and texture resolution. An optimization algorithm is used to build a billboard clouds given a geometric error threshold. Billboard clouds give a better visual appearance comparing to ordinary billboards because they can provide appropriate parallax and silhouettes.

When depth information is available for every point in one or more images, 3D warping techniques [65, 62, 69] can be used to render image at new viewpoints. An image can be rendered from any nearby point of view by projecting the pixels of the original image to their proper 3D locations and re-projecting them onto the new image. The most significant problem of 3D warping is holes in the warped image, due to the difference of sampling resolution between the input and output images, and also to the occlusions where parts of the scene is not visible in the input images.

To deal with the disocclusion artifacts in 3D warping , Shade et al. [84] proposed Layered Depth Images (LDIs) which store multiple depth pixels at each discrete location in the image. Thus, when rendering from a LDI, the previously occluded regions from the upper layer can still be rendered from data stored in the lower layer of a layered depth pixel.

Saito et al. [74] proposed *Appearance-Based Virtual-View Generation* which is a hybrid approach between model-based approach and transfer-based approach. The reconstructed model, from *multiple baseline stereo* (MBS) [72] and shape from silhouettes [12, 73], is used for making dense correspondences between the two original views. Then, conventional image interpolation techniques [11, 82] can be used to render the novel views.

Yaguchi and Saito [102] extended appearance-based method to use with uncalibrated cameras. They use projective grid space (PGS) [75], which has a special 3D coordinate system established by epipolar geometry of cameras, for 3D reconstruction without cameras calibration. Our method [39] presented in chapter 4 extends this work to the uncalibrated pure rotating and zooming cameras. We also extend the original projective grid space framework from using fundamental matrices to using trifocal tensors. This gives more numerically stable points transfer between cameras and can be used with more general cameras settings as we discuss in detail in chapter 3.

2.1.3 Rendering with Implicit Geometry

The transfer-based techniques rely on positional correspondences across a small number of images to render new views. In the cases where cameras are only weakly calibrated, as in our proposed methods in chapters 4 and 5, the Euclidean 3D information is not available. New views are usually computed based on this direct manipulation of these correspondences.

Chen and Williams [11] proposed view interpolation, which uses dense optical flow to generate intermediate views directly. Seitz and Dyer [83] extends this method and proposed a *view morphing* which is a specialized version of view interpolation that interpolated views are always geometrically correct.

Laveau and Faugeras [48] use reference views and fundamental matrix to predict new views. Avidan and Shashua [6, 5] employed a trifocal tensor for image transfer. In [6, 5], they assume that camera's intrinsic parameters are known to simplify the specification of virtual camera.

More recently, view interpolation has been applied to images captured from different positions, at different time. Manning and Dyer [61] proposed *dynamic view morphing* which extends view morphing [83] to the rigid objects with translation. Wexler and Shashua [99] proposed another technique to morph a dynamic view with a moving object along a straight line path from three viewpoints. Xiao et al. [101] extended the view morphing to the non-rigid objects with complicated motion.

The biggest challenges of view interpolation are pixel matching and visibility handling. Dense correspondences between the original images are required to generate intermediate views. The correspondences are often generated manually or by optical flow which does not work well when two images are not very similar. Visibility handling becomes more difficult in the cases where the source images are uncalibrated. There is usually no depth information for determining the occlusion in that case.

Lhuillier and Quan [51] proposed *joint view triangulation* (JVT) to handle these problems. Quasi-dense matching and planar patches are constructed first. Then, JVT representation is used to determine visible and half-occluded patches in two images to handle their visibility during the creation of new view.

Our method in chapter 4 uses view interpolation to create new views. 3D model in

projective grid space is reconstructed for making dense correspondences between two reference cameras and also used to determine visibility during new view rendering. So our method can handle both problems that commonly occur in transfer-based rendering.

2.2 Video Based Rendering

Video Based Rendering (VBR) [59] is the extension of IBR approaches to handle dynamic scenes. View synthesis is accomplished in both space and time dimension. The ultimate goal in VBR is to render photorealistic arbitrary views of dynamic, real-world events at interactive frame rates. In the telecommunications industry, conventional television is envisioned to be superseded by interactive television and 3D TV. Instead of sitting passively in front of the television, viewers will have the opportunity to decide themselves from which vantage point to watch a movie or sports event.

This section surveys the recent works on VBR techniques on both off-line and online methods. The online VBRs are the systems that can recover 3D shapes and rendering new views from input videos in real-time, while the off-line VBRs are the ones that cannot. Normally, the time that is needed for 3D reconstruction is much longer than for rendering. Some of the off-line VBRs can provide an interactive frames rate of new view rendering, from the prerecorded videos, by doing 3D reconstruction before hand as a preprocessing step.

2.2.1 Off-line Video-Based Rendering

One of the earliest VBR method is the Virtualized Reality proposed by Kanade et al [41, 42]. In this research, 51 cameras are placed around hemispherical dome called 3D Room to transcribe a scene. 3D structure of a moving human is extracted using multi-baseline stereo (MBS) [72]. Then free viewpoint video is synthesized from the recovered 3D model.

Immersive Video system proposed by Moezzi et al [66] use three to six synchronized cameras to capture different viewpoints of a scene. The static portion of the scene(background) is first manually built. Dynamic objects are extracted as time-varying voxel representations extracted through volume intersection, from which iso-surface objects are created and subsequently rendered. All model construction is done offline.

Goldlücke et al. [22] proposed a system that can render a dynamic scene from novel

viewpoints at 20 frames per second. Their rendering method is based on warping and blending images recorded from multiple synchronized video cameras. The quality of the new view images depends on the accuracy of the disparity maps which are reconstructed off-line and provided together with the images.

Zitnick et al. [108] generated high quality new view images in real-time from 8 cameras. Color segmentation-based stereo algorithm is used to generate photoconsistent correspondences. Mattes for areas near depth discontinuities are automatically extracted to reduce artifacts, then rendering is performed with a layered image representation. Their proposed stereo algorithm, while very effective, is not fast enough for the entire system to operate in real-time.

Carranza et al. [9] recover human motion at off-line process by fitting a human shaped model to multiple view silhouettes. Multi-view texturing is employed during rendering to reproduce the time-dependent changes in the body surface in high detail. The rendering runs at real-time frame rates using conventional graphics hardware.

Starck and Hilton [88] also recover human shape from multiple images using a human model. They use not only silhouette information (which is done by Carranza et al.[9]), but also stereo correspondences and feature cues. The feature cues are manually selected from the image and correspond to projected 3D locations of articulated joints and facial features such as eyes, ears, nose, and mouth. These are used to manually align the model to the images which is a major drawback of the approach.

Moezzi et al. [67] created a free viewpoint video by recovering visual hull of the objects from silhouette images using 17 cameras at the off-line stage. Their approach creates 3D models with fine polygons. Each polygon is separately colored thus requiring no texture-rendering support. Their 3D model can use standard 3D model format such as VRML (Virtual Reality Modeling Language) delivered though the Internet and viewed with VRML browsers.

There are many research focusing on using view synthesis for sports events [45]. *iview* [25, 24] is a British DTI project between BBC, Snell & Wilcox and University of Surrey to develop a free viewpoint system that allows the capture and interactive replay of events such as sport scenes using multiple cameras.

Most of the proposed free viewpoint systems for sports event usually use a prior

about the scene that it consists of static areas (e.g. soccer stadium, tennis court, etc.) and dynamic areas (i.e. players) [34, 46, 31, 30, 32]. The reconstruction or segmentation of static regions in these systems are done manually while the dynamic regions can be done automatically.

Proposed systems in off-line VBR category cannot get a real-time processing for the whole process mainly because they are dealing with a large number of cameras (ranging from tens to hundred) [41, 67], manual preprocessing is needed [88, 34, 31, 30], or they are focusing on the quality of the generated video rather than the processing time [108, 88, 9]. The large amount of data or the time consuming reconstruction algorithms used by these previous methods make it hard to achieve as the online systems.

Our proposed method in chapter 4 is categorized as an off-line VBR because the static background is needed to be segmented manually. We also deal with non-static multiple cameras which need to be calibrated at every frames. Even our method for calibration is done automatically based on features matching, the processing time for calibration, reconstruction and rendering is slower than being implemented as a real-time system.

2.2.2 Online Video-Based Rendering

Only a few VBR methods reach on-line rendering. Complex algorithms used for off-line methods are simply too slow for real-time implementation. Therefore, the generated new view images from online methods might have less accuracy comparing to the off-line ones.

One of the popular online VBR methods is the visual hulls algorithm. This method extracts the silhouette of the main object of the scene on every input image. The 3D shape of this object is then approximated by the intersection of the projected silhouettes. There are some online implementations of the visual hulls algorithm [52, 53, 93]. The main drawback of the visual hulls methods is the impossibility to handle the background of the scene. Hence, only one main object can be rendered. Furthermore, the visual hulls methods usually require several computers, which make

their use more difficult.

Among all these visual hulls methods, the image-based visual hulls presented by Matusik et al.[64] is an online VBR method from uncalibrated cameras. This method reconstruct visual hull of the object using epipolar geometry in an image space instead of 3D space. Thus, it does not suffer from quantization artifacts of voxels like in ordinary visual hull. This method can creates news views in real-time from four cameras. Each camera is controlled by one computer and an additional computer creates the new views.

Another method for on-line rendering is to use a distributed light field as proposed by Yang et al. [104]. They presented a 64-camera device based on a client-server scheme. The cameras are clustered into groups controlled by several computers. These computers are connected to a main server and transfer only the image fragments needed to compute the requested new view. This method provides real-time rendering but requires at least 8 computers for 64 cameras and additional hardware.

Schirmacher et al. [79] presented a system for reconstructing arbitrary views from multiple images with depth using a generalized Lumigraph data structure and a warping-based rendering algorithm. With their technique, it is possible to render arbitrary views of dynamic, non-diffuse scenes at interactive frame rates.

Some plane-sweep implementations achieve online rendering using graphic hardware, graphics processing unit (GPU). The plane-sweep algorithm introduced by Collins[13] was adapted to on-line rendering by Yang et al. [105]. They computed new views in real-time from five cameras using four computers. Geys et al.[21] also used a plane-sweep approach to find out the scene geometry and rendered new views in real-time from three cameras and one computer. Nozick and Saito [70] introduced a plane-sweep implementation for moving camera where all the input cameras are calibrated in real-time using ARToolkit [44] markers.

Our method in chapter 5 belongs to the online VBR group. In the previous works, they usually assume that cameras are strongly calibrated. We present a novel method for online video-based rendering from uncalibrated cameras using plane-sweep algorithm.

2.3 Datasets and Evaluation of Free Viewpoint Video Algorithms

In order to compare and measure the effectiveness of the 3D reconstruction algorithms, common datasets and evaluation methods are necessary. Previous works in image-based reconstruction of static scenes evaluate geometric accuracy using ground-truth 3D shape. One of the possible ways to get the ground truth references is to use the synthetic images generated by computer graphics from a previously known 3D model. Using synthetic images can guarantee that the ground truth references are perfectly accurate. The drawback is that the appearance of the synthetic images is usually not realistic. Images taken from the real cameras include noise, distortion and other artifacts. These artifacts must be properly simulated in the synthetic images to make them not look much different from the real ones.

A more suitable way to get a ground-truth reference of 3D object is to use a laser-scanner [81] or a camera with structures light from projector [78] which provides high accurate depth maps. Acquiring ground-truth like this has an advantage that the input images can be captured from the real scene. Another way is to use a real image captured from another view as a ground-truth reference. The accuracy of 3D reconstruction is then implicitly measured by warping the captured image using the estimated depths or 3D model to the ground truth reference view and do the comparison [77].

There are publicly available datasets with ground-truth references for evaluating the accuracy of image-based reconstruction algorithms. Scharstein et al. [77] described a taxonomy and evaluated two-frame stereo correspondence algorithms. They made several stereo datasets and ground-truth references, acquired by high accuracy stereo depth maps using structured light [78], available on the website [76]. Researchers can use their stereo algorithm to compute disparity maps from these datasets and upload the result for benchmark with other algorithms via this website.

Seitz et al. [81] also gave a quantitative comparison of multi-view stereo reconstruction algorithms using several calibrated multi-view image sets and corresponding ground-truth 3D mesh models. They provided high quality datasets with ground

truth at website [80] for benchmark and evaluating the performance of multi-view stereo reconstruction algorithms. Each dataset is registered with a ground-truth 3D model acquired via a laser scanning process. Similar to [77], researchers can use their multi-view stereo algorithm to reconstruct the 3D model and upload their result to compare the accuracy with other algorithms via this website.

In contrast, research in free-viewpoint video, which focuses on reconstruction and rendering a dynamic scene, currently lacks a consistent framework for quality assessment [89]. To my best knowledge, relatively few works in free viewpoint video have addressed the accuracy or quality of their view synthesis. Most of previous works usually give only a subjective evaluation or only pixel-wise error metrics with respect to the ground truth images [100].

Even though there is no common website for benchmark the quality of free viewpoint video, as for the case of static object reconstruction [77, 81], there are some datasets publicly available for download. Zitnick et al. made dataset that was used in [108] available online at website [107]. There are two sequences and each sequence is 100 frames long. The camera resolution is 1024×768 and the capture rate is 15fps. The depth maps, computed using the stereo technique described in [108], also come with the datasets but only for using as references and should not be regarded as ground truths.

The Computer Graphics Laboratory at Stanford University has acquired several light fields and made this data publicly available to researchers at their website [3]. Liu et al. [55] also provided several datasets on the website for download at [54]. Each dataset consists of images captured by 20 cameras which are mounted on a ring around the studio. The camera resolution is 1024×768 and the capture rate is 25fps.

In Chapter 4, we use our own datasets because the publicly available ones usually assume that cameras are calibrated and static while our cameras are the zooming and rotating ones. Using non-static cameras makes cameras calibration become more difficult. Our contribution is the method to weakly calibrate these cameras. In Chapter 5, our system can render new views from live input videos, so it is easier to capture and render from our own datasets. To make a quantitative evaluation of the results, we use accuracy measurement proposed in [89]. Later works can compare

with our results by computing the same metrics even using the different datasets.

Chapter 3

Projective Grid Space

3.1 Geometry of Multiple Views

The 3D shape reconstruction from multiple view generally requires camera calibration that is used for relating the geometry among cameras. Techniques for calibrating multiple cameras can be categorized roughly into two groups, strong calibration and weak calibration [20].

Strong camera calibration consists of the estimation of the external parameters (position and orientation relatively to a world co-ordinate system), and the internal parameters of the camera (principal point or image centre, focal length and distortion coefficients) [96, 97]. 3D position in Euclidean space of several points and their 2D projection on each view must be measured precisely. For this reason, when there are many cameras, much effort is needed to calibrate every camera. Especially, in the case of large space such as sports stadiums, it is difficult to set many calibration points whose positions are precisely measured throughout the large area.

Weak camera calibration is a process for estimating epipolar geometry [95, 18, 28] from a set of point correspondences between images taken by cameras with unknown intrinsic parameters. These epipolar constraints are represented in fundamental matrix (for two views) or trifocal tensor (for three views). To weakly calibrate cameras, only 2D-2D correspondences among views are necessary.

Projective grid space (PGS) is a weak camera calibration framework proposed by Saito and Kanade [75]. It allows us to define 3D space and to find the projection without knowing the cameras' intrinsic parameters or Euclidean coordinate information of a scene. In the original work [75], fundamental matrices were used to relate every cameras to PGS. We extend the originally proposed PGS by using trifocal tensors for relating non-basis cameras instead of fundamental matrices. This extension increases the stability of calibration and is compatible with more general camera configurations. Our free viewpoint video systems uses the extended framework for calibration. This chapter describes both original proposed scheme (Section 3.3) and our extension (Section 3.4). For related theory about geometry of two-view and three-view, please refer to Appendix A and B

3.2 Definition of Projective Grid Space

Projective Grid Space (PGS) is a 3D space defined by the image coordinates of two arbitrarily selected cameras, called basis camera 1 and basis camera 2. To distinguish this 3D space from the Euclidean one, the coordinate system in PGS are denoted by P-Q-R axis. Figure 3.1 shows the definition of PGS. x and y axes in the image of basis camera 1 correspond to the P and Q axes, while x axis of the basis camera 2 corresponds to the R axis in PGS.

Homogeneous coordinate $\mathbf{X} = (p, q, r, 1)^T$ in PGS is projected on image coordinate $\mathbf{x} = (p, q, 1)$ of the basis camera 1 and $\mathbf{x}' = (r, s, 1)$ of the basis camera 2. $\mathbf{x}'$ must lie on the epipolar line of $\mathbf{x}$, so s coordinate of $\mathbf{x}'$ is determined from ${\mathbf{x}'}^T \mathrm{F} \mathbf{x} = \mathbf{0}$.

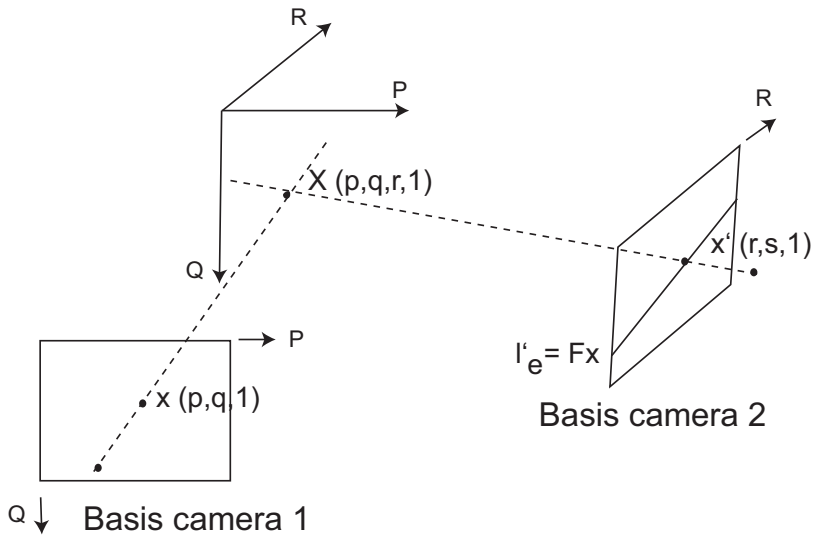


Figure 3.1: Definition of Projective Grid Space.

Other cameras (non-basis cameras) are said to be weakly calibrated once we can find the projection of a 3D point from the same PGS to those cameras. Either fundamental matrices or trifocal tensors between the basis cameras and the non-basis camera can be used for this task. The key idea is that 3D points in PGS will be projected onto the two basis cameras first to make 2D-2D point correspondence. Then, this correspondence is transferred to a non-basis camera by either the intersection of epipolar lines computed from fundamental matrices (Figure 3.2) or point transfer by

trifocal tensor (Figure 3.4).

However, point transfer using fundamental matrices gives less accurate results if a 3D point lies near the trifocal plane (the plane defined by three camera centers). Thus, trifocal tensors are used for weakly calibrating non-basis cameras in our implementation of PGS. For completeness, we will explain about calibrating the non-basis cameras using fundamental matrices as describe in [75] first. Then, we will explain about calibrating non-basis camera using trifocal tensor.

3.3 Weakly calibrating non-basis camera using fundamental matrices

When using fundamental matrices, the fundamental matrix between the basis cameras and a non-basis camera is estimated from at least 7 point correspondences. The projected point in the non-basis camera is computed from the intersection of two epipolar lines from the basis cameras. If the projected point in basis camera 1 and basis camera 2 is $\mathbf{x}$ and $\mathbf{x}'$, respectively, the correspondence in the non-basis camera will be

$$\mathbf{x}'' = (\mathbf{F}_{31}\mathbf{x}) \times (\mathbf{F}_{32}\mathbf{x}'), \tag{3.1}$$

as illustrated in Figure 3.2.

