University of Alberta

A Cluster based Free Viewpoint Video System using Region-tree based Scene Reconstruction

by

Cheng Lei

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

Fall 2009
Edmonton, Alberta

Examining Committee

Yee-Hong Yang, Computing Science

Walter Bischof, Computing Science

Janelle Harms, Computing Science

Dil Joseph, Electrical and Computer Engineering

Kiriakos Neoklis Kutulakos, Computer Science, University of Toronto

Abstract

Free viewpoint video (FVV) has been widely speculated as one of the next generation of visual media applications. By taking advantage of camera array based multiple imaging techniques, FVV enables free viewpoint navigation to invoke a sense of "being immersed" for the viewers.

This thesis presents a cluster based FVV system which is designed as a specific application using a new proposed framework for general camera array applications. Our FVV system enables centralized workflow management and distributed computation to take advantage of the cluster's computation power for fast FVV-oriented video processing. For its implementation, effort is mainly focused on the FVV workflow stages of multi-view video acquisition and dense depth based scene reconstruction. Specifically, a new automatic method is proposed for the efficient geometric, photometric and temporal calibrations of a camera array. With this novel integrated calibration method, the use of unsynchronized cameras becomes possible and the multi-view video acquisition is made easy, which greatly facilitate the practical use of a FVV (or camera array based) system. On the other hand, the dense depth based FVV scene reconstruction is addressed as an image discrete labeling problem using a novel coarse-to-fine region-tree based framework. As a general framework, its high ranking evaluations in standard binocular stereo matching and optical flow estimation benchmarking show its effectiveness and versatility. By further extending it for general position multi-view temporal stereo and integrating with inconsistency map/background based progressive optimization, spatial-temporal consistency is enforced in a new and unified way, which greatly helps the final FVV rendering quality. Extensive experimental results show that the new system with its accompanying algorithms can provide high quality rendering results.

Acknowledgements

First of all, I wish to express my sincere gratitude to my wonderful supervisor Dr. Herb Yang for mentoring and supporting me throughout the past six years. His devotion, encouragement, knowledgeable perspective, enthusiasm and guidance have been of great help in the completion of this thesis project. I am greatly indebted to him for giving me confidence, inspirations and mental supports to overcome obstacles in the process of conducting the research.

Sincere thanks are extended to the members of my supervisory and candidacy examination committees, Dr. Dale Schuurmans, Dr. Hong Zhang, Dr. Walter Bischof and Dr. Vicky Zhao, for their helpful insights and useful suggestions for the progression of my research. Special thanks go to Dr. Kiriakos Kutulakos, Dr. Dil Joseph and Dr. Janelle Harms, for being the examiners of my thesis defense and providing valuable comments and constructive advices.

I also would like to thank the present and past members of our group, Dr. Minglun Gong, Dr. Xujie Qin, Dr. Hai Mao, Dr. Daniel Neilson, Danielle Sauer, Jason Selzer, Nathan Funk, YiLei Zhang, Xida Chen and Allen Shen, for their tremendous help, warm friendship and great fun. I enjoyed working in this extraordinary laboratory.

I am very grateful to the supporting and academic staff of the Department of Computing Science at the University of Alberta for supplying a first-class research and education environment. Special thanks go to Mr. Steve Sutphen for building the cluster and its maintenance. His clear sight and warm heart have influenced me tremendously. Special thanks also go to Mr. Baochun Bai for his selfless help and ardent encouragement.

I cannot end without thanking my parents and sisters, on whose constant encouragement and love I have relied throughout these years. Without their keeping me away from family responsibilities, I couldn't have concentrated on my study. It is to them that I dedicate this work.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1 Introduction

Interactive visual entertainment has become more and more popular. The huge progress in graphics software and hardware features and the increase in performance in hardware make it possible to render highly indistinguishable lifelike dynamic scenes of virtual world as seen in many movies or games. In addition to realism, immersiveness is another key feature for achieving their appealing viewing experience [1]. For example, as arguably the first shown in the movie "*The Matrix,*" smoothly transitioning between the viewpoints by successively switching between multiple real still images captured at different viewpoints gives the viewer the sensation of flying-around the scene with time frozen. It has been widely speculated that the next generation of visual media applications with such an ability to invoke a sense of "being immersed" by empowering them with the ability to navigate the environment freely will become ubiquitous and drastically impact many aspects of businesses and industries [2].

To make the traditional visual media such as a TV show "immersible," the viewer must be provided with interactive functions on a control to adjust important viewing parameters to his/her likings. One popular technology for this purpose is the so-called Free Viewpoint Video (FVV), which is one of the applications of ***multiple view imaging*** [3]. FVV uses multiple cameras to capture a dynamic scene from different viewpoints and allows a viewer to navigate within the dynamic real-world scene by changing the virtual viewpoint and viewing direction during video playback. Therefore it can be regarded as an extension of the more traditional interaction provided in a virtual world environment using computer graphics based techniques to one using the natural representation of appearance and motion of real world objects.