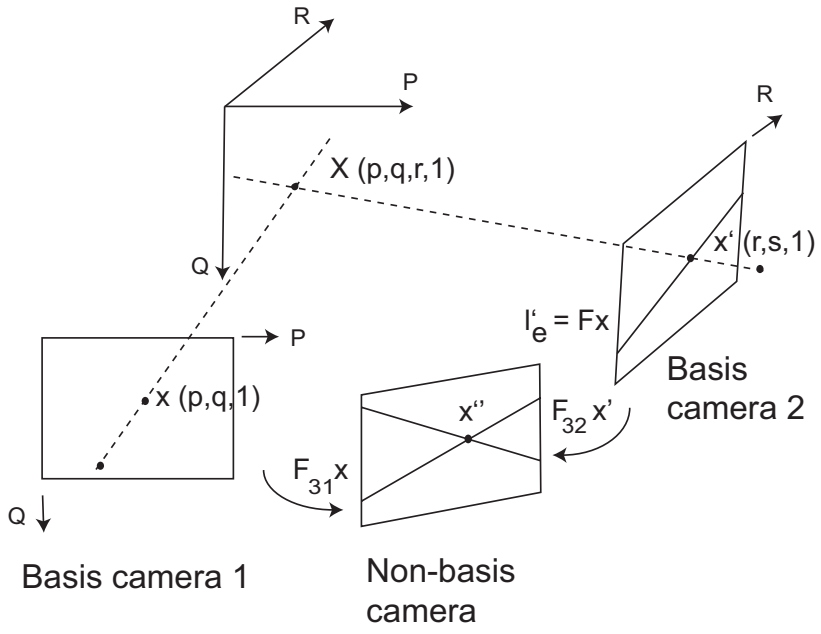
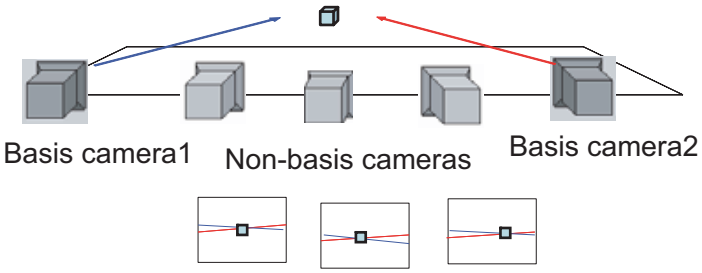


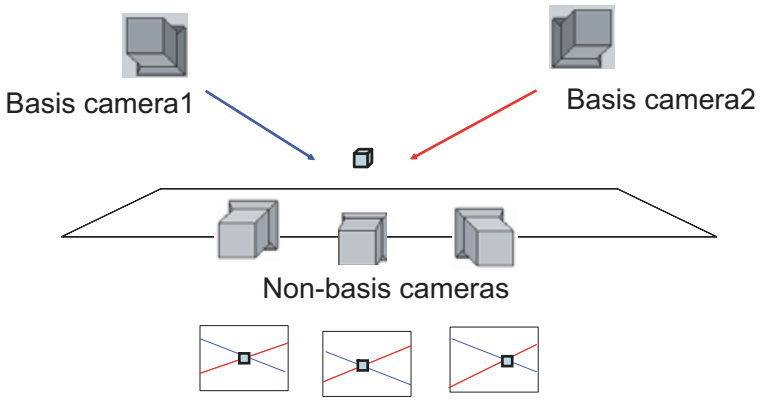
Figure 3.2: Point transfer using fundamental matrices.

However, point transfer using fundamental matrices will fail when two epipolar lines are collinear. This happens when point $\mathbf{X}$ lies on the trifocal plane. Even in the less severe case, the transferred point will also become numerically unstable for the points lying near this plane as illustrated in Figure 3.3(a). This deficiency of point

transfer using fundamental matrices can be avoided by arranging two basis cameras at different heights from the other cameras, like in Figure 3.3(b). By arranging cameras this way, 3D points in the scene will not lie on the trifocal plane, and the intersection of epipolar lines will be well-defined. This approach is also used in [33, 36, 38].



(a) Horizontal camera setting



(b) Non-horizontal camera setting

Figure 3.3: Camera settings. (a) is a bad arrangement of cameras for using epipolar transfer. (b) is a good arrangement of cameras for using epipolar transfer.

3.4 Weakly calibrating non-basis camera using trifocal tensor

Trifocal tensor τ_i^{jk} is a homogeneous 3×3×3 array (27 elements) that satisfies

$$l_i = l'_j l''_k \tau_i^{jk}, \tag{3.2}$$

where l_i,l'_j and l''_k are the corresponding lines in the first, second, and third image, respectively. For more details about tensor notation, please refer to Appendix B.

Trifocal tensors can be estimated from point correspondences or line correspondences between three images. In case of using only point correspondences, at least 7 correspondences are necessary to estimate the trifocal tensor.

Given point correspondence $\mathbf{x}$ and $\mathbf{x}'$, we can find corresponding point $\mathbf{x}''$ in the third camera by Equation (3.3).

$$x''^k = x^i l'_j \tau_i^{jk}, \tag{3.3}$$

where $\mathbf{l}'$ is the line in the second camera that passes though point $\mathbf{x}'$.

We can choose any line $\mathbf{l}'$ that passes point $\mathbf{x}'$, except the epipolar line corresponding to $\mathbf{x}$ because if line $\mathbf{l}'$ is the epipolar line corresponding to $\mathbf{x}$, then $x^i l'_j \tau_i^{jk} = 0^k$ which makes the point $\mathbf{x}''$ undefined. A convenient choice for selecting the line $\mathbf{l}'$ is to choose the line perpendicular to epipolar line of $\mathbf{x}$.

To summarize, given a 3D point $\mathbf{X} = (p, q, r, 1)^T$ in PGS and tensor τ defined by basis camera 1, basis camera 2 and the non-basis camera, we can project point $\mathbf{X}$ to the non-basis camera as follows (see Figure 3.4):

1. Project $\mathbf{X} = (p, q, r, 1)^T$ to $\mathbf{x} = (p, q, 1)^T$ and $\mathbf{x}' = (r, s, 1)^T$ on basis camera 1 and basis camera 2 respectively. s is found by solving $\mathbf{x}'^T \mathrm{F} \mathbf{x} = \mathbf{0}$.

2. Compute epipolar line $\mathbf{l}'_e = (l_1, l_2, l_3)^T$ of $\mathbf{x}$ on basis camera 2 from $\mathbf{l}'_e = \mathrm{F}\mathbf{x}$.

3. Compute the line $\mathbf{l}'$ that passes $\mathbf{x}'$ and perpendicular to $\mathbf{l}'_e$ by $\mathbf{l}' = (l_2, -l_1, -rl_2 + sl_1)^T$.

4. The transferred point in the non-basis camera is $x''^k = x^i l'_j \tau_i^{jk}$.

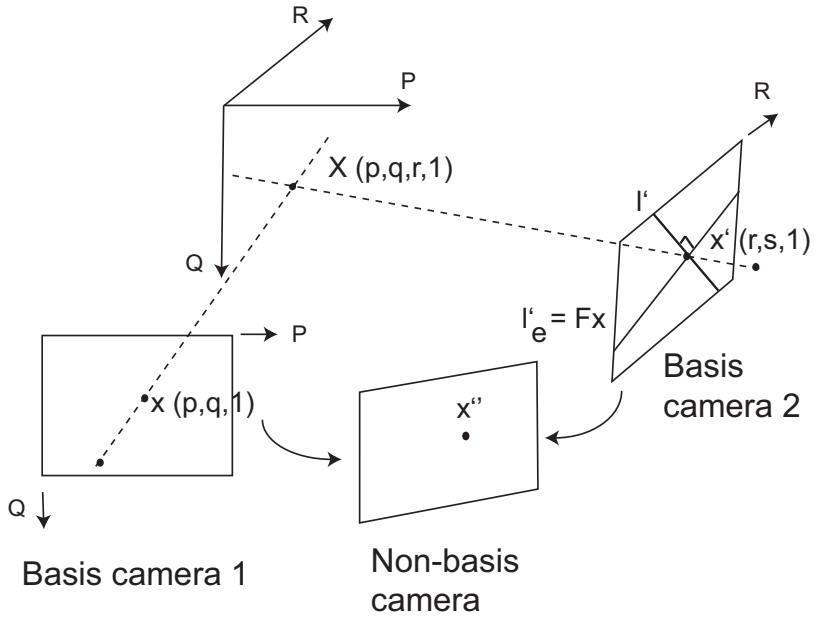


Figure 3.4: Point transfer using the trifocal tensor.

3.5 Camera position in Projective Grid Space

In Figure 3.5, the 3D camera position of basis camera 1 in PGS is ($C1_x$, $C1_y$, $e12_x$), where ($C1_x$, $C1_y$) is the camera center in basis camera 1, and ($e12_x$, $e12_y$) is the epipole of basis camera 1 in basis camera 2. In the same way, the camera position of the basis camera 2 is ($e21_x$, $e21_y$, $C2_x$), where ($e21_x$, $e21_y$) is the epipole of basis camera 2 in basis camera 1, and ($C2_x$, $C2_y$) is the camera center in basis camera 2. For the non-basis camera, 3D camera position in the PGS is ($e1_x$, $e1_y$, $e2_x$) where ($e1_x$, $e1_y$) and ($e2_x$, $e2_y$) are epipoles on basis camera 1 and basis camera 2, respectively.

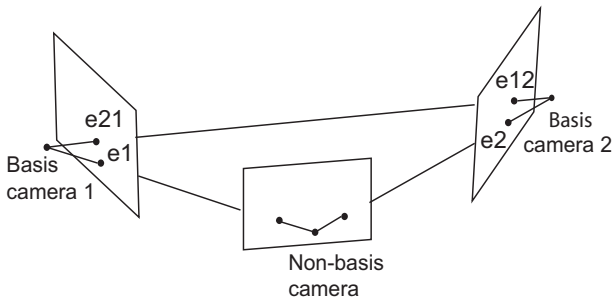


Figure 3.5: Camera position in Projective Grid Space.

Chapter 4

Free Viewpoint Video from Pure Rotating and Zooming Cameras

4.1 Introduction

In most of the free viewpoint video creation from multiple camera systems, cameras are assumed to be fixed. This is guaranteed by mounting the cameras on poles or tripods for the duration of the capturing, and calibration is only done before starting video acquisition. During video acquisition, cameras cannot be moved, zoomed or rotated. The field of view of each camera in these systems must be wide enough to cover the area in which the object moves. If this area is large, the moving object's resolution in the captured video and in the free viewpoint video will decrease.

Allowing cameras to be zoomed and rotated during capture is more flexible in terms of video acquisition. However, in this case, all cameras must be dynamically calibrated at every frame. Doing strong calibration at every frame with multiple cameras is possible by using special markers [44, 70] or some prior about scene, e.g. lines on soccer field [94], etc. In case of using markers, the size of markers must be large enough compared to the scene to make calibration accurate. When the capturing space is large, it is unfeasible to use a huge artificial marker.

In this chapter, we propose a novel off-line video based rendering method for synthesizing free viewpoint video in a natural scene from uncalibrated pure rotating and zooming cameras. Our method does not require special markers or information about intrinsic camera parameters. For obtaining geometrical relation among the cameras, projective grid space (PGS), which is 3D space defined by epipolar geometry between two basis cameras, is used. All other cameras are weakly calibrated to the PGS via trifocal tensors.

We approximate the scene background as several planes. Preprocessing tasks including the selection of 2D-2D correspondences among views and the segmentation of the background are manually done only once at the initial frames (see Figure 4.3). For the other frames, the homographies that relate these frames to the initial frame are automatically estimated. Trifocal tensors of the other frames are then recomputed using these homographies. SIFT [58] is used for finding corresponding points between the initial frame and the other frame for homography estimation. We recover the shape of the moving object in PGS by silhouette volume intersection

[47]. The recovered shape in PGS provides dense correspondences among the multiple cameras, which are used for synthesizing free viewpoint images by view interpolation [11].

4.2 Overview of the proposed method

This section provides the overview of our proposed method. To reconstruct a 3D model without strong camera calibration, we utilize projective grid space (PGS) as already described in chapter 3. Fundamental matrix and trifocal tensors for weakly calibrating cameras, can be estimated from 2D-2D correspondences.

Because our cameras are not static, the fundamental matrices and trifocal tensors must be estimated for all frames. One straightforward way for calibration is finding 2D-2D correspondences among cameras and compute the fundamental matrix and trifocal tensors at every frame. Corresponding points among views can be found by a keypoint detector and descriptor, such as the Scale Invariant Feature Transform (SIFT)[58]. However, robustness of feature point matching of a 3D scene dramatically deceases as the viewpoint between the two images increases [68] because the images of a 3D scene from different views have different appearances due to motion parallax and perspective distortion.

In pure rotating and zooming cameras, all frames from the same camera are related to each other by a homography matrix. If the fundamental matrix and trifocal tensors have already been estimated for one frame, we can compute the fundamental matrix and trifocal tensors of the other frames using the homography matrices relating these frames. This is described in Subsection 4.3.1. Finding correspondences using SIFT for estimating homography is easier and more robust because the capturing position of two images are the same. There is no motion parallax between these images so the two images are more similar. Accurate corresponding points can be found automatically using SIFT and the computational cost does not increase with the complexity of the 3D scene.

From this, we capture the whole background scene without the moving object at the initial frame of each camera. Then two cameras are selected for defining PGS. 2D-2D correspondences between cameras at the initial frame are selected manually (or automatically in case the number of correct correspondences is enough). The fundamental matrix and trifocal tensors of the initial frame are then estimated from these correspondences. To calibrate the other frames to PGS, homography matrices

between that frame and the initial frame are estimated from 2D-2D correspondences automatically found using SIFT. Then, the fundamental matrix and trifocal tensors are re-estimated.

Figure 4.1 shows an example of camera setting in our experiments. We use four DV cameras to capture the scene. Each camera is hand-held without tripod and each person does not move during capture. Because our calibration method is based on finding corresponding points with the initial frame, each camera is rotated and zoomed within the field of view of that frame.

Figure 4.1: The camera setting in our experiment.

Figure 4.2 shows example input frames from each camera. We can see that each camera changes the view direction and focal length from frame to frame. The overall process is illustrated in Figure 4.3 where the detail of each process is explained in the section written in the box. Our main contribution is the calibration part, which is described in Section 4.3. In the rest of the paper, we present the detailed algorithm of each step in Section 4.3-4.5. Finally, the quantitative and qualitative results are shown in Sections 4.6.

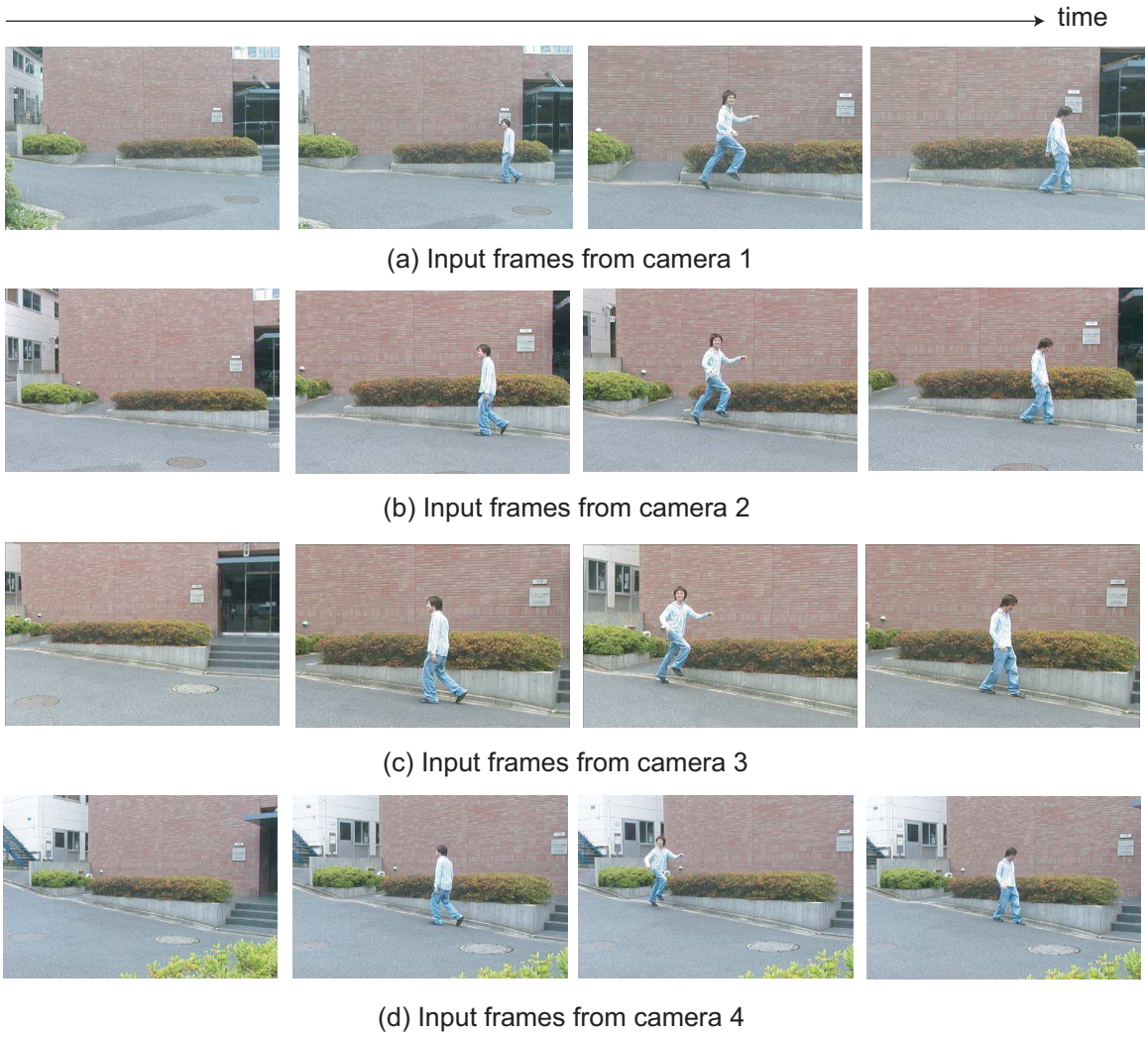


Figure 4.2: Example input frames from four camera.

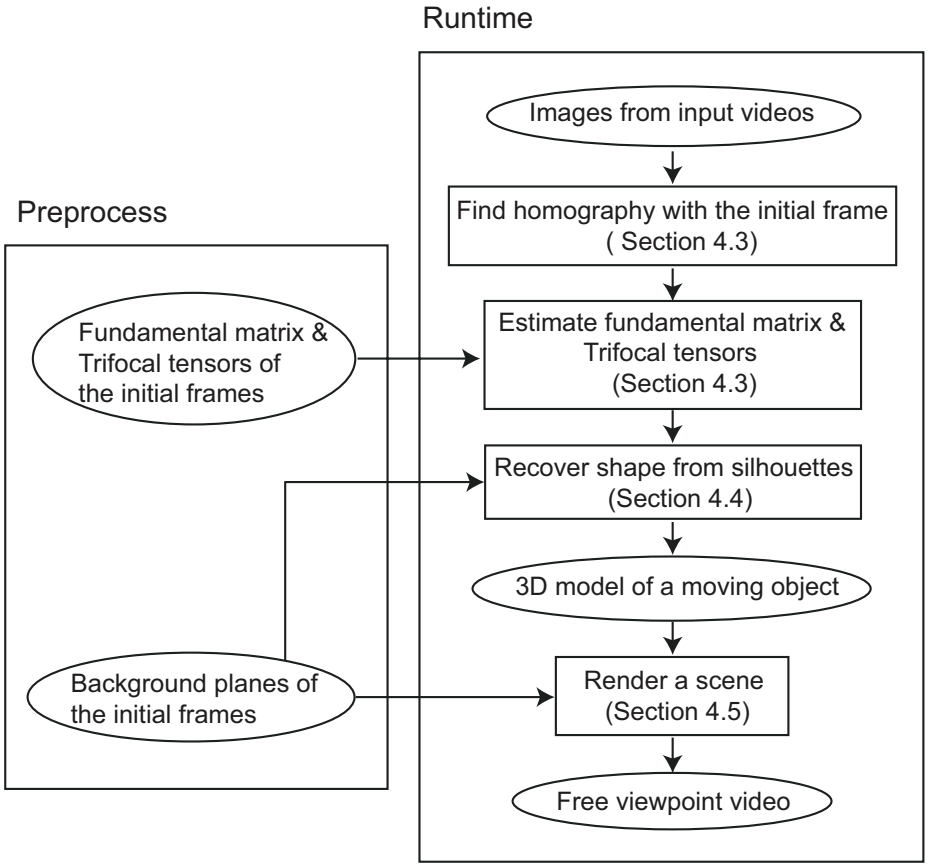


Figure 4.3: Overview of our method.

4.3 Weak camera calibration

To weakly calibrate cameras to PGS, the fundamental matrix between the two basis cameras, and the trifocal tensors between the two basis cameras and the other non-basis camera need to be computed.

For example, in our experiment we use four hand-held camera inputs as shown in Figure 4.2. If we select cameras 1 and 4 to be the basis cameras defining PGS, this means that we need to compute fundamental matrix between cameras 1 and 4 at every frame, and two trifocal tensors defined by cameras 1,4,2 and cameras 1,4,3, respectively.

Our approach for calibration includes two phases: preprocessing and runtime. During the preprocessing phase, we select one initial frame and estimate the fundamental matrix and trifocal tensors manually. During runtime, our method can compute the fundamental matrix and trifocal tensors of the other frames automatically.

To demonstrate the process, we will explain the three camera case. Generalizing to more than three cameras is straightforward by increasing the number of non-basis cameras. Let ψ, ψ', ψ'' represent the initial frames of basis camera 1, basis camera 2, and the non-basis camera, respectively. Let $\hat{\psi}, \hat{\psi}', \hat{\psi}''$ represent the other frames of the same camera.

4.3.1 Preprocessing phase

For the initial frames ψ, ψ', ψ'', we zoom out all cameras to capture the whole area of a scene without an actor. 2D-2D corresponding points for estimating fundamental matrix F between ψ and ψ' and trifocal tensor τ_i^{jk} of ψ, ψ', ψ'' are assigned manually. Once the fundamental matrix and the trifocal tensor are estimated, PGS is completely defined. These images will be used as the reference image for calibrating the other input frames to PGS, as will be described in Subsection 4.3.2. Figure 4.4 shows the initial frames ψ, ψ' and ψ''.

ψ ψ' ψ''

Figure 4.4: Initial frames.

4.3.2 Runtime phase

Let $\hat{F}$ be the fundamental matrix from $\hat{\psi}$ to $\hat{\psi}'$. Let $\hat{\tau_i^{jk}}$ be trifocal tensor of $\hat{\psi}, \hat{\psi}', \hat{\psi}''$. We wish to compute $\hat{F}$ and $\hat{\tau_i^{jk}}$ automatically. The straightforward way is to estimate from corresponding points between $\hat{\psi}, \hat{\psi}'$ and $\hat{\psi}''$. However finding such correspondences is error prone and difficult to achieve robustly in cases where the scene is a 3D scene and the base-line between cameras is large, as shown in [68].