1.1 Pipeline of an FVV System

As shown in Figure 1.1, the pipeline of a typical FVV system mainly includes stages of multi-view video acquisition, scene reconstruction, content coding/transmission, and decoding/rendering.

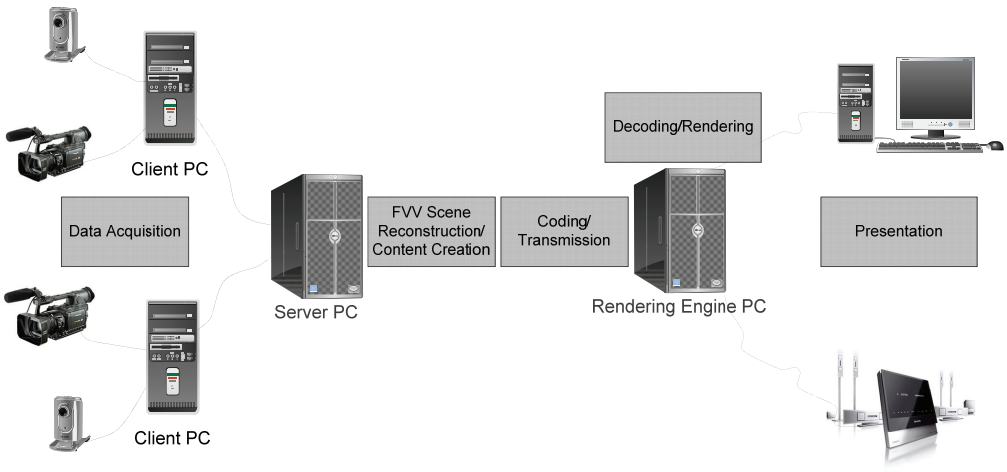


Figure 0.1.1: FVV system pipeline.

1.1.1 Multi-view Video Acquisition

The task of this stage is to capture a set of video sequences of dynamic scenes from different viewpoints using multiple cameras. The cameras are usually managed and coordinated by the controlling computers to work synchronously in integration mode or work autonomously and independently without central control in standalone mode. The number and topologic arrangement of the cameras, which determine the scene geometry sampling density, depends on the scene representation and the rendering technique to be used in subsequent stages. Since the amount of video data that needs to be captured is usually huge, even just for a small-scale FVV system, special hardware and software considerations have to be made for streaming the videos.

On the other hand, the cameras have to be synchronized and geometrically calibrated if explicit scene geometry reconstruction is needed. Moreover, to compensate for the color response differences between cameras, photometric camera calibration is also needed.

1.1.2 Scene Reconstruction and Representation

In addition to video frame color information, FVV rendering usually requires implicit or explicit 3D geometry information of the captured scene, which is crucial for synthesizing intermediate views. The chosen scene geometry representation is usually application-specific and depends on the type of scene, the desired level of realism and the available bandwidth for data transmission. The chosen scene representation has a significant impact to all other stages of the FVV pipeline.

1.1.3 Content Coding and Transmission

The potential of commercial success and the popularity of FVV in applications largely depend on whether or not they are compatible with existing public broadcasting infrastructures. Therefore, the excessive communication bandwidth and storage requirements of FVV media compared to conventional video media should be kept as low as possible. Lots of efforts such as that from the 3DAV working group established by ISO/IEC MPEG [4] were made to design "compatible and economical" encoding schemes and standards to take advantage of the correlation between multiple video streams of the same scene. Even though no FVV-oriented coding standard has been finalized so far, FVV applications can still take advantage of many widely deployed standards such as MPEG-2 or MPEG-4. New compression paradigm called multi-view video coding (MVC) has been proposed to further exploit inter-view redundancies in addition to traditional temporal redundancies [5].

1.1.4 Decoding and Rendering

In this stage, the received FVV stream data is decoded and consumed by an FVV rendering engine for synthesizing views requested by the viewer. Generally speaking, the underlying rendering techniques all belong to the ***image based rendering*** (IBR) family, but with different "genres" in taking advantage of the corresponding scene geometry information.

For collecting viewer's requests and displaying corresponding rendering, a presentation/interaction component is usually provided. FVV can be presented on

a computer monitor, a TV, a projected screen, a virtual reality HMD (headed mounted display) or other display devices. For stereoscopic FVV, special stereo-capable display devices can be used as well. The viewer can use a mouse, keyboard, remote control or head movement/gesture to control the viewpoint of the playback FVV.