We assume that the person recording the input video will not change position during capture. Thus, we may also assume that each camera is only rotating and zooming. The image coordinate $\mathbf{x}, \mathbf{x}', \mathbf{x}''$ of ψ, ψ' and ψ'' are transformed to the image coordinate $\hat{\mathbf{x}}, \hat{\mathbf{x}}', \hat{\mathbf{x}}''$ of $\hat{\psi}, \hat{\psi}'$ and $\hat{\psi}''$ via homography matrices

$$\hat{\mathbf{x}} = \mathrm{H}\mathbf{x}, \tag{4.1}$$

$$\hat{\mathbf{x}}' = \mathrm{H}'\mathbf{x}', \tag{4.2}$$

$$\hat{\mathbf{x}}'' = \mathrm{H}''\mathbf{x}''. \tag{4.3}$$

Under these point transformations, the fundamental matrix F will transform according to

$$\hat{\mathrm{F}} = \mathrm{H}'^{-\mathrm{T}}\mathrm{F}\mathrm{H}^{-1}, \tag{4.4}$$

while the trifocal tensor τ_r^{st} will transform according to

$$\hat{\tau}_i^{jk} = (H^{-1})_i^r H_s'^j H_t''^k \tau_r^{st}. \tag{4.5}$$

Note that Equation 4.4 is the same equation used in [38] to redefine fundamental matrices of each camera after the initial position. For the detailed proof of Equations 4.4 and 4.5, please refer to [28]. From Equations 4.4 and 4.5, this means that we can estimate the fundamental matrix $\hat{\mathrm{F}}$ and $\hat{\tau}_i^{jk}$ from the homographies between the initial frame given that the initial F and τ_i^{jk} are known.

In our experiments, we use the implementation for trifocal tensor estimation from [63]. To estimate homography matrix, corresponding points between ψ, ψ', ψ'' and $\hat{\psi}, \hat{\psi}', \hat{\psi}''$ are necessary. We employ SIFT for finding such correspondences. Example corresponding points that are automatically found using SIFT are shown in Fig.4.5. In Fig.4.5, the left image is initial frame and the right image is the other frame which will be calibrated to projective grid space. In our experiment, we zoom cameras in and out approximately 1X to 2X. Thus, we extract feature points in two octaves. Other parameters of SIFT features detection are the same as described in [58].

RANSAC (RANdom SAmple Consensus) [19], a general algorithm for fitting data that has errors in measurement, is used to reject outliers in correspondences. The lines in Figure 4.5 show inlier corresponding points that will be used for estimating homography.

Please note that finding correspondences between two images captured from the same position but with a change in focal length and rotation is more robust than finding correspondences between different views. This is because the two images captured from the same position will not have motion parallax. This is the motivation behind our calibration method.

ψ $\hat{\psi}$

ψ'' $\hat{\psi}''$

Figure 4.5: Corresponding points found using SIFT for estimating homography.

4.4 3D Reconstruction

In this section we describe the 3D reconstruction of a human actor. The reconstructed model is used for making dense correspondences between the two original views for image interpolation.

4.4.1 Voxels model

We reconstruct a visual hull of the human actor in projective grid space using the silhouette volume intersection method [47]. To get a silhouette of human actor, we have to generate a virtual background for background subtraction. In the initial frame, we captured a background scene without human actor. In the later frames, a homography matrix, which is estimated for camera calibration as described in Section 4.3, is used for warping the initial frame to the current frame as a virtual background. Then background subtraction can be done as shown in Figure 4.6.

The RGB color $\mathbf{I}$ of a pixel p in the input image is compared to the RGB color $\mathbf{I_{bg}}$ of the same pixel in the warped background image by computing

$$\theta = cos^{-1}(\frac{\mathbf{I} \cdot \mathbf{I}_{bg}}{|\mathbf{I}||\mathbf{I}_{bg}|}), \tag{4.6}$$

$$d = |\mathbf{I} - \mathbf{I}_{bg}|. \tag{4.7}$$

Then, the pixel p is segmented as a foreground pixel if $\theta > \theta_T$ or $d > d_T$ where θ_T and d_T are some thresholds. These thresholds are set based-on the segmentation result of the actual input videos. In our experiment, we use $\theta_T = cos^{-1}(0.999)$ and $d_T = 35$. We then apply morphological operations to reduce the segmentation errors.

In more challenging scenes where a simple pixel-based background subtraction cannot give a satisfying result, more sophisticated algorithms for foreground extraction can be used [90, 37]. After generating silhouettes from all views, each voxel in PGS is projected onto silhouette images to test voxel occupancy. The voxels that are projected outside the silhouettes from one or more views, will be carved out.

The coverage area of all voxels in 3D space and the appropriate size of each voxel

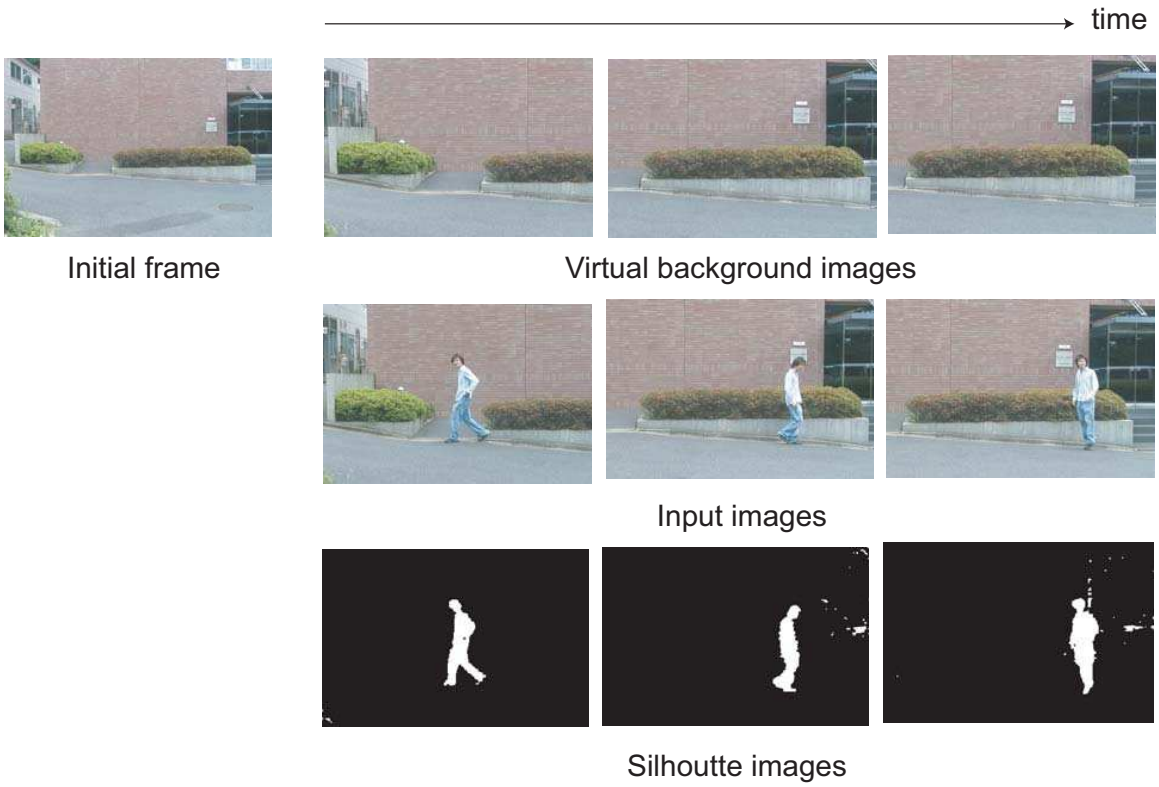
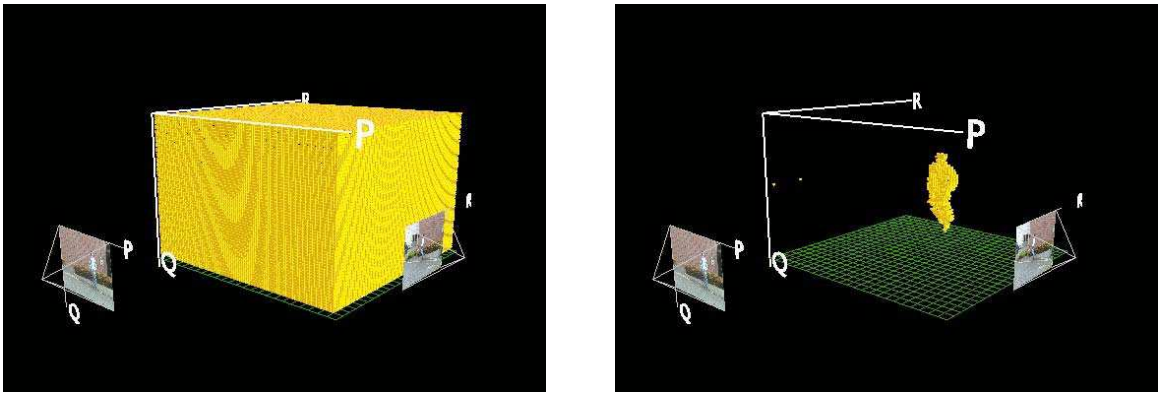


Figure 4.6: Generating silhouette images of a human actor

in PGS can be determined from the input images directly. In our case we use input video that has resolution 720×480, this means that 3D space in PGS which is defined from two basis cameras starts from $(0, 0, 0)$ to $(720, 480, 720)$. We define all voxels for 3D reconstruction to be in this space as shown in Figure 4.7. In our experiment we use $90 \times 60 \times 90 = 486000$ voxels for 3D reconstruction. Figure 4.8 and 4.9 show the example 3D reconstruction results.

4.4.2 Triangular meshes model

The reconstructed 3D voxel model is converted into a polygonal meshe which is more suitable for rendering. We use Marching Cube algorithm [57] for extracting a polygonal mesh from a 3D voxel model. The algorithm proceeds through the voxels, taking eight vertices at a time (thus forming a single temporary cube), then determining



(a) All voxels before 3D reconstruction

(b) Voxels that represent 3D shape after reconstruction

Figure 4.7: Defining voxels in projective grid space

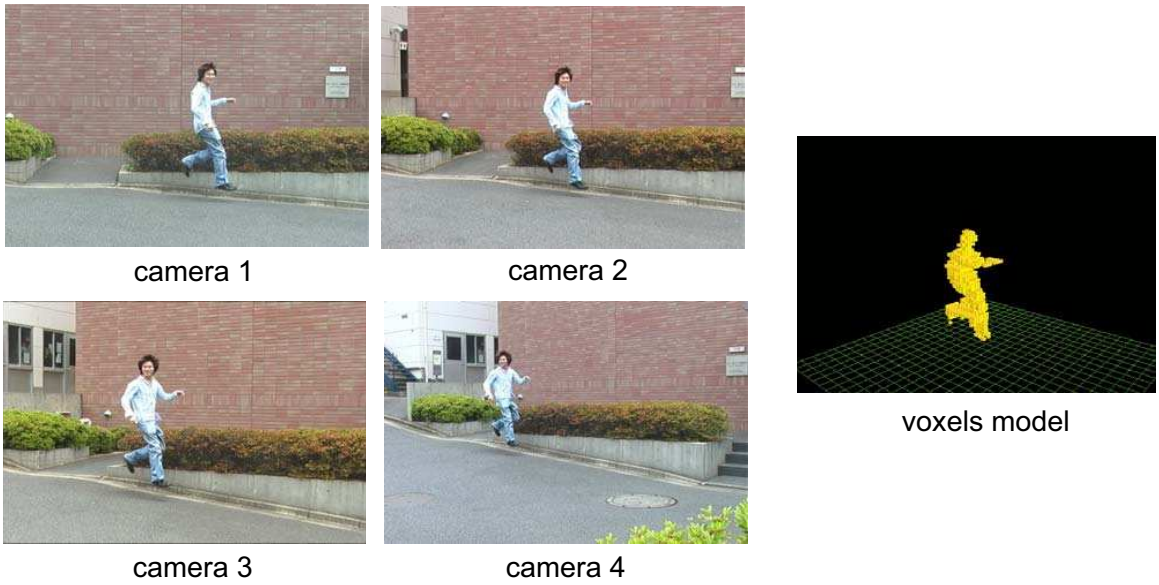


Figure 4.8: Example result of voxels model reconstruction

the polygon(s) needed to represent the part of the isosurface that passes through this cube. The individual polygons are then fused into the desired surface.

The marching cube algorithm is implemented by creating an index to a precalculated array of all 256 possible polygon configurations ($2^8 = 256$) within a single cube. The precalculated array of 256 cube configurations can be obtained by reflections and

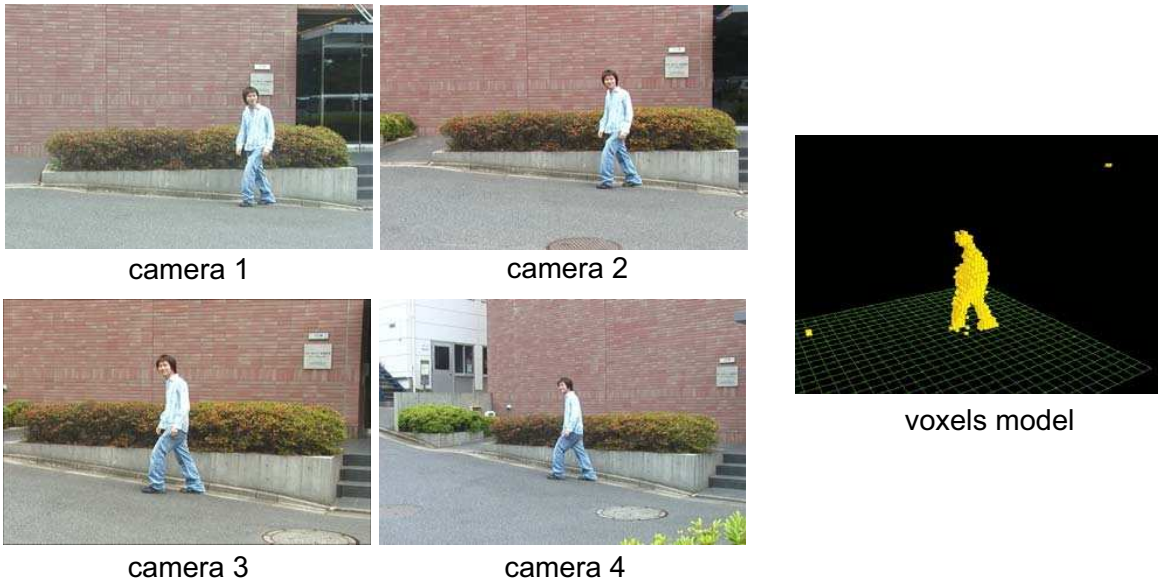


Figure 4.9: Example result of voxels model reconstruction

symmetrical rotations of 15 unique cases as shown in Figure 4.10.

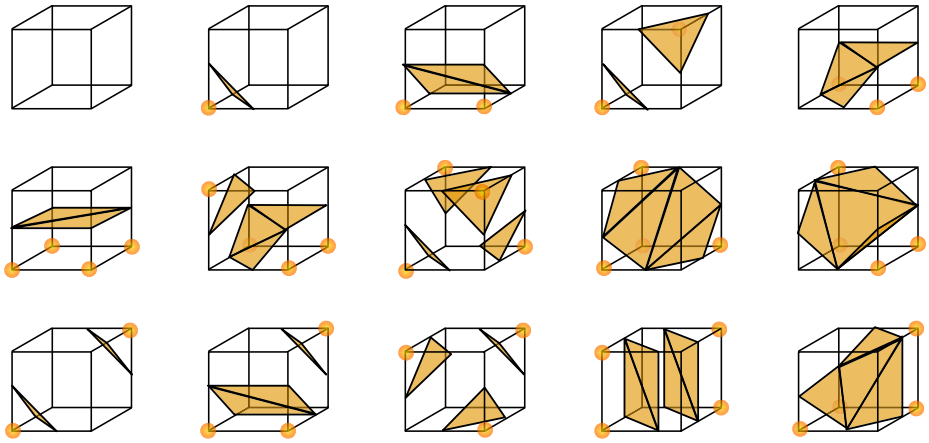


Figure 4.10: 15 unique configurations of marching cubes algorithm.

Figure 4.11 shows an example of extracting a mesh model from voxels model using the Marching Cubes algorithm. This 3D triangular mesh will be used for making dense correspondences for view interpolation as described in the next section.

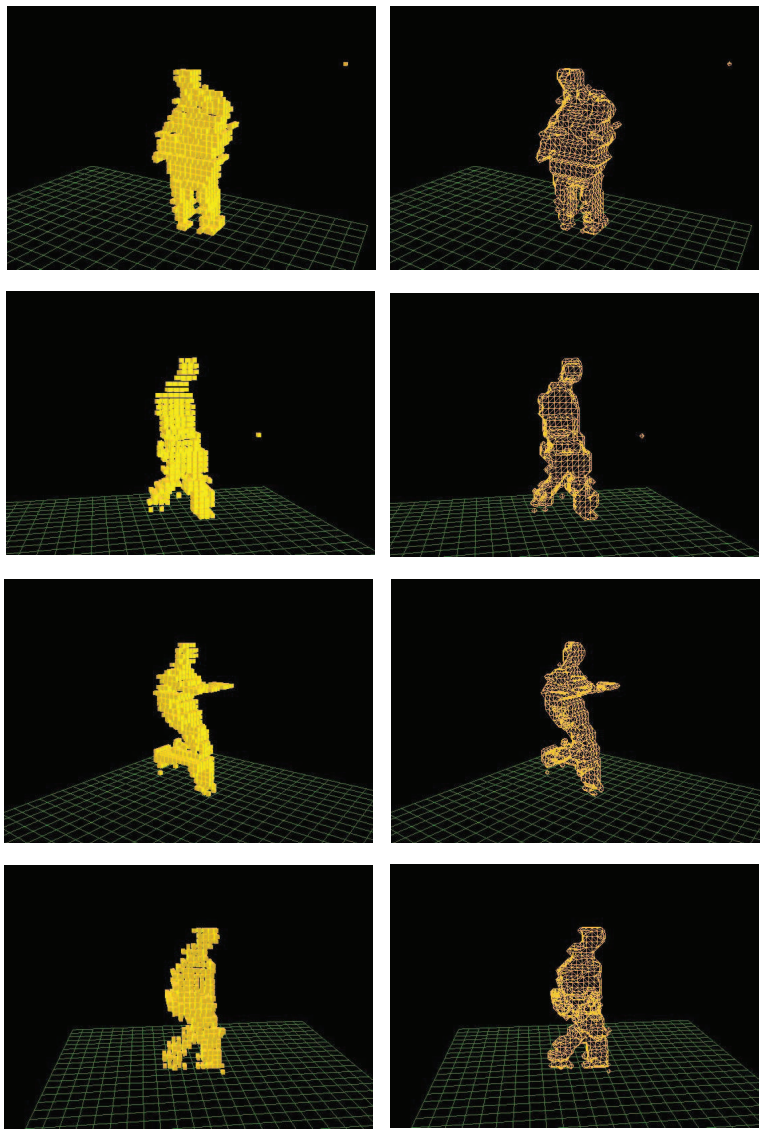


Figure 4.11: Results of extracting meshes from voxels using the Marching Cubes algorithm.

4.5 Free viewpoint video rendering

The biggest challenges of view interpolation are pixel matching and occlusion handling. Dense correspondences between the original images are required to generate intermediate views. The correspondences are often generated manually or by optical flow which does not work well when two images are not very similar. Visibility handling becomes more difficult in the cases where the source images are uncalibrated. There is usually no depth information to determine the occlusion in that case.

Our method can synthesize free viewpoint video using interpolation between the two reference views and can handle both stated problems. The reconstructed model in PGS provide us information about both correspondences and visibility handling.

Free viewpoint video is rendered in three steps. Background planes in a scene are rendered first. Moving object is then rendered and overlaid to the synthesized planes. Finally, holes are filled by the interpolated color from nearby pixels. Figure 4.12 illustrates all these rendering steps. The following subsections explain the details of the two rendering phases.

4.5.1 Background rendering

The scene background is segmented into several planes, as illustrated by Figure 4.13. During preprocessing, the initial frames that we used for calibration are manually segmented. The 3D positions of points that lie on those planes are reconstructed by specifying the corresponding points between basis camera 1 and basis camera 2. If $(p, q)^T$ and $(r, s)^T$ are correspondences in basis camera 1 and basis camera 2, respectively, then the 3D position in PGS of this point will be $(p, q, r)^T$.

These 3D positions in PGS are projected on to both reference views. 2D positions of these points on free viewpoint image are determined using linear interpolation

$$\begin{pmatrix} x \\ y \end{pmatrix} = w \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + (1 - w) \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}, \tag{4.8}$$

where w is a weight, ranging from 0 to 1, defining the distance from the virtual view to the second reference view. $(x_1, y_1)^T$ and $(x_2, y_2)^T$ are the corresponding points on

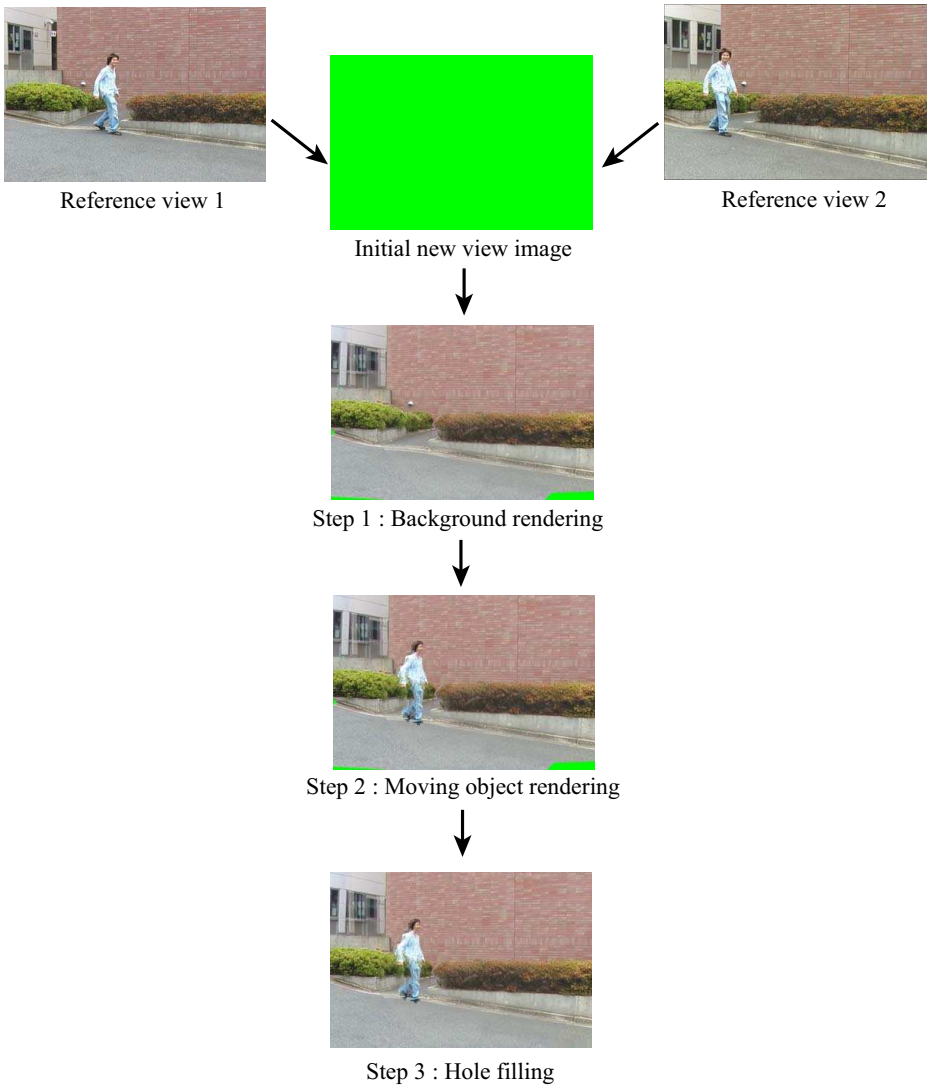


Figure 4.12: New view image after each rendering step.

the first reference view and the second reference view, respectively. Corresponding points between the initial frames of the reference view and the virtual view are used for estimating a homography. The plane in the background image that is segmented during preprocessing is warped to the virtual view. Warped planes from two reference views are then merged together. In case that the scene consists of more than one plane, two or more planes in the virtual view are synthesized in this way and merged together. Fig.4.14 illustrates how the plane is rendered in the free viewpoint image.

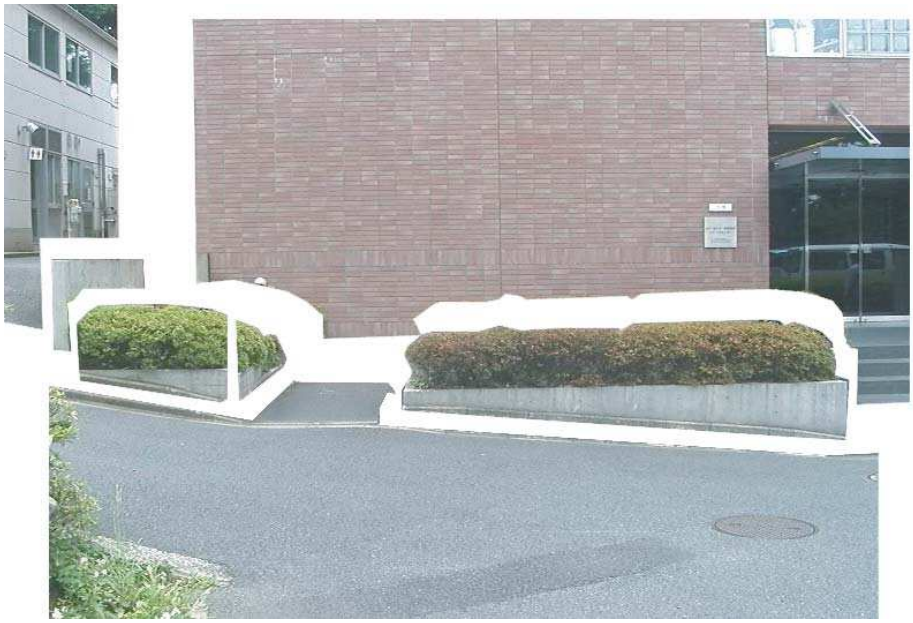


Figure 4.13: Background scene is segmented into several planes

4.5.2 Moving object rendering

Free viewpoint images of a moving object is synthesized by an image-based rendering method. 3D triangular mesh is used for making dense correspondences and also for testing occlusion between the reference images. Each corresponding triangular mesh is warped to the virtual viewpoint image based on the view interpolation method [11].

To test occlusion of triangular patches, the z-buffer of each camera is generated. All triangle patches of a combined 3D model are projected onto the z-buffer of each camera. The values in the z-buffer for each pixel store the 3D distance from the focal point of a camera to the projected triangle patch. If some pixels are projected by more than one patch, the shortest distance is stored. The distance of point $\mathbf{a}(p_1, q_1, r_1)$ and $\mathbf{b}(p_2, q_2, r_2)$ in PGS is defined as

$$D = \sqrt{(p_1 - p_2)^2 + (q_1 - q_2)^2 + (r_1 - r_2)^2}. \tag{4.9}$$

To synthesize a free viewpoint image, each triangle is projected onto the two reference images. Any patch whose distance from the focal point of the input camera

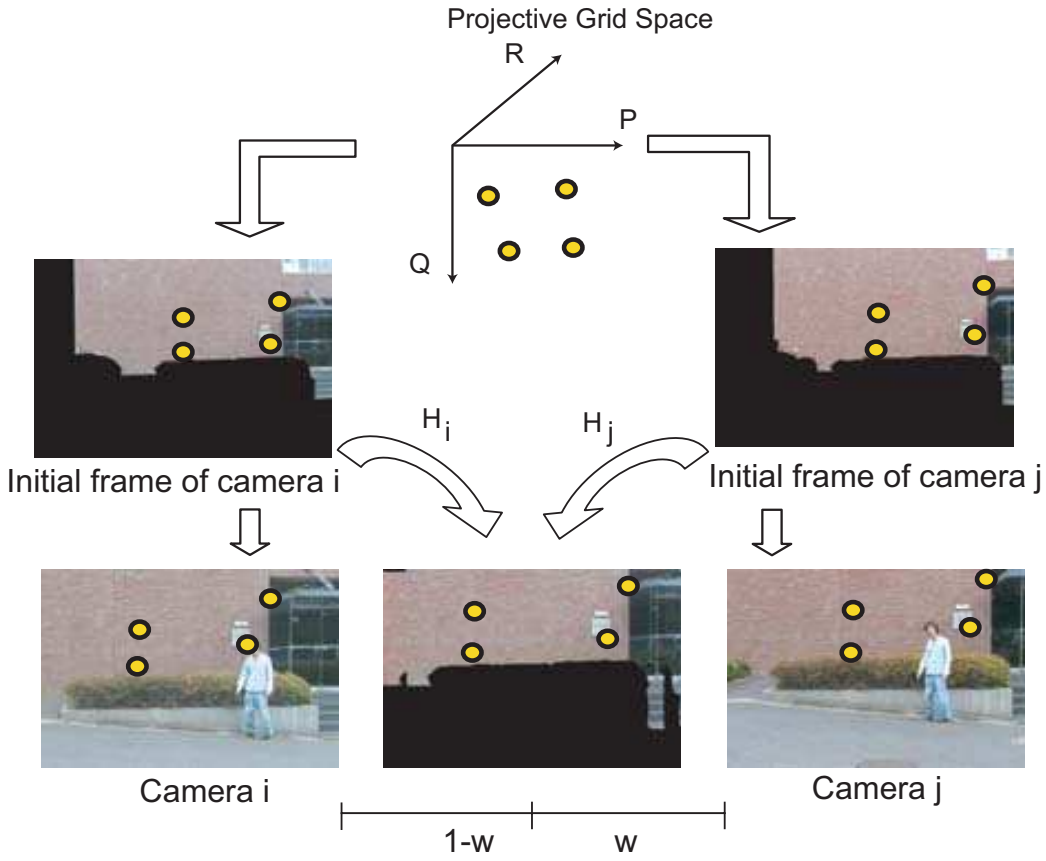


Figure 4.14: Rendering a plane on a free viewpoint image.

is greater than the value stored in the z-buffer is decided to be occluded. In the case that a patch is occluded in both input views, this patch will not be interpolated in a free viewpoint image. If a patch is seen from one or both input views, this patch will be warped and merged into a new viewpoint image. The position of a warped pixel in a new viewpoint image is determined by Equation 4.8.

To merge two warped triangular patches, RGB colors of the pixel are computed by the weighted sum of the colors from both warped patches. If a patch is seen from both input views, the weight used for interpolating RGB color is the same for determining the position of a patch. In case that the patch is occluded in one view, the weight of the occluded view is set to 0 while the weight of the other view is set to 1. Figure 4.15 shows an example of free viewpoint image of a moving object.

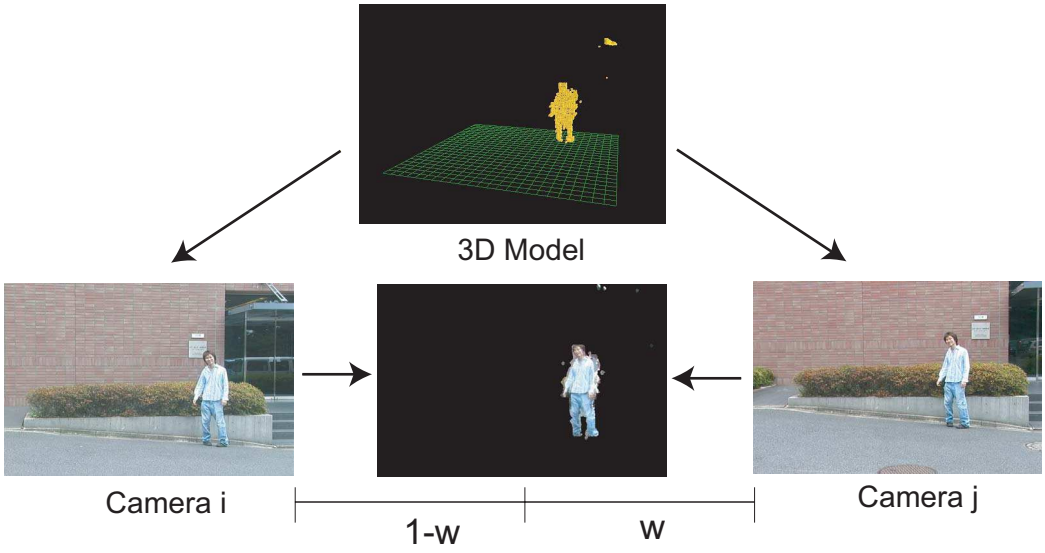
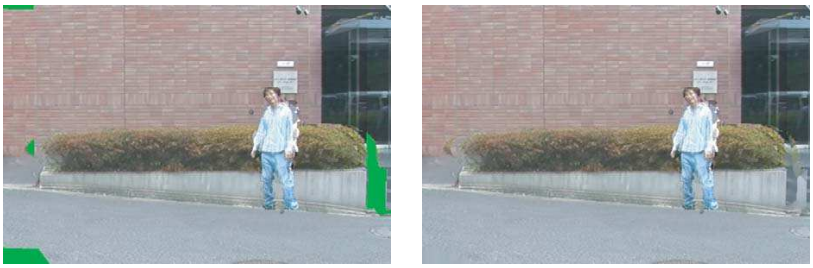


Figure 4.15: Rendering a moving object on a free viewpoint image.

4.5.3 Hole filling

To combine the background with the moving object, the free viewpoint image of the moving object is rendered on top of the background. There might be some holes in the combined image if some areas are not visible in both reference views. These holes are easily noticed and also degrade the quality of the final output video. We use linear interpolation to fill out these holes. The hole filling process finds holes that are adjacent to some color pixels, and then interpolate that hole pixel using the average of the colors of nearby pixels. The process will stop when there are no more holes in the output video. Figure 4.16 show an example image before and after filling holes.



(a) Before filling holes (b) After filling holes

Figure 4.16: Hole filling in the interpolated image. The green color pixels are holes that are not visible in both reference views.

4.6 Experimental results

In this section we show our experimental results by synthesizing free viewpoint video from uncalibrated hand-held cameras using the proposed method. We use four Sony-DV cameras with 720x480 resolution. All cameras are hand-held and captured without tripod as in Figure 4.1. Videos synchronization is done during digitization by Adobe Premiere Pro 2.0 (Adobe Premier Pro is a registered trademark of Adobe, Inc.).

Note that the cameras used in our experiments are almost on the same horizontal line, which is not a suitable camera setting if the fundamental matrices are used for transferring correspondences (see Figure 3.3). Trifocal tensors are used in the implementation of projective grid space. Thus we do not have a problem with this horizontal cameras setting.

4.6.1 Free viewpoint video from consecutive 300 frames

We synthesize free viewpoint video from 300 consecutive frames by our proposed method. During the capturing process, each cameraman stood still, zoomed the camera and changed the view direction within the range of the initial frame independently. We zoom in and out approximately 1X to 2X. The rotation angle of the cameras during capture from the left most view to the right most view is approximately 45 degrees. Example input frames from each camera are shown in Figures 4.17,4.18,4.19 and 4.20. There is no artificial markers placed in the scene. Only natural features are used for finding corresponding points. After the initial frame, our method can correctly calibrate all other frames to PGS and synthesize free viewpoint video without manual operation. Figure 4.21 shows some example frames from the resulting free viewpoint video.

4.6.2 Free viewpoint video from one instance of time

We select one frame from each the input video to create a time freeze effect in which the virtual camera is moved around the scene at one particular time. New views

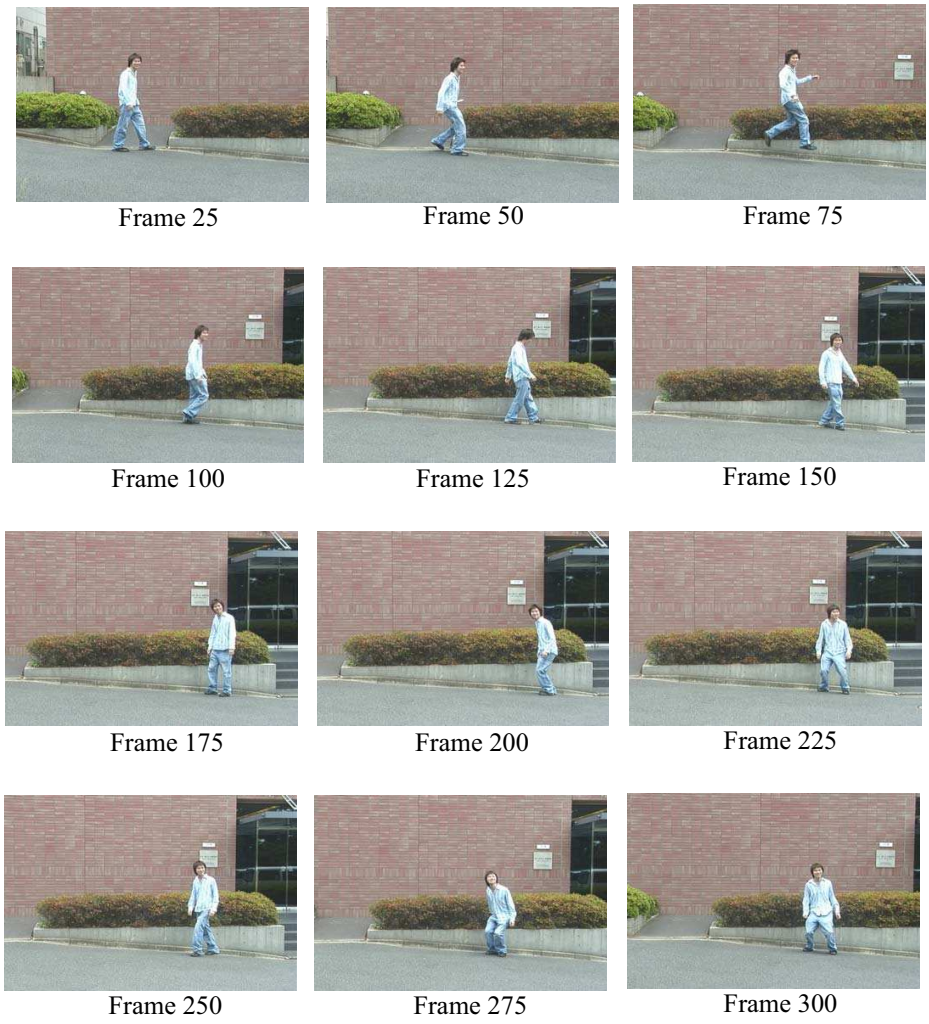


Figure 4.17: Frames captured by camera 1, selected within the 300 frames of the seqeunce.

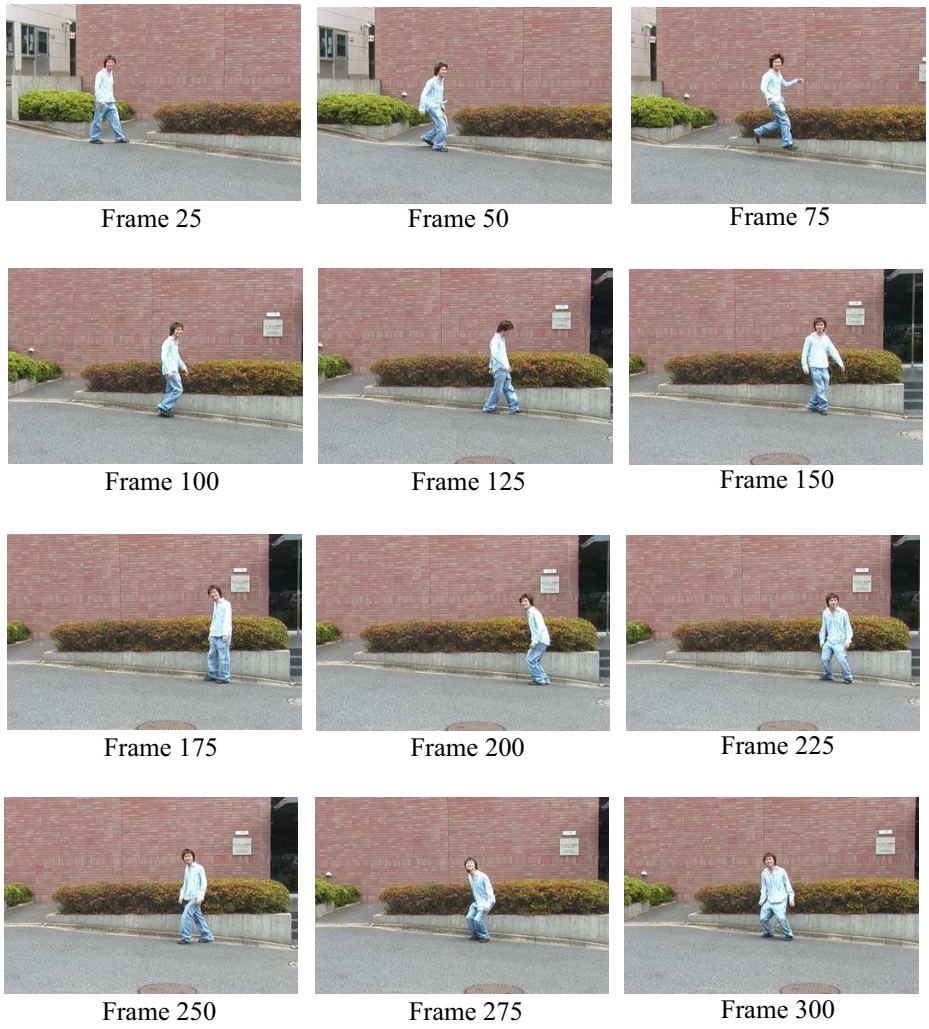


Figure 4.18: Frames captured by camera 2, selected within the 300 frames of the seqeunce.

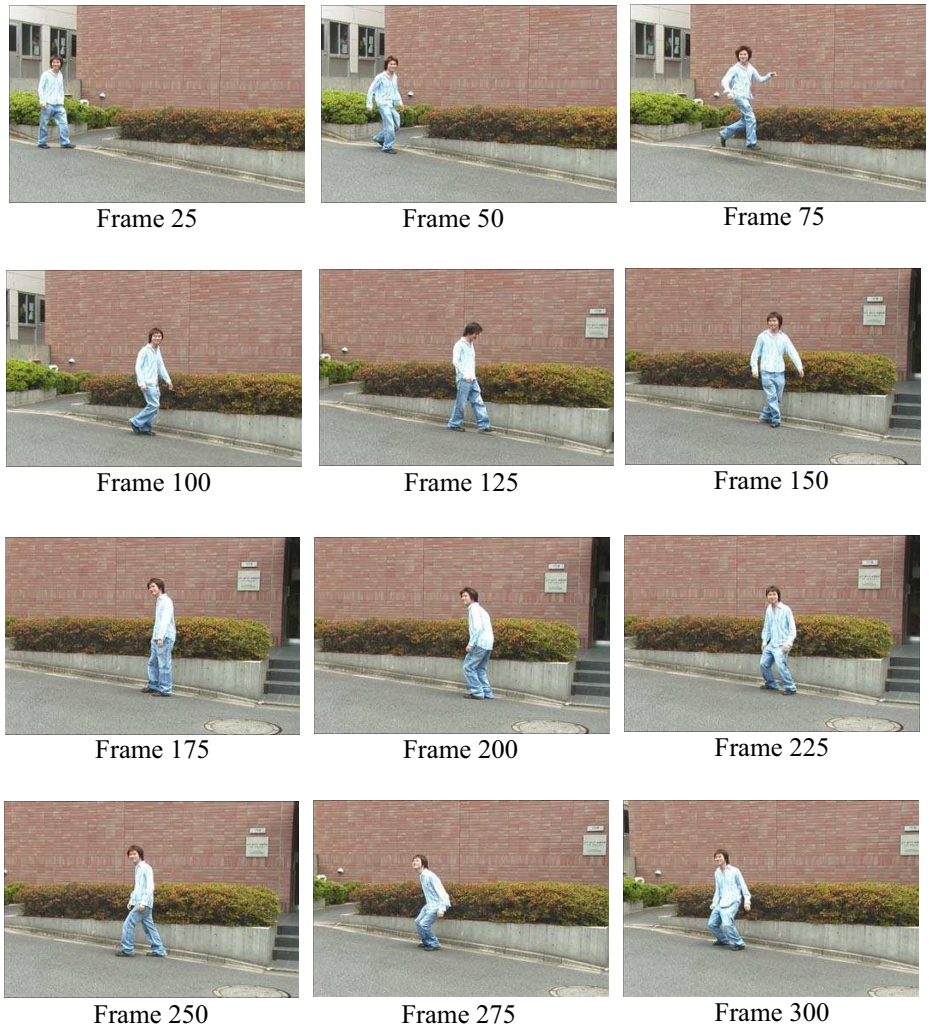


Figure 4.19: Frames captured by camera 3, selected within the 300 frames of the seqeunce.

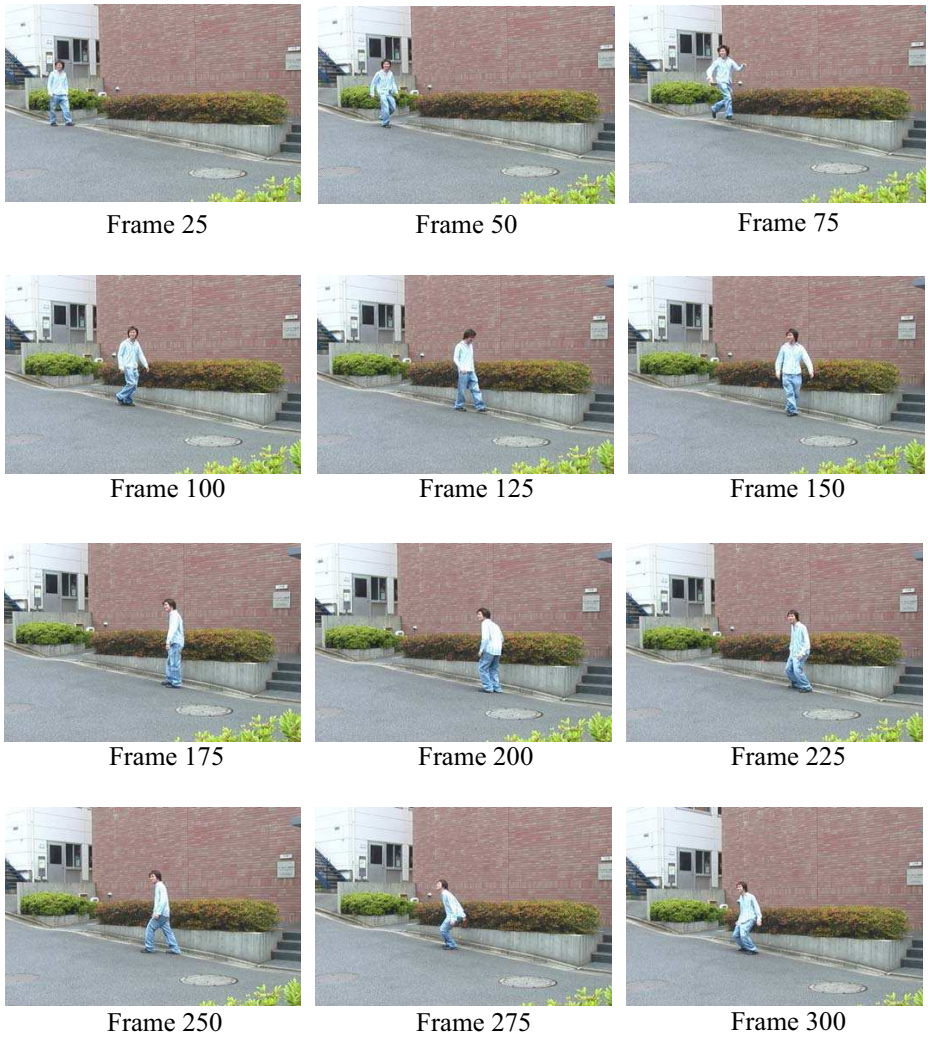


Figure 4.20: Frames captured by camera 4, selected within the 300 frames of the seqeunce.

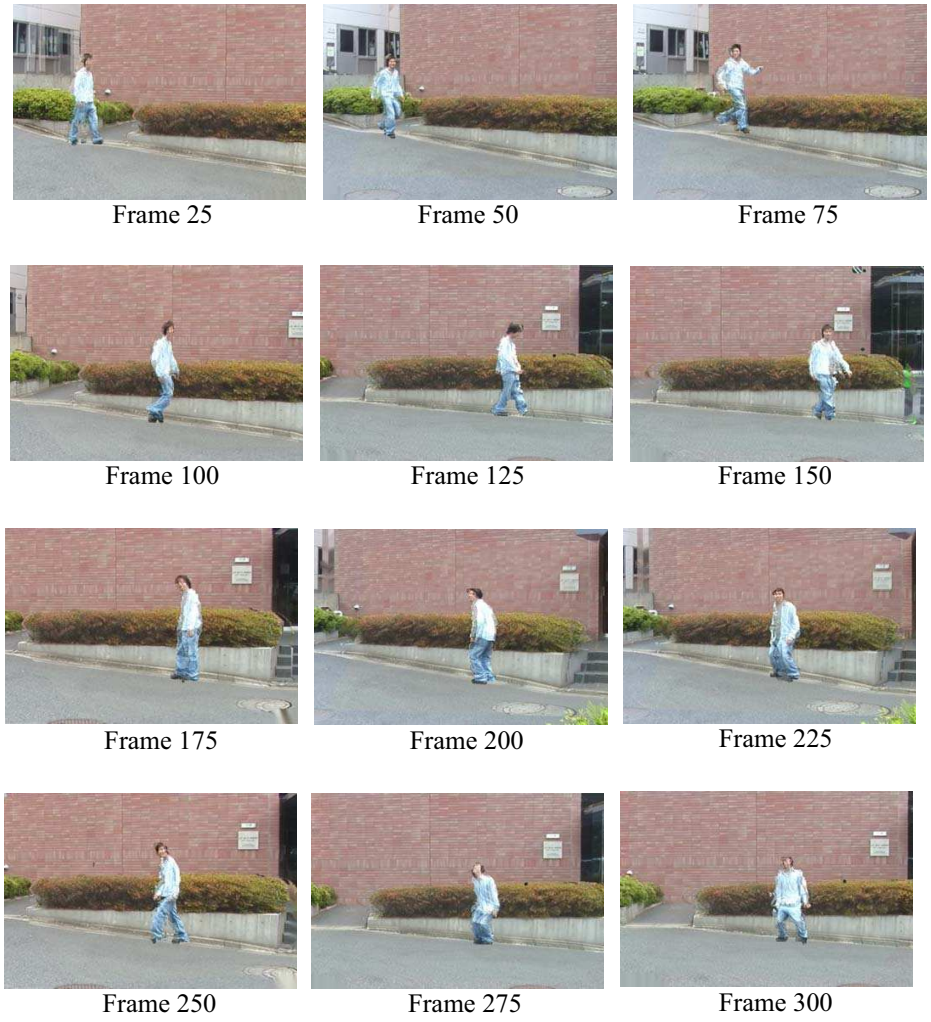


Figure 4.21: Example free viewpoint video from consecutive 300 frames.

between four cameras at several ratios are created as shown in Figure 4.22. The ratio between two views is given under each frame for different virtual views.

4.6.3 Subjective evaluation

From the results, we successfully create new viewpoint images from pure rotation and zoom cameras. Even if there are some artifacts, such as blurred texture or missing part of the moving object, overall quality is acceptable given that only four cameras are used for 3D reconstruction and the base line between cameras is large (approximately 1.5 to 2.0 meters). In this section we give a more detailed analysis of the cause of each artifact and discuss about potential solutions.

Figure 4.23 (a) shows that hole filling does not give a satisfactory result. If a hole appears near a particular object or dense textures, the result seems to be unconvincing. One possible solution is using information from the other views (not reference views) to fill holes.

In Figure 4.23 (b), there are some blurred textures or ghosting (double imaging) on the moving object in the synthesized image because of the inaccuracy of the reconstructed triangular mesh model. If the reconstructed mesh model is different from the real object, the warped textures from both reference cameras will be misaligned in the virtual view.

To reduce blurring or ghosting artifacts, one possible solution is to improve the accuracy of the 3D model. The straightforward way is to increase the number of cameras in the system. The newly added cameras will carve out the non-object voxels during volumetric reconstruction, so the difference between the reconstructed shape and the real one will be reduced. However, the reconstructed visual hull gives only a coarse approximation to the actual shape of the object (concave areas can not be reconstructed). An algorithm for optimizing meshes based on image textures and silhouette can be applied [17, 103].

Because the blurred textures occur when blending intensity of two misaligned textures, another solution is finding a good seam between textures instead of blending. Using this method, blurring or ghosting artifacts could be reduced without optimizing

Figure 4.22: Free viewpoint images from one instance of time.

(a) Hole filling fails

(b) Blurred textures or ghosting

Figure 4.23: Artifacts in the result new view images.

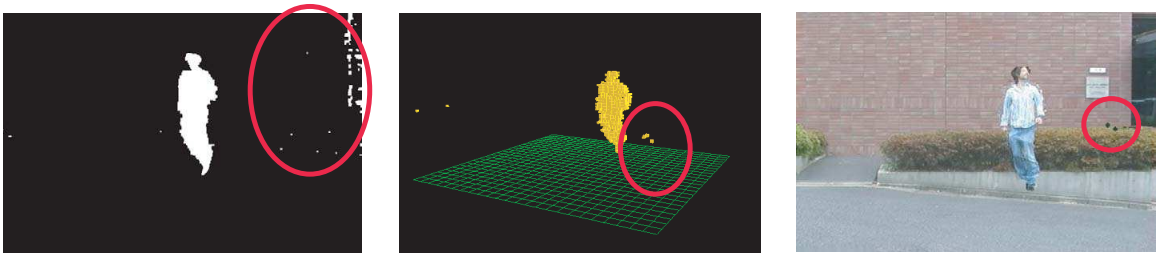
the 3D shape. This approach has been proposed in [49].

Another factor causing blurred textures is the trifocal tensor estimation error. Our method for computation is based on the assumption that the cameras are pure rotating and zooming. However, to show a practical application that this method is not limited to the case where the cameras are purely rotating as placed on a tripod, we use hand-held cameras that are held by cameramen. Each cameraman tries not to move the camera position, but there is still some handshake or other small movement. These contribute to the error during camera calibration.

Imperfect silhouette segmentation cause two kinds of artifacts: missing parts of the moving object and a hole-like region in the new view image. Missing parts of the silhouette images in some views cause missing parts of the moving object in the final free viewpoint image. The background area that is missegmented as the foreground area causes a phantom (no real object) in the reconstructed 3D model. This will appear as a hole-like artifact in the output video, as illustrated in Figure 4.24. The color of this hole-like artifact will depend on the color of the texture in the reference cameras. Because our background is a natural scene, a completely clear silhouette is difficult to achieve using background subtraction.

4.6.4 Objective evaluation

This section gives objective quality measurements of our result. We use no-reference (no ground truth) evaluation method proposed in [89] to measure the error in registering scene appearance in image-based rendering. Two new viewpoint images at the



(a) Imperfect silhouette (b) Phantom in a 3D model (c) Hole-like artifacts

Figure 4.24: The background area that is missegmented as the foreground causes a phantom in the 3D model and cause a hole-like artifact in the new view image.

center (ratio 50:50) between the two reference cameras are rendered. Each new view image is rendered using the texture from only the corresponding reference camera, as shown in Figure 4.25.

(a) Reference camera 1 (b) New view image using texture from camera 1 (c) New view image using texture from camera 2 (d) Reference camera 2

Figure 4.25: New view image rendered for evaluating appearance registration errors.

Two metrics d^{90} [89] and PSNR (Peak Signal to Noise Ratio) are computed over the overlapping pixels to measure the registration error in these new view images (reprojected appearances). d^{90} tells us about the overall distance of misaligned pixels between two images. The lower the value of d^{90}, the better the quality of the output in new view images. If the rendered image from one reference camera is much different from the other, then there will be visual artifacts, like blurred texture or ghosting in the blended image. We measure these values for 100 consecutive input frames between every adjacent cameras. Figures 4.26 and 4.27 show each error metric of our new view images. Table 4.1 shows the average d^{90} and $PSNR$ values over 100 frames.

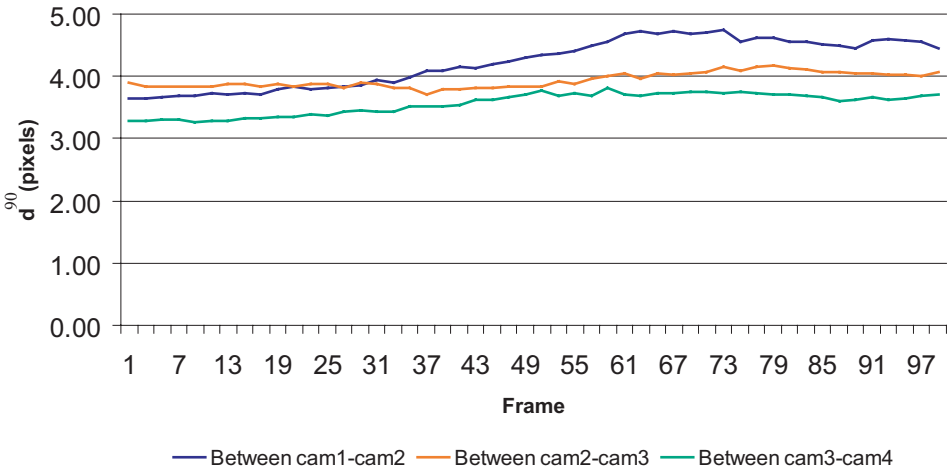


Figure 4.26: d^{90} registration error of new view images.

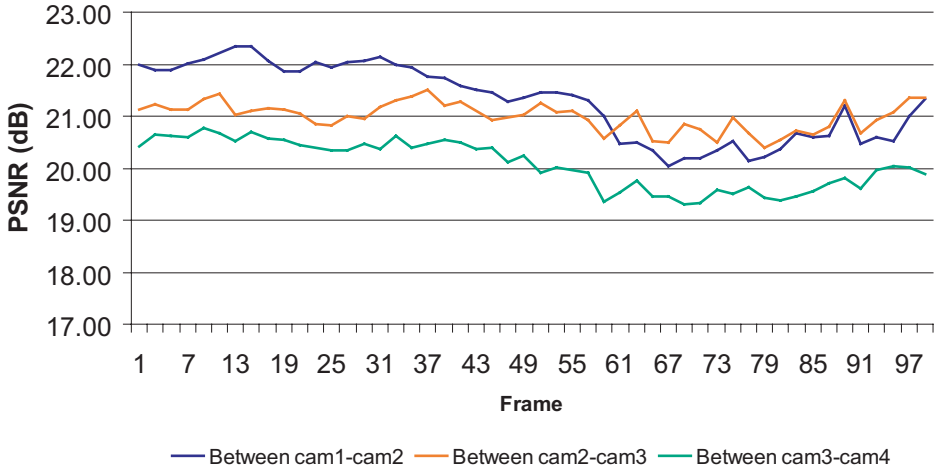


Figure 4.27: PSNR registration error of new view images.

Virtual camera between	$d^{90}(pixels)$	$PSNR(dB)$
cam1 – cam2	4.23	21.29
cam2 – cam3	3.94	20.99
cam3 – cam4	3.56	20.07

Table 4.1: Error measurements for the resulting new view images (average of 100 frames).

Chapter 5

Realtime Free Viewpoint Video using Plane-Sweep Algorithm

5.1 Introduction

In this section, we present a new online VBR method that creates new views of the scene from uncalibrated cameras from live input videos. The contribution of this work concerns the use of uncalibrated cameras in online VBR. Most of previous methods usually assume that camera are strongly calibrated which means that intrinsic parameters of cameras are estimated. Our method does not require information about intrinsic parameters. For obtaining geometrical relation among the cameras, we use Projective Grid Space (PGS)[75] which is a 3D space defined by epipolar geometry between two basis cameras. All other cameras can be related to the same 3D space by trifocal tensors between two basis cameras. Near and far planes in PGS for doing plane-sweep are easily defined and visualized from basis camera 2. In a standard plane-sweep for the Euclidean space, the near and far plane are selected from the real 3D space which is not so convenient. We simultaneously reconstruct and render novel view using plane-sweep algorithm. To achieve real-time performance, we implemented our plane-sweep method in a graphics processing unit (GPU).

In the following sections, we first explain the general concept of the plane-sweep algorithm and introduce our plane-sweep algorithm in PGS. Implementation detail of plane-sweep in GPU for real-time rendering is then described. Finally, we present our experimental results using the proposed method.

5.2 Plane-Sweep in The Euclidean Space

The plane-sweep algorithm creates novel views of a scene from several input images. This section first explains the conventional plane-sweep algorithm in the Euclidean space of the calibrated cameras. Then we shows the modification for using it in Projective Grid Space in the section 5.3.

Considering a scene where the objects are exclusively Lambertian surfaces, the viewer should place the virtual camera cam_x somewhere around the real video cameras and define a *near* plane and a *far* plane such that every object of the scene lies between these two planes. Then, the space between *near* and *far* planes is divided into several parallel planes π_k as depicted in figure 5.1.

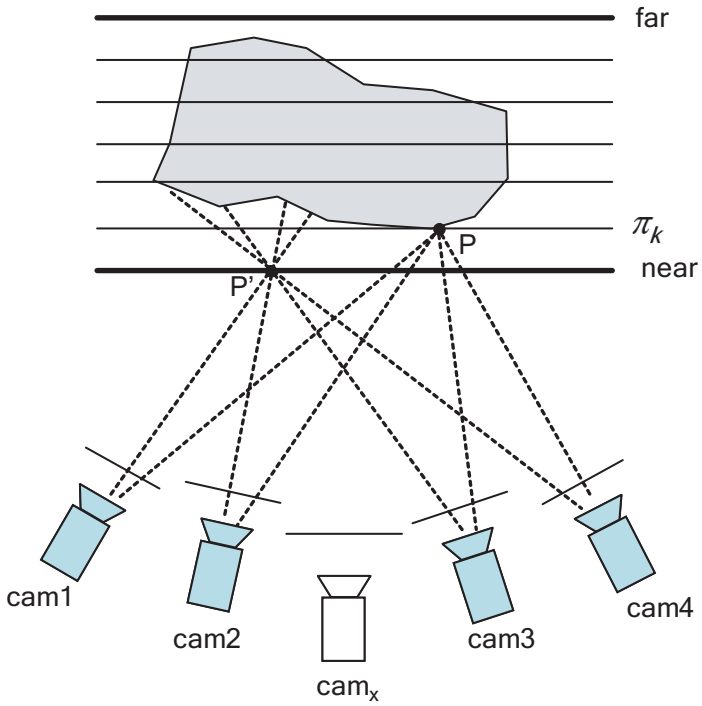


Figure 5.1: Plane-sweep algorithm in the Euclidean space.

Plane-sweep algorithm is based on the following assumption: a point lying on a plane π_k whose projection on every input camera provides a similar color will potentially correspond to the surface of an object. Considering a visible object of the scene lying on one of these plane π_k at a point P, this point will be seen by every

camera with the same color, i.e., the object color. Now consider another point P' lying on a plane but not on the surface of the visible object, this point will probably not be seen by the capturing cameras with the same color. Figure 5.1 illustrates this principal idea of the plane-sweep algorithm.

During the new view creation process, every plane π_k is computed in a back to front order. Each point P of a plane π_k is projected onto the input images. A score and a representative color are computed according to the matching colors. A good score means every cameras see a similar color. The computed scores and colors are projected onto the virtual camera cam_x. The pixel color in the virtual view will be updated only if the projected point p provides a better score than the current one. Then the next plane π_{k+1} is computed. The final new view image is obtained once every plane has been computed. This method is detailed in [71].

5.3 Plane-Sweep in Projective Grid Space

To achieve plane-sweep in PGS, we need to define the position of the virtual camera, to define planes in PGS, and then compute a new view from the defined planes. In this section we describe each step in detail.

5.3.1 Defining Virtual Camera Position

To perform the plane-sweep algorithm, the 3D points on a plane must be projected to a virtual camera. In the calibrated cameras case, the projection matrix of the virtual camera can be defined from its pose, because intrinsic camera parameters are known. This makes possible to move the virtual camera anywhere around a scene.

In our case where PGS is used, the intrinsic parameters of used cameras are unknown. The method for defining virtual camera in the calibrated case is not applicable to our case. In our method, the position of the virtual camera is limited to move only between two real reference cameras. A ratio r from 0 to 1 is used for defining the distance between these reference cameras. Figure 5.2 illustrate this definition. In figure 5.2, a ratio r equals to 0 (respectively 1) means the virtual camera has the same position as camera 1 (respectively camera 2).

To find the projection of a 3D point $\mathbf{X}$ in PGS on the virtual camera, $\mathbf{X}$ is projected onto both real reference cameras first. The projection on the virtual camera is calculated using linear interpolation. If the projected points in the reference cameras 1 and 2 are $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively, the projected point $\mathbf{x}_3$ in a virtual camera is calculated from

$$\mathbf{x}_3 = (1 - r)\mathbf{x}_1 + r\mathbf{x}_2, \tag{5.1}$$

as in figure 5.2.

5.3.2 Defining Planes in PGS

Any arbitrary *near* and *far* plane in PGS can be defined for doing plane-sweep. In our method we define the planes along the R axis (x image coordinate of basis camera 2) as shown in figure 5.3. This approach makes easy to adjust the 3D *near* and *far*

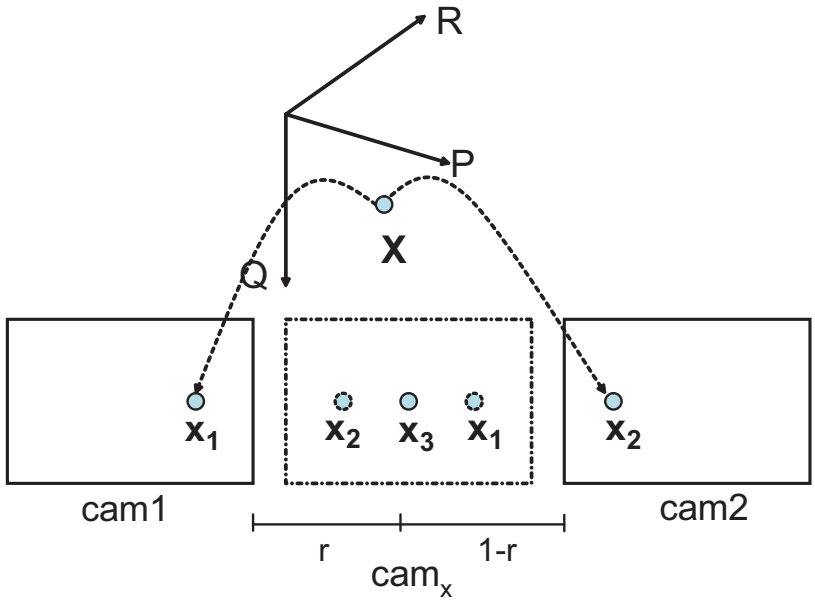


Figure 5.2: Defining a virtual camera in Projective Grid Space.

planes since we can visualize them directly from the image of basis camera 2. This is impossible for the case of the normal plane-sweep algorithm in the Euclidean space in which full calibration is used. In that case, actual depth of a scene has to be measured so that *near* and *far* planes cover all volume of interest.

In our approach, basis camera 2 will not be used for color consistency testing during perform plane-sweep because every planes would be projected as a line in this image. So the basis camera 2 is needed only for weakly calibrated cameras to PGS, after that we can disable it to save CPU time.

5.3.3 Computing New Views

In this section, we explain how we implemented the plane-sweep algorithm after defining the virtual camera's position and planes in PGS. If pixel p in a virtual camera is back projected on a plane π_k to a point P, we want to find the projection of P on every input image for the score computation step. As illustrated on 5.4, the projection of the 3D point P on the input image i can be performed by a homography

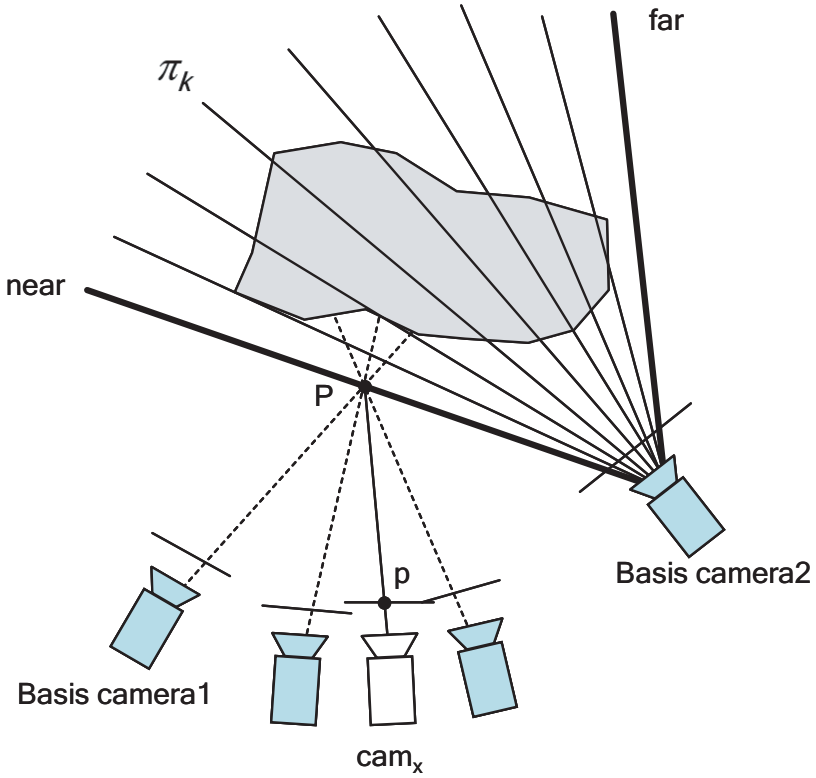


Figure 5.3: Defining planes for doing plane-sweep in Projective Grid Space.

$\mathbf{H}_i$. Thus, the projection p_i of a 3D point P on the camera i is calculated from

$$\mathbf{x}_i = \mathbf{H}_i \mathbf{H}_x^{-1} \mathbf{x}, \tag{5.2}$$

where $\mathbf{x}$ and $\mathbf{x}_i$ are the position of the pixel p and p_i respectively.

Homography H_i, where i is a camera number, can be estimated from at least four point correspondences. In our situation, we select four points defined as the image corners of the basis camera 1 as shown in figure 5.4. Then, we project these points onto every real cameras as described in section 3.2 for making 2D-2D point correspondences. Then, all homographies used for the plane-sweep method can be estimated from these correspondences. During the score computation, we estimate these homographies instead of projecting every 3D points one by one for computation time purpose.

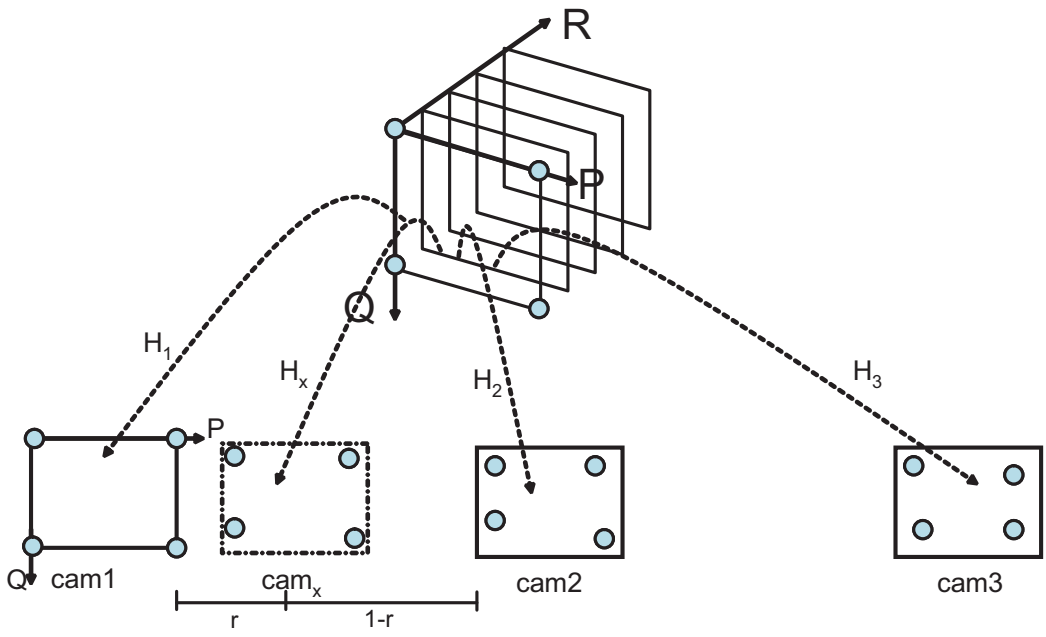


Figure 5.4: Estimating homography matrices for plane-sweep

Algorithm 1 summarizes our plane-sweep algorithm in PGS. In algorithm 1, we use the score function proposed in [71].

Reset color consistency score of the virtual camera to the max value.

foreach *plane* π_k *in PGS* **do**

foreach *pixel* p *in* cam_x **do**

- project pixel p to n input images excluding basis camera 2 . c_j is the color from this projection on the j-th camera
- compute average color : $color_p = \frac{1}{n} \sum_{j=1..n} c_j$
- compute color consistency score from variance: $score_p = \sum_{j=1..n} (c_j - color_p)^2$

if $score_p$ *is lower than current score of pixel* p **then**
update score and color on virtual camera to $score_p$ and $color_p$.

end

end

end

Algorithm 1: Plane-sweep algorithm in Projective Grid Space.

5.4 Implementing Real-Time Plane-Sweep on GPU

To achieve real-time computation, we implement our plane-sweep algorithm in Projective Grid Space on GPU. Because GPU has a massive parallel processing, using GPU can gives much more computation power in many application comparing to CPU. This section gives some details about our implementation. We use OpenGL for the rendering part. Input images that will be used for color consistency checking are transfer to GPU as multi-textures. In drawing function we loop though each plane in PGS from near to far plane. Homographies for warping points on virtual camera to the other cameras are sent to GPU as texture matrices.

We use Orthographic projection and draw square to cover the whole image of virtual camera. In fragment shader we apply the homography to compute the color consistency score as described in algorithm 1. Fragment color is assigned to be an average color from all views. The score of fragment is sent to the next rendering pipeline(frame buffer operation) via gl_FragDepth while the average color is sent via gl_FragColor. Then we let OpenGL select the best scores with the z-test and update the color in the frame buffer. When rendering is done for all planes, we obtain the novel view in the frame buffer.

The maximum number of cameras for our implementation is limited by the number of texture unit on a GPU (usually 8). This is enough for normal cases because when using more than 8 cameras, the bandwidth for image acquisition would already be saturated.

5.5 Experimental Results

We tested our proposed method on PC Intel(R) Core(TM) 2 Duo 2.00 GHz CPU with graphic card NVIDIA GeForce 8600M GT. Five Logitech fusion webcams with a resolution 320x240 are used to capture input videos. The camera setting is depicted by figure 5.5. We select two cameras for defining projective grid space, as illustrated by figure 5.5

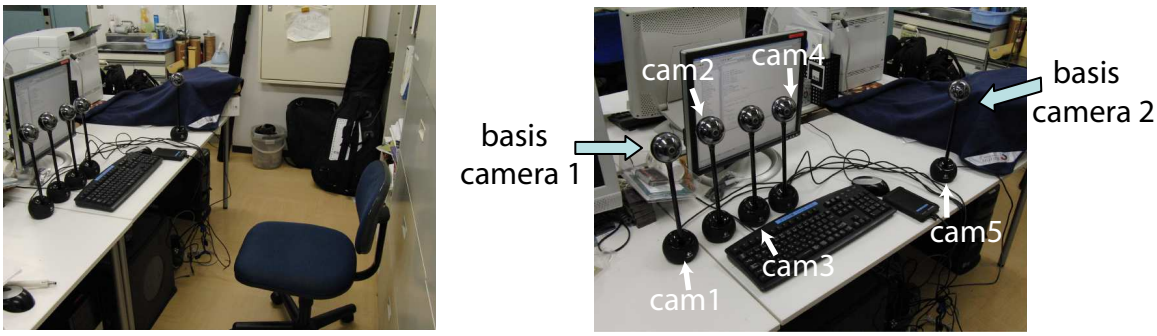


Figure 5.5: Camera configuration.

The fundamental matrix between camera 1 and 5, the three trifocal tensors defined by camera 1,5,2, camera 1,5,3, and camera 1,5,4 are estimated for weakly calibrating cameras to PGS. 2D-2D correspondences for estimating fundamental matrix and trifocal tensors can be selected from feature points in a scene. However, in our implementation, we wave a marker around and track features for 2D-2D correspondence automatically. Thus, we do not have a problem of calibrating even in the scene with only a few natural features. We use the code for estimating trifocal tensors from [63].

Figure 5.6 shows the user interface of this system. Input can be either from the prerecord videos or from the live cameras. The transparent yellow rectangle in the main window represents the position of the virtual camera. Planes for doing plane-sweep is adjusted and visualized from this user interface. Viewer can change the viewpoint easily by dragging mouse to the left or right of current position.

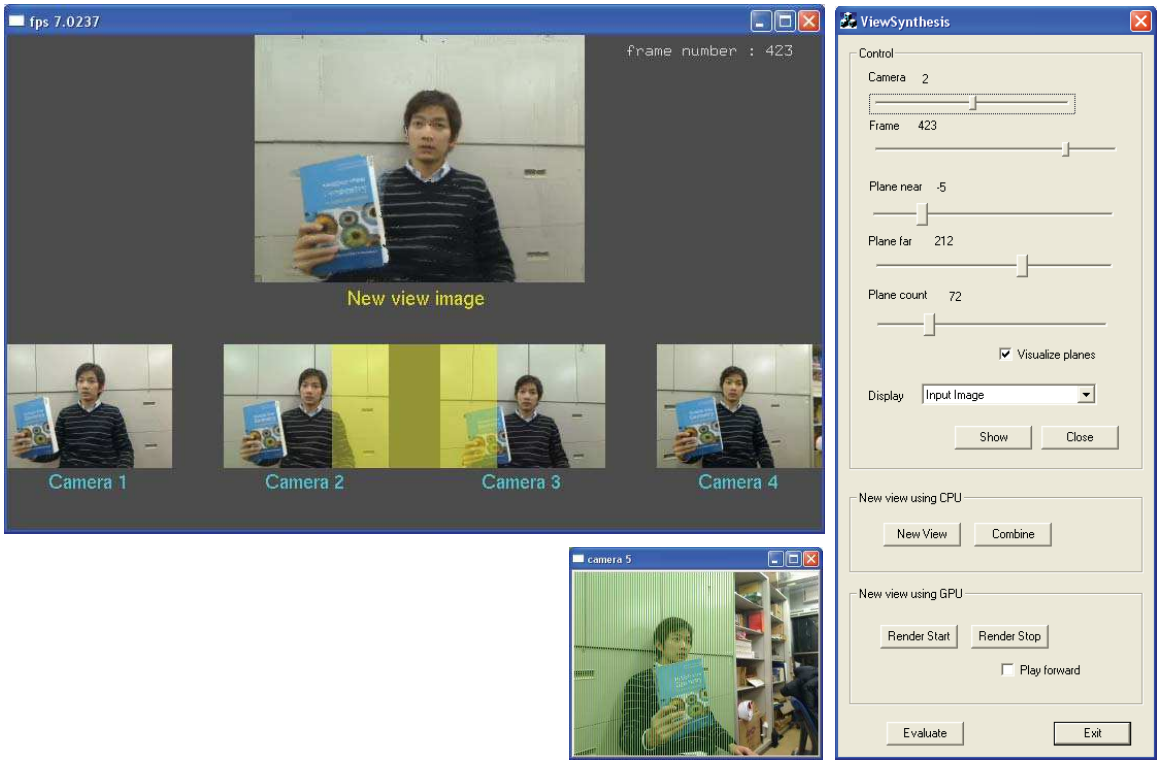


Figure 5.6: User interface.

5.5.1 Running time

Running time and quality of new view rendering depend on the complexity of the scene and on the number of planes used in the plane-sweep algorithm. The appropriate number of planes varies depending on the complexity of the scene. Using more planes makes processing time increase but usually gives a better result. In our experiment, it is shown that using 40 planes or more makes the visual result become satisfying.

Table 5.1 shows the number of planes and the running time for rendering new views using 5 webcams implemented on CPU and GPU. Both implementations are tested on Intel(R) Core(TM) 2 Duo 2.00 GHz CPU with graphic card NVIDIA GeForce 8600M GT. Implementation of our proposed plane-sweep algorithm on GPU is significantly faster than on CPU. Our system gives the same frame rates as the input webcams (30 fps.) when using 60 planes or less for scene reconstruction. When implementing the plane-sweep algorithm on GPU, most of the computation is done by the graphic

card, hence the CPU is free for the video stream acquisition and the virtual camera control.

	Number of planes					
	40	50	60	70	80	90
CPU	0.096	0.078	0.066	0.057	0.050	0.046
GPU	30.54	30.06	29.58	20.71	20.68	15.96

Table 5.1: Frame rates (frame/sec.) of our plane-sweep algorithm implemented on CPU and GPU.

5.5.2 Qualitative Evaluation

We do our plane-sweep algorithm in PGS as described in section 5.3. In our experiment, planes are defined from x axis of basis camera 2 (corresponds to R axis in PGS). *near* and far planes are adjusted so that all objects in the other cameras lie between these planes. Figure 5.7 shows new view images synthesized from our proposed method using different numbers of planes for scene reconstruction.

Figure 5.7: Result new view images using the different number of planes in plane-sweep algorithm.

Some artifacts in the rendered view come from planes discretization. The object that lies between two planes is sometimes reconstructed at the plane that is far from the actual one, so this object will be noticed as artifacts in the rendered view. One possible solution to reduce this errors is to increase the number of planes used in plane-sweep algorithm.

Figure 5.8,5.9 and 5.10 show the result new view video at several view point from the selected one input frames. We use 80 planes for reconstructing the scene. With 80 planes, quality of render views are satisfied and our implementation can still reach 20 fps. The ratio written under each figure is a virtual camera position between two real reference cameras as described in section 5.3.1. The result shows that our method gives a good visual quality and is fast enough for online VBR applications.

5.5.3 Quantitative Evaluation

This section gives objective quality measurements of our result. One camera is selected as a ground-truth reference and excluded from the plane-sweep algorithm. View at ground-truth camera is then synthesized to measure visual errors. Two metrics d^{90} proposed in [89] and PSNR (Peak Signal to Noise Ratio) are computed to measure the errors in the synthesized images. d^{90} tells us about the overall distance of misaligned pixels between synthesized image and ground-truth reference. The lower the value of d^{90}, the better the quality of the output in new view images.

If the rendered image is much different from the ground-truth, then there will likely be visual artifacts or blurred textures in the synthesized image. We measure these values for 100 consecutive input frames using camera 2 as a ground-truth reference. Camera 2 is leaved-out from plane-sweep algorithm and views at that camera are synthesized. Figures 5.11 and 5.12 show each error metric of our new view images using the different number of planes for scene reconstruction. Table 5.2 shows the average d^{90} and $PSNR$ values over 100 frames.

Number of planes	$d^{90}(pixels)$	$PSNR(dB)$
40 planes	11.000	21.738
60 planes	10.929	21.838
80 planes	10.788	21.909

Table 5.2: Error measurements for the resulting new view images (average of 100 frames).

Figure 5.8: New views produced by our proposed plane-sweep algorithm in projective grid space using 80 planes.

Figure 5.9: New views produced by our proposed plane-sweep algorithm in projective grid space using 80 planes.

Figure 5.10: New views produced by our proposed plane-sweep algorithm in projective grid space using 80 planes.

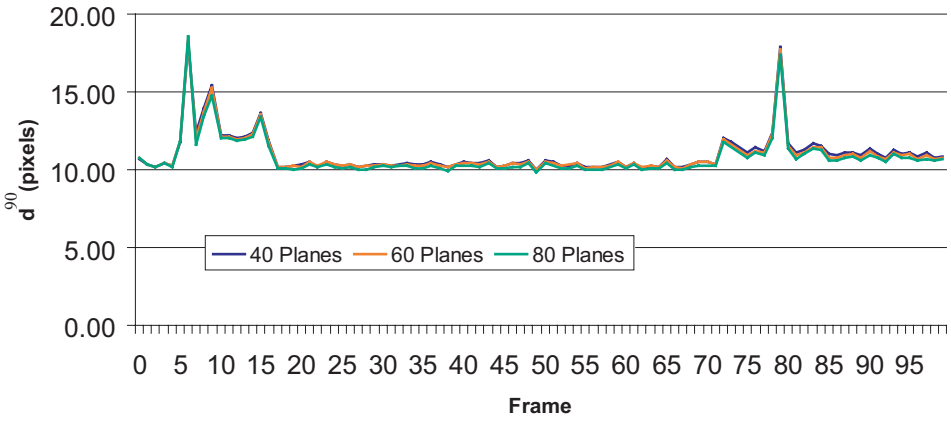


Figure 5.11: d^{90} registration error of new views.

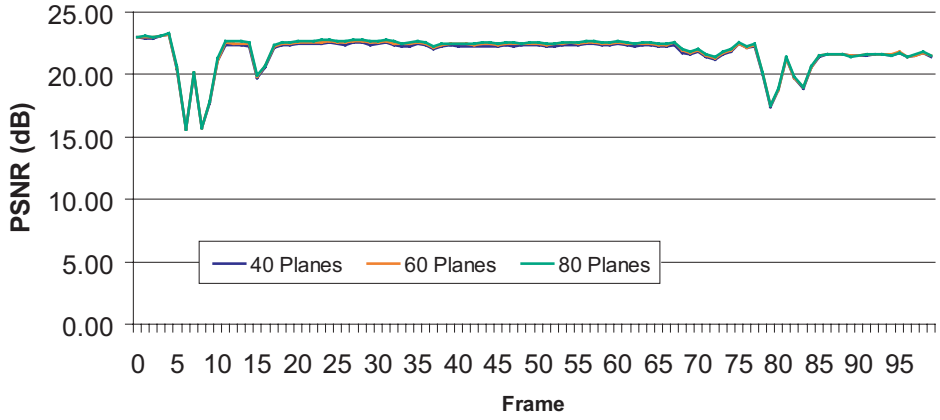


Figure 5.12: PSNR registration error of new views.

5.6 Results of Changing The Number of Cameras

The quality of new views depends on several factors, such as complexity of a scene, the number of planes and the number of cameras. This section experiments about the effect of changing the number of cameras in the free viewpoint video system. Figure 5.13 shows the cameras configuration in the experiment. We use 6 cameras to capture input videos while camera 6 is selected for defining planes.

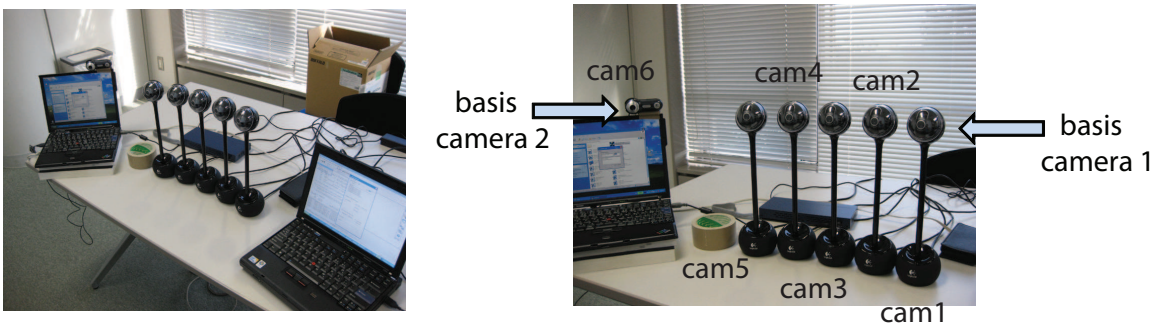


Figure 5.13: Camera configuration.

The goal is to generate new views between camera 1 and camera 5. We show the results in two cases which are

- Use all camera 1,2,3,4 and 5
- Use only camera 1,3 and 5

Then we compare the results qualitatively. Note that camera 6 is used in both camera setting. Because camera 6 is used for defining planes, so it is excluded from color consistency testing as already explained in Section 5.3.2. Figure 5.14 shows the new views generated by using 5 cameras for color consistency testing, while Figure 5.15 shows the new views generated by using only 3 cameras. Figure 5.16 shows the comparison of new views generated from 5 and 3 cameras. From the results it is clearly seen that the result generated from 5 cameras have much less artifacts. When we increase the number of cameras (while fixing the left most and the right most cameras), the baseline of the adjacent cameras becomes shorter and the new views

become nearer to the reference views. This significantly increases the visual quality of interpolated views.

The number of necessary cameras depends on several factors, such as the desired visual quality, acceptable computation time (increasing the number of cameras also increases the computation time) and scene complexity. Thus, it is difficult to answer exact number in general cases. However, increasing the number of cameras usually provides a better visual quality.

Figure 5.14: New views produced from 5 cameras using 80 planes.

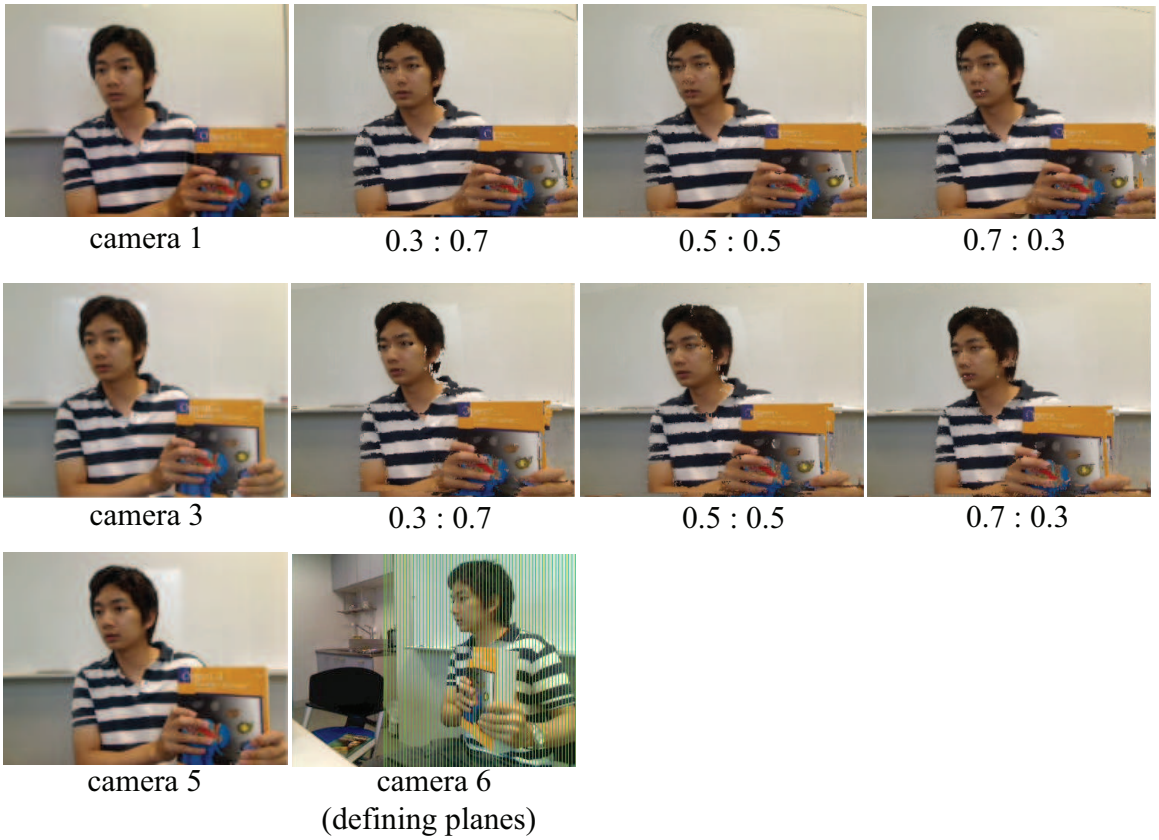


Figure 5.15: New views produced from 3 cameras using 80 planes.

Produced from 5 cameras

Produced from 3 cameras

Figure 5.16: Comparison of new views.

Chapter 6

Conclusion

6.1 Summary

There is no universal algorithm for free viewpoint video system (at least, not yet). Specific algorithms are needed for different application scenarios, e.g., with respect to camera setup (small baseline vs. wide baseline) and processing performance (on-line vs. off-line).

In this thesis, we proposed two free viewpoint methods targeting for different scene and cameras configurations. The first method described in Chapter 4 targets a large natural scene where cameras must zoom to capture a moving object in the scene at a high resolution. In this case, all cameras must be dynamically calibrated due to the changes in focal length and camera rotation. In the proposed method, we use projective grid space (PGS), as described in Chapter 3, for weak calibration. The homography between the input frame and the initial frame is used for recomputing trifocal tensors of the zoomed and rotated camera. The 3D structure of moving object is reconstructed using a silhouette volume intersection method. Background areas which are approximated as several planes are rendered together with the foreground moving object using view interpolation. This proposed scheme is suitable for a large area environment with a wide base-line between cameras.

Secondly, we proposed a method for real-time free viewpoint video using uncalibrated cameras. We use PGS as a weak calibration framework in both systems. However, in this real-time system, we use our proposed plane-sweep algorithm in PGS for reconstructing and rendering free viewpoint videos. Thanks to the implementation of our algorithm on Graphics Processing Unit (GPU), our system can achieve real-time reconstruction and rendering. In this system, the user can choose its own viewpoint between any cameras interactively from live input video cameras or from prerecorded videos. Due to the fact that the plane-sweep algorithm generally gives a good visual result when based-line between cameras are small, this real-time free viewpoint video system is suitable for a smaller area environment with a small base-line between cameras.

The most immediate applications of the proposed free viewpoint video systems are in visual entertainment. For example, our systems may be used to produce more

interesting video contents that allows the viewer to use a joystick or a mouse to freely navigate his/her viewpoint all around and through the scene.

6.2 Contribution

Our main contribution to free viewpoint video research is that we proposed methods for video based rendering from uncalibrated cameras using projective grid space (PGS) framework. Most of previous methods assume that cameras are strongly calibrated, i.e. focal length, optical axis are known. For strongly calibrating cameras, several corresponding points in 3D Euclidean space and 2D image must be measured precisely. For this reason, when there are many cameras, much effort is needed to calibrate every camera.

In contrast, projective grid space is a weak calibration framework in which geometrical relations between cameras are defined based on fundamental matrices and trifocal tensors. Estimating fundamental matrices and trifocal tensors only needs 2D-2D correspondences between input images which is much easier to measure.

In chapter 4, we proposed a novel framework for free viewpoint videos from uncalibrated calibrated cameras that allows cameras to be rotated and zoomed during captures. This differs from conventional proposed systems that usually assume that cameras are static and strongly calibrated.

In chapter 5, we proposed a new plane-sweep algorithm in projective grid space for real-time free viewpoint video. The advantage of our method compared to the conventional plane-sweep algorithm is that the near and far planes in PGS for doing plane-sweep are easily defined and visualized from basis camera 2. This is impossible for the case of the normal plane-sweep algorithm in the Euclidean space in which strong calibration is used. In that case, near and far planes must be measured from the real-world position.

6.3 Future works

Our work can be extended in several directions. We discuss here the possible extensions to each proposed method as future works.

In the method proposed for pure rotating and zooming cameras (Chapter 4), the quality of the resulting new views can be improved. One important factor to visual quality is the accuracy of the reconstructed 3D model. The reconstructed visual hull as we used in this work gives only a coarse approximation to the actual shape of the object since concave areas can not be reconstructed. More sophisticate algorithms for recovering meshes based on image textures and silhouette can be applied.

In Chapter 4, we proposed a method for generating free viewpoint videos from purely rotating and zooming cameras. Generalizing cameras constraint to rotating, zooming and *translating* cameras is not straightforward. Cameras calibration to PGS by finding correspondences between views automatically may be possible but with less accuracy and it may need to reduce the based-line between cameras. However, the challenge is that both the reconstruction and the rendering algorithms must be completely changed. We used a visual hull algorithm in which the silhouettes of the object are necessary. If cameras can be translated, a virtual background cannot be generated by warping initial background as we are doing.

In contrast, generalizing from the fixed cameras to rotating, zooming and translating cameras for the system in Chapter 5 is easier in terms of finding correspondences and 3D reconstruction algorithm. Based-line of cameras in this system is short and it is more easier to get accurate correspondences among cameras. The plane-sweep algorithm in PGS also does not need prior about the background scene, so it can adapt to moving cameras more easily. The challenge is that finding correspondences and rendering new view must be done fast enough for this real-time system.

Using our methods for a stereoscopic display is an interesting challenge. In order to generate a stereoscopic view, producing the disparity between the left image and the right image is necessary. If each dynamic object has an appropriate disparity according to its distance to the viewer, displaying it on stereoscopic display becomes possible. As uncalibrated cameras are used in this work, the distance between the

objects and the viewer cannot be obtained explicitly (not in the Euclidean distance). However, the relative distance, or distance in projective grid space, may somehow be useful for producing such parallax and can be applied to stereoscopic display.

Appendix A

Two-View Geometry

A.1 Epipolar Geometry

Consider the images $\mathbf{x}$ and $\mathbf{x}'$ of a point $\mathbf{X}$ observed by two cameras with optical centers $\mathbf{C}$ and $\mathbf{C}'$ as illustrated in Figure A.1. The camera centers $\mathbf{C}$, $\mathbf{C}'$, and $\mathbf{X}$ lie in a common plane Π, called *epipolar plane*. The rays back-projected from $\mathbf{x}$ and $\mathbf{x}'$ intersect at $\mathbf{X}$, and the rays are coplanar, lying in Π. This is an important property in searching for a correspondence.

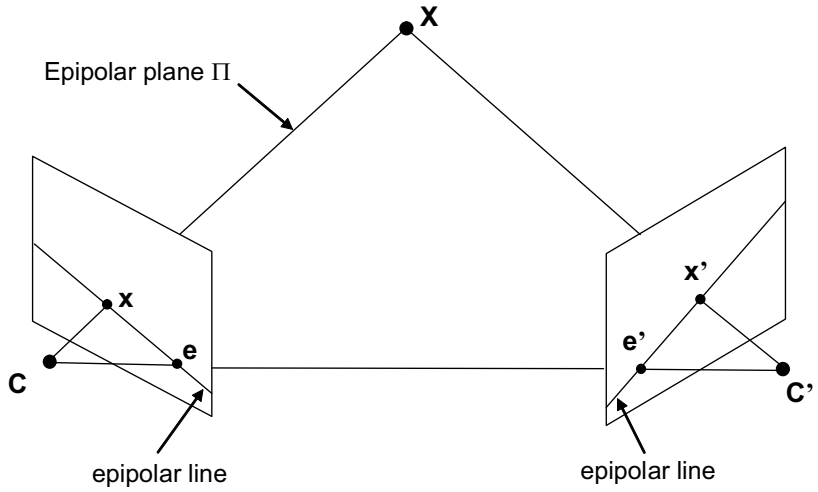


Figure A.1: Epipolar geometry

Giving a point $\mathbf{x}$ in one view and suppose that we want to find the corresponding point $\mathbf{x}'$ in the other view. Without knowing about the epipolar geometry between these two views, we have to search for a matching position from the entire image. In contrast, if the epipolar geometry between two views is available, we know that the ray corresponding to the (unknown) point $\mathbf{x}'$ lie in Π. Thus, the point $\mathbf{x}'$ must lie on the line $\mathbf{l}'$ which is the intersection of plane Π with the second image plane. This line $\mathbf{l}'$ is the image in the second view of the ray back-projected from x. It is the *epipolar line* corresponding to $\mathbf{x}$.

The terminology of epipolar geometry are

- **baseline** is the line joining optical centers.
- **epipole** is the point of intersection between the baseline and the image plane.

Epipole is the image in one view of the camera center of the other view.

- **epipolar plane** is the plane defined by a 3D point and the optical centers.
- **epipolar line** is the line of intersection of the epipolar plane with the image plane. An epipolar plane intersects the image planes of both views in epipolar lines, and defines the correspondence between the lines. All epipolar lines intersect at the epipole.

A.2 Essential Matrix and Fundamental Matrix

Both essential matrix and fundamental matrix are the algebraic representation of epipolar geometry between a stereo pair of cameras. The difference between these two is that the essential matrix is used for calibrated cameras, while the fundamental matrix is used for uncalibrated cameras.

From Figure A.1, we have a stereo pair of cameras viewing a point $\mathbf{X} = (x, y, z, 1)$ in the world, this point is projected onto image planes of the first and the second camera at $\mathbf{x} = (u, v, 1)$ and $\mathbf{x}' = (u', v', 1)$ respectively ($\mathbf{X}$, $\mathbf{x}$, and $\mathbf{x}'$ are represented in homogeneous coordinate). Assume that cameras are calibrated, then $\mathbf{x}$ and $\mathbf{x}'$ are given in normalized coordinates which we will denote as $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ respectively. In other words, $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ is given with respect to its camera's coordinate frame.

The epipolar constraint says that the vector from the first camera's optical center to the first imaged point, the vector from the second optical center to the second imaged point, and the vector from one optical center to the other are all coplanar, on epipolar plane. In normalized coordinates, this constraint can be expressed simply as

$$\hat{\mathbf{x}}^\top(\mathbf{t} \times \mathbf{R}\hat{\mathbf{x}}') = 0, \tag{A.1}$$

where $\mathbf{t}$ is the coordinate vector of the translation $\overrightarrow{CC'}$ separating the two coordinate systems, and $\mathbf{R}$ is the rotation matrix such that a free vector with coordinates $\hat{\mathbf{w}}'$ in the coordinate system of the second camera has coordinates $\hat{\mathbf{w}} = \mathbf{R}\hat{\mathbf{w}}'$ in the coordinate system of the first camera. The multiplication by $\mathbf{R}$ is necessary to transform $\hat{\mathbf{x}}'$ into the coordinate frame of camera $\mathbf{C}$.

By defining

$$\Big[\mathbf{t}\Big]_\times = \begin{bmatrix} a \\ b \\ c \end{bmatrix}_\times = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}, \tag{A.2}$$

we can rewrite Equation A.1 as a linear equation

$$\hat{\mathbf{x}}^\top([\mathbf{t}]_\times \mathbf{R}\hat{\mathbf{x}}') = \hat{\mathbf{x}}^\top \mathbf{E}\hat{\mathbf{x}}' = 0\ , \tag{A.3}$$

where $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$ is called *essential matrix* [56]. Nine coefficients of essential matrix are only defined up to scale, and they can be parameterized by the three degrees of freedom of the rotation matrix $\mathbf{R}$ and the two degrees of freedom defining the direction of the translation vector $\mathbf{t}$.

Now let suppose that the cameras are uncalibrated (intrinsic parameters are unknown), we will change the normalized coordinate from $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ to the image coordinate $\mathbf{x}$ and $\mathbf{x}'$. From the projection matrix

$$\mathbf{P} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}] = \begin{bmatrix} \alpha_x & s & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \Big[\mathbf{R} \mid \mathbf{t}\Big], \tag{A.4}$$

where $\mathbf{K}$ is camera intrinsic parameters, we can substitute $\mathbf{x} = \mathbf{K}\hat{\mathbf{x}}$ and $\mathbf{x}' = \mathbf{K}\hat{\mathbf{x}}'$ to Equation A.3 as

$$\mathbf{x}^\top \mathbf{K}^{-\top} \mathbf{E} \mathbf{K}'^{-\mathbf{1}} \mathbf{x}' = 0. \tag{A.5}$$

From equation A.5, by defining *fundamental matrix* $\mathbf{F} = \mathbf{K}^{-\top}\mathbf{E}\mathbf{K}'^{-\mathbf{1}}$ we get

$$\mathbf{x}^\top \mathbf{F} \mathbf{x}' = 0. \tag{A.6}$$

The important properties of the fundamental matrix are listed below.

- **Transpose :** If $\mathbf{F}$ is the fundamental matrix of the pair of cameras $(\mathbf{C}, \mathbf{C}')$, then $\mathbf{F}^\top$ is the fundamental matrix of $(\mathbf{C}', \mathbf{C})$
- **Epipolar lines :** For any point x in the first image, the corresponding epipolar is $\mathbf{l}' = \mathbf{F}\mathbf{x}$. Similarly, $\mathbf{l} = \mathbf{F}^\top \mathbf{x}'$ is the epipolar line corresopnding to $\mathbf{x}'$ in the second image
- **Epipole:** for any point $\mathbf{x}$ (other than $\mathbf{e}$) the epipolar line $\mathbf{l}' = \mathbf{F}\mathbf{x}$ contains the epipole $\mathbf{e}'$. Thus $\mathbf{e}'$ satisfies $\mathbf{e}'^\top(\mathbf{F}\mathbf{x}) = 0$ which can be rewritten as $\mathbf{x}^\top(\mathbf{F}^\top\mathbf{e}') = 0$ for all x. It follows that $\mathbf{e}'$ is the null-space of $\mathbf{F}^\top$. Similarly $\mathbf{F}\mathbf{e} = 0$, i.e. $\mathbf{e}$ is the null-vector of $\mathbf{F}$

A.3 Estimation of Fundamental Matrix

The fundamental matrix $\mathbf{F}$ is 3×3 rank 2 homogeneous matrix which is defined up to scale by seven independent parameters. In principle, it can be estimated from at least seven point correspondences between two views [26]. The process of estimation is known as weak calibration.

Here, we describe the 8-point algorithm, which is the simplest method of computing the fundamental matrix, introduced by Longuett Higgins [56] for the case of essential matrix. For more numerical stable solution normalized 8-point [27] is recommended.

Let $\mathbf{x_i} = (u_i, v_i, 1)^\top$ and $\mathbf{x'_i} = (u'_i, v'_i, 1)^\top$ are the i-th corresponding points in two views, equation A.6 can be rewritten using matrix elements as

$$\mathbf{u_i}^\top \mathbf{f} = 0 \ , \tag{A.7}$$

where

$$\mathbf{u_i} = [u_i u'_i, u_i v'_i, u_i, v_i u'_i, v_i v'_i, v_i, u'_i, v'_i, 1]^\top \ ,$$

$$\mathbf{f} = [f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}]^\top \ ,$$

f_{ij} is an element in the i-th row, j-th column of the fundamental matrix. Given n pairs of correspondence, we obtain

$$\mathbf{Uf} = \mathbf{0} \ , \tag{A.8}$$

where

$$\mathbf{U} = \begin{bmatrix} \mathbf{u_1}^\top \\ \vdots \\ \mathbf{u_n}^\top \end{bmatrix} .$$

When n > 8 correspondences are available, $\mathbf{F}$ can be estimated using linear least squares by minimizing

$$\sum_i (\mathbf{x_i}^\top \mathbf{F} \mathbf{x'_i})^2 . \tag{A.9}$$

To enforce the rank two constraint of fundamental matrices, $\mathbf{F}$ has singular value decomposition as

$$\mathbf{F} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top, \tag{A.10}$$

where $\Sigma = diag(\sigma_1, \sigma_2, \sigma_3)$ is a diagonal 3×3 matrix with entries $\sigma_1 \geq \sigma_2 \geq \sigma_3$, and $\mathbf{V}$ and $\mathbf{U}$ are orthogonal 3×3 matrices. Then, fundamental matrix $\hat{\mathbf{F}}$ that has rank two is computed as

$$\hat{\mathbf{F}} = \mathbf{V}diag(\sigma_1, \sigma_2, 0)\mathbf{U}^\top. \tag{A.11}$$

Appendix B

Three-View Geometry

B.1 Tensor notation

This appendix gives an introduction to the tensors for the reader who is unfamiliar with tensor notation. A tensor is a multidimensional array that extends the notion of scalar, vector and matrix. A tensor is written using an alphabet with contravariant (upper) and covariant (lower) indexes. For example, the trifocal tensor τ_i^{jk} has two contravariant indexes and one covariant index.

Considering a representation of vector and matrix using tensor notation, entry at row i and column j of matrix $\mathbf{A}$ is written using tensor notation as a_j^i, index i being contravariant (row) index and j being contravariant (column) index. An image point represented by the homogeneous column vector $\mathbf{x} = (x^1, x^2, x^3)^T$ is written using tensor notation as x^i, while a line represented using the row vector $\mathbf{l} = (l_1, l_2, l_3)$ is written as l_i.

Writing two tensors together means doing a contraction operation. The contraction of two tensors produce a new tensor where each element is calculated from a sum of product over the repeated index. For example consider a matrix multiplication $\hat{\mathbf{x}} = \mathbf{A}\mathbf{x}$, this can be written using tensor notation as $\hat{x}^i = a_j^i x^j$. This notation imply a summation over the repeated index j as $\hat{x}^i = \sum_j a_j^i x^j$.

Bibliography

[1] Eyevision, http://www.ri.cmu.edu/events/sb35/tksuperbowl.html.

[2] The matrix, http://whatisthematrix.warnerbros.com.

[3] The (new) stanford light field archive, http://lightfield.stanford.edu/index.html.

[4] E. H. Adelson, J. R. Bergen, The plenoptic function and the elements of early vision, in: Computational Models of Visual Processing, MIT Press, Cambridge, MA, USA, 1991, pp. 3–20.

[5] S. Avidan, A. Shashua, Novel view synthesis in tensor space, in: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR'97), 1997, pp. 1034–1040.

[6] S. Avidan, A. Shashua, Novel view synthesis by cascading trilinear tensors, IEEE Transactions on Visualization and Computer Graphics 4 (4) (1998) 293–306.

[7] T. Beier, S. Neely, Feature-based image metamorphosis, ACM Computer Graphics (Proceedings of ACM SIGGRAPH'92) 26 (2) (1992) 35–42.

[8] C. Buehler, M. Bosse, L. McMillan, S. Gortler, M. Cohen, Unstructured lumigraph rendering, in: Proceedings of ACM SIGGRAPH'01, 2001, pp. 425–432.

[9] J. Carranza, C. Theobalt, M. Magnor, H.-P. Seidel, Free-viewpoint video of human actors, in: Proceedings of ACM SIGGRAPH'03, 2003, pp. 569–577.

[10] S. E. Chen, Quicktime VR: an image-based approach to virtual environment navigation, in: Proceedings of ACM SIGGRAPH'95, 1995, pp. 29–38.

[11] S. E. Chen, L. Williams, View interpolation for image synthesis, in: Proceedings of ACM SIGGRAPH'93, 1993, pp. 279–288.

[12] C. H. Chien, J. K. Aggarwal, Identification of 3D objects from multiple silhouettes using quadtrees/octrees, Computer Vision, Graphics, and Image Processing 36 (2–3) (1986) 256–273.

[13] R. T. Collins, A space-sweep approach to true multi-image matching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'96), 1996, pp. 358–363.

[14] P. E. Debevec, G. Borshukov, Y. Yu, Efficient view-dependent image-based rendering with projective texture-mapping, in: Proceedings of the 9th Eurographics Workshop on Rendering, 1998, pp. 105–116.

[15] P. E. Debevec, C. J. Taylor, J. Malik, Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach, in: Proceedings of ACM SIGGRAPH'96, 1996, pp. 11–20.

[16] X. Decoret, F. Durand, F. X. Sillion, J. Dorsey, Billboard clouds for extreme model simplification, in: Proceedings of ACM SIGGRAPH'03, 2003, pp. 689–696.

[17] G. Eckert, J. Wingbermuhle, W. Niem, Mesh based shape refinement for reconstructing 3D-objects from multiple images, in: The 1st European Conference on Visual Media Production (CVMP'04), 2004, pp. 103–110.

[18] O. Faugeras, Q.-T. Luong, T. Papadopoulou, The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications, MIT Press, Cambridge, MA, USA, 2001.

[19] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395.

[20] D. A. Forsyth, J. Ponce, Computer Vision: A Modern Approach, Prentice Hall Professional Technical Reference, 2002.

[21] I. Geys, S. D. Roeck, L. V. Gool, The augmented auditorium: Fast interpolated and augmented view generation, in: Proceedings of the 2nd IEE European Conference on Visual Media Production, 2005, pp. 94–103.

[22] B. Goldlücke, M. Magnor, B. Wilburn, Hardware-accelerated dynamic light field rendering, in: Proceedings Vision, Modeling and Visualization (VMV'02), aka, Erlangen, Germany, 2002, pp. 455–462.

[23] S. J. Gortler, R. Grzeszczuk, R. Szeliski, M. F. Cohen, The lumigraph, in: Proceedings of ACM SIGGRAPH'96, 1996, pp. 43–54.

[24] O. Grau, A. Hilton, J. Kilner, G. Miller, T. Sargeant, J. Starck, A free-viewpoint video system for visualisation of sport scenes, in: International Broadcasting Conference, 2006.

[25] O. Grau, G. A. Thomas, A. Hilton, J. Kilner, J. Starck, A robust free-viewpoint video system for sport scenes, in: Proceedings of the 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2008, pp. 1–4.

[26] R. I. Hartley, Projective reconstruction and invariants from multiple images, IEEE Transaction on Pattern Analysis Machine Intelligent 16 (10) (1994) 1036–1041.

[27] R. I. Hartley, In defense of the eight-point algorithm, IEEE Transaction on Pattern Analysis Machine Intelligent 19 (6) (1997) 580–593.

[28] R. I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd ed., Cambridge University Press, ISBN: 0521540518, 2004.

[29] K. Hayashi, H. Saito, Synthesizing free-viewpoint images from multiple view videos in soccer stadium, in: Proceedings of the IEEE International Conference on Computer Graphics, Imaging and Visualisation (CGIV'06), 2006, pp. 220–225.

[30] N. Inamoto, H. Saito, Fly through view video generation of soccer scene, in: Proceedings of International Workshop on Entertainment Computing (IWEC'02), 2002, pp. 94–101.

[31] N. Inamoto, H. Saito, Arbitrary viewpoint observation for soccer match video, in: Proceedings of Visual Media Production, (CVMP'04), 2004, pp. 21–30.

[32] N. Inamoto, H. Saito, Virtual viewpoint replay for a soccer match by view interpolation from multiple cameras, IEEE Transaction on Multimedia 9 (6) (2007) 1155–1166.

[33] Y. Ito, H. Saito, Free-viewpoint image synthesis from multiple-view images taken with uncalibrated moving cameras, in: The IEEE International Conference on Image Processing (ICIP'05), 2005, pp. 29–32.

[34] S. Jarusirisawad, K. Hayashi, H. Saito, N. Inamoto, T. Kawamoto, N. Kubokawa, T. Fujiwara, The intermediate view synthesis system for soccer broadcasts, in: Proceedings of Asiagraph, Shanghai, China, 2008, pp. 7–12.

[35] S. Jarusirisawad, V. Nozick, H. Saito, Real-time video-based rendering from uncalibrated cameras using plane-sweep algorithm, in: The 3rd IEEE Pacific-Rim Symposium on Image and Video Technology (PSIVT) (Demo), 2009.

[36] S. Jarusirisawad, H. Saito, Free viewpoint video synthesis based on natural features using uncalibrated moving cameras, ECTI Transaction on Electrical Eng., Electronics, and Communications 5 (2) (2007) 181–190.

[37] S. Jarusirisawad, H. Saito, Free viewpoint video synthesis based on visual hull reconstruction from hand-held multiple cameras, in: Proceedings of ACCV'07

Workshop on Multi-dimensional and Multi-view Image Processing, 2007, pp. 39–46.

[38] S. Jarusirisawad, H. Saito, 3DTV view generation using uncalibrated cameras, in: Proceedings of the 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2008, pp. 57–60.

[39] S. Jarusirisawad, H. Saito, 3DTV view generation using uncalibrated pure rotating and zooming cameras, Signal Processing: Image Communication 24 (1–2) (2009) 17–30.

[40] S. Jarusirisawad, H. Saito, V. Nozick, Real-time free viewpoint video from uncalibrated cameras using plane-sweep algorithm, in: Proceedings of the IEEE International Workshop on 3-D Digital Imaging and Modeling (3DIM'09) (To appear), Kyoto, Japan, 2009.

[41] T. Kanade, P. J. Narayanan, P. W. Rander, Virtualized reality: concepts and early results, in: Proceedings of the IEEE Workshop on Representation of Visual Scenes, 1995, pp. 69–76.

[42] T. Kanade, P. W. Rander, P. J. Narayanan, Virtualized reality: Constructing virtual worlds from real scenes, IEEE Transaction on Multimedia, Immersive Telepresence 4 (1) (1997) 34–47.

[43] S. B. Kang, R. Szeliski, Extracting view-dependent depth maps from a collection of images, International Journal of Computer Vision 58 (2) (2004) 139–163.

[44] H. Kato, M. Billinghurst, Marker tracking and hmd calibration for a video-based augmented reality conferencing system, in: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality, 1999, pp. 85–94.

[45] J. Kilner, J. Starck, A. Hilton, A comparative study of free-viewpoint video techniques for sports events, in: Proceedings of Visual Media Production (CVMP'06), 2006, pp. 87–96.

[46] K. Kimura, H. Saito, Video synthesis at tennis player viewpoint from multiple view videos, in: Proceedings of Virtual Reality (VR'05), 2005, pp. 281–282.

[47] A. Laurentini, The visual hull concept for silhouette based image understanding, IEEE Transaction on Pattern Analysis and Machine Intelligence 16 (2) (1994) 150–162.

[48] S. Laveau, O. D. Faugeras, 3-D scene representation as a collection of images, in: Proceedings of International Conference on Pattern Recognition (ICPR'94), vol. 1, 1994, pp. 689–691.

[49] V. Lempitsky, D. Ivanov, Seamless mosaicing of image-based texture maps, in: In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07), Los Alamitos, CA, USA, 2007, pp. 1–6.

[50] M. Levoy, P. Hanrahan, Light field rendering, in: Proceedings of ACM SIGGRAPH'96, 1996, pp. 31–42.

[51] M. Lhuillier, L. Quan, Image interpolation by joint view triangulation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99), vol. 2, 1999, pp. 139–145.

[52] M. Li, M. Magnor, H.-P. Seidel, Hardware-accelerated visual hull reconstruction and rendering, in: Proceedings of Graphics Interface, 2003, pp. 65–71.

[53] M. Li, M. Magnor, H.-P. Seidel, Online accelerated rendering of visual hulls in real scenes, Journal of WSCG 11 (2) (2003) 290–297.

[54] Y. Liu, Q. Dai, W. Xu, Free viewpoint video data sets, `http://media.au.tsinghua.edu.cn/fvv.jsp`.

[55] Y. Liu, Q. Dai, W. Xu, A point cloud based multi-view stereo algorithm for free-viewpoint video, IEEE Transactions on Visualization and Computer Graphics (To appear).

[56] H. C. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, Nature 293 (1981) 133–135.

[57] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, in: Proceedings of ACM SIGGRAPH'87, 1987, pp. 163–169.

[58] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

[59] M. A. Magnor, Video-based Rendering, A K Peters Ltd, 2005.

[60] S. Mann, R. W. Picard, Virtual bellows: constructing high quality stills from video, in: Proceedings of the IEEE International Conference on Image Processing, vol. 1, 1994, pp. 363–367.

[61] R. A. Manning, C. R. Dyer, Interpolating view and scene motion by dynamic view morphing, in: Proceedings of Computer Vision and Pattern Recognition (CVPR'98), 1998, pp. 388–394.

[62] W. R. Mark, L. McMillan, G. Bishop, Post-rendering 3D warping, in: Proceedings of the 1997 Symposium on Interactive 3D Graphics, 1997, pp. 7–16.

[63] B. Matei, P. Meer, A general method for errors-in-variables problems in computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00), vol. 2, 2000, pp. 18–25.

[64] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, L. McMillan, Image-based visual hulls, in: Proceedings of ACM SIGGRPAH'00, 2000, pp. 369–374.

[65] L. McMillan, An image-based approach to three-dimensional computer graphics, Ph.D. thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA (1997).

[66] S. Moezzi, A. Katkere, D. Y. Kuramura, R. Jain, Reality modeling and visualization from multiple video sequences, IEEE Computer Graphics and Applications 16 (6) (1996) 58–63.

[67] S. Moezzi, L. C. Tai, P. Gerard, Virtual view generation for 3D digital video, IEEE Transaction on MultiMedia 4 (1) (1997) 18–26.

[68] P. Moreels, P. Perona, Evaluation of features detectors and descriptors based on 3D objects, International Journal of Computer Vision 73 (3) (2007) 263–284.

[69] Y. Mori, N. Fukushima, T. Yendo, T. Fujii, M. Tanimoto, View generation with 3D warping using depth information for ftv, Signal Processing: Image Communication 24 (1-2) (2009) 65–72.

[70] V. Nozick, H. Saito, Real-time free viewpoint from multiple moving cameras, in: Advanced Concepts for Intelligent Vision Systems (ACIVS'07), 2007, pp. 72–83.

[71] V. Nozick, H. Saito, On-line free-viewpoint video : From single to multiple view rendering, International Journal of Automation and Computing (IJAC) 5 (3) (2008) 257–267.

[72] M. Okutomi, T. Kanade, A multiple-baseline stereo, IEEE Transaction on Pattern Analysis and Machine Intelligence 15 (4) (1993) 353–363.

[73] M. Potmesil, Generating octree models of 3D objects from their silhouettes in a sequence of images, Computer Vision, Graphics and Image Processing 40 (1) (1987) 1–29.

[74] H. Saito, S. Baba, T. Kanade, Appearance-based virtual view generation from multicamera videos captured in the 3D room, IEEE Transaction on Multimedia 5 (3) (2003) 303–316.

[75] H. Saito, T. Kanade, Shape reconstruction in projective grid space from large number of images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99), vol. 2, 1999, pp. 49–54.

[76] D. Scharstein, R. Szeliski, Stereo datasets and evaluation results, `http://vision.middlebury.edu/stereo/`.

[77] D. Scharstein, R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, International Journal of Computer Vision 47 (1-3) (2002) 7–42.

[78] D. Scharstein, R. Szeliski, High-accuracy stereo depth maps using structured light, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03), 2003, pp. 195–202.

[79] H. Schirmacher, M. Li, H.-P. Seidel, On-the-fly processing of generalized lumigraphs, in: Proceedings of Eurographics, 2001, pp. 165–173.

[80] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, R. Szeliski, Multi-stereo datasets and evaluation results, `http://vision.middlebury.edu/mview/`.

[81] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, R. Szeliski, A comparison and evaluation of multi-view stereo reconstruction algorithms, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06), 2006, pp. 519–528.

[82] S. M. Seitz, C. Dyer, View morphing, in: Proceedings of ACM SIGGRAPH'96, 1996, pp. 21–30.

[83] S. M. Seitz, C. R. Dyer, Physically-valid view synthesis by image interpolation, in: Proceedings of the IEEE Workshop on Representations of Visual Scenes, 1995, pp. 18–25.

[84] J. Shade, S. Gortler, L.-W. He, R. Szeliski, Layered depth images, in: Proceedings of ACM SIGGRAPH'98, 1998, pp. 231–242.

[85] H.-Y. Shum, S.-C. Chan, S. B. Kang, Image-Based Rendering, Springer, 2007.

[86] H.-Y. Shum, L.-W. He, Rendering with concentric mosaics, in: Proceedings of ACM SIGGRAPH'99, 1999, pp. 299–306.

[87] H.-Y. Shum, S. B. Kang, S.-C. Chan, Survey of image-based representations and compression techniques, IEEE Transaction on Circuits and Systems for Video Technology 13 (11) (2003) 1020–1037.

[88] J. Starck, A. Hilton, Model-based multiple view reconstruction of people, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV'03), 2003, pp. 915–922.

[89] J. Starck, J. Kilner, A. Hilton, Objective quality assessment in free-viewpoint video production, in: Proceedings of the 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2008, pp. 225–228.

[90] J. Sun, W. Zhang, X. Tang, H.-Y. Shum, Background cut, in: Proceedings of the European Conference on Computer Vision (ECCV'06), vol. 2, 2006, pp. 628–641.

[91] R. Szeliski, Video mosaics for virtual environments, IEEE Computer Graphics and Applications 16 (2) (1996) 22–30.

[92] R. Szeliski, H.-Y. Shum, Creating full view panoramic image mosaics and environment maps, ACM Computer Graphics (Proceedings of ACM SIGGRAPH'97) (1997) 251–258.

[93] C. Theobalt, M. Li, M. Magnor, H.-P. Seidel, A flexible and versatile studio for multi-view video recording, in: Proceedings of Vision, Video and Graphics, Bath, UK, 2003, pp. 9–16.

[94] G. A. Thomas, Real-time camera pose estimation for augmenting sports scenes, in: Proceedings of The 3rd European Conference on Visual Media Production (CVMP'06), 2006, pp. 10–19.

[95] E. Trucco, A. Verri, Introductory Techniques for 3-D Computer Vision, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[96] R. Y. Tsai, An efficient and accurate camera calibration technique for 3D machine vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'86), 1986, pp. 364–374.

[97] R. Y. Tsai, A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, IEEE Journal of Robotics and Automation 3 (4) (1987) 323–344.

[98] S. Vedula, P. W. Rander, H. Saito, T. Kanade, Modeling, combining, and rendering dynamic real-world events from image sequences, in: Proceedings of International Conference on Virtual Systems and Multimedia, vol. 1, 1998, pp. 326–332.

[99] Y. Wexler, A. Shashua, On the synthesis of dynamic scenes from reference views, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00), 2000, pp. 1576–1581.

[100] O. J. Woodford, I. D. Reid, A. W. Fitzgibbon, Efficient new view synthesis using pairwise dictionary priors, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07), 2007, pp. 1–8.

[101] J. Xiao, C. Rao, M. Sha, View interpolation for dynamic scenes, in: Proceedings of Eurographics, 2002.

[102] S. Yaguchi, H. Saito, Arbitrary viewpoint video synthesis from multiple uncalibrated cameras, IEEE Transactions on Systems, Man, and Cybernetics, Part B 34 (1) (2004) 430–439.

[103] S. Yaguchi, H. Saito, Improving quality of free-viewpoint image by mesh based 3D shape deformation, Journal of WSCG 14 (1–3) (2006) 57–64.

[104] J. C. Yang, M. Everett, C. Buehler, L. McMillan, A real-time distributed light field camera, in: Proceedings of the 13th Eurographics Workshop on Rendering, 2002, pp. 77–86.

[105] R. Yang, G. Welch, G. Bishop, Real-time consensus-based scene reconstruction using commodity graphics hardware, in: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications (PG'02), 2002, p. 225.

[106] C. Zhang, T. H. Chen, A survey on image-based rendering: Representation, sampling and compression, Signal Processing: Image Communication 19 (1) (2004) 1–28.

[107] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, R. Szeliski, Breakdancing and ballet sequences, `http://research.microsoft.com/en-us/um/people/sbkang/3dvideodownload/`.

[108] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, R. Szeliski, High-quality video view interpolation using a layered representation, ACM Transactions on Graphics 23 (3) (2004) 600–608.