1.2 Motivations and Contributions

The potential of commercialization and of the wide range of possible deployments of FVV have motivated many research groups and companies as well as this thesis work. As can be seen from the last section, FVV research and development involve many different disciplines such as imaging, computer vision/graphics, encoding/decoding and networking and there exist numerous scientific and technical challenges. Addressing them in a better way, even only partially, can greatly help the progress of FVV deployment. In this thesis, the focus is mainly in improving the current state-of-the-art in FVV oriented multiple view video acquisition and depth-based scene reconstruction, based on which a new practical FVV system is designed and developed.

In particular, as pointed out above, the choice of specific 3D scene representation is of crucial importance for the whole FVV system design and implementation. Among all the popular scene representations, the dense depth based one is the most promising one due to its lower requirements on the number of cameras and higher flexibility on the applicable scenes. For recovering the dynamic depth information from multiple view video streams, although active time-of-flight based acquisition systems [6] could be used, stereo vision based approaches are still the most popular and promising ones due to their much lower cost requirements and their potential in real time applications. As a research topic, despite many years of work has been devoted to stereo vision based depth reconstruction , very few frameworks or methods have been proposed to address the FVV-specific requirements. Therefore, in this thesis, much effort is devoted to designing and implementing novel algorithms for reconstructing high quality video depth maps for FVV rendering. As one main contribution of this research, a

general and versatile region-tree based image labeling framework has been proposed, with promising performance illustrated in its application to binocular stereo matching [7], optical flow estimation [8] and FVV-related video depth recovery [9].

Another main contribution of this thesis is in developing a new cluster based FVV system to facilitate the implementation and development of FVV algorithms. In addition to implementing the proposed FVV scene reconstruction algorithms, a novel method for the automatic and efficient camera array geometric, photometric and temporal calibrations [10] is designed for convenient and easy multi-view video acquisition and calibration.

Furthermore, we believe that the results from this research to be general, not just specific to FVV domain. Specifically, the depth based FVV scene reconstruction is addressed through formulating our method as a general framework for discrete image labeling problems [7-9]. Designing and implementing our cluster based FVV system is also done by building a general framework for camera array based applications [11]. In this way, the applicability of this thesis work to other areas is greatly extended.

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows. In the next chapter, related work is reviewed and some basic concepts and mathematical notations are introduced. Then a camera array application framework is presented in Chapter 3 based on which our FVV is designed and developed. In Chapters 4 and 5, the issues of video synchronization and camera array total calibration are addressed by implementing the multi-view video acquisition module for our FVV system. Then in Chapter 6, a novel region-tree based labeling framework is presented with its applications in binocular stereo matching, optical flow estimation and spatial-temporal consistent FVV video depth map recovery illustrated in Chapters 7, 8 and 9 respectively. Conclusions and possible future works are given and discussed in Chapter 10.

Chapter 2
Background and Related Work

As expected to evoke a revolution in the next generation of video technology, extensive research has been conducted on capturing, processing and rendering FVV in the last two decades. With new technologies kept emerging or advancing in the closely related disciplines such as imaging, computing and rendering, the implementation of real-world FVV application has become more and more feasible and cost-effective.

In the following, previous research and development work related to FVV will be briefly reviewed, especially the ones on FVV data acquisition, scene representation and reconstruction and rendering, which are the main focus of this thesis.

2.1 Multi-view Video Acquisition

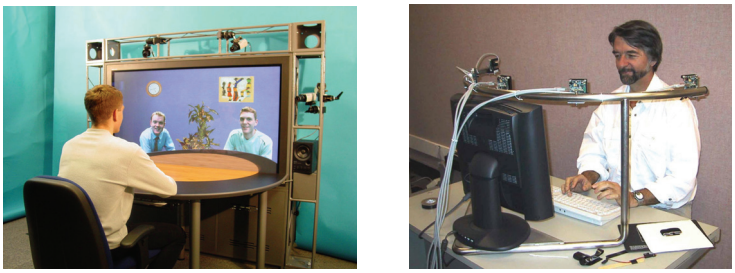
Most existing FVV systems use a camera array for multiple view video acquisition. Such examples include the Virtualized Reality system at CMU [12], the view-dependent visual hull system at MIT [13], the Blue-C system from ETH Zürich [14], the view interpolation system at Microsoft Research [15] and the Stanford multi-camera array [16].

(1) Camera array based scene sampling

Multiple view video acquisition is equivalent to a sampling process of the captured 3D dynamic scene. Its sampling pattern and density are determined by the topological arrangement and the number of cameras in the array.

Usually the camera array's topological arrangement is application-specific and depends on how freely the viewer is allowed to navigate his/her viewpoint when viewing the FVV rendering. For example, FVV based teleconferencing

applications need less viewpoint freedom and usually take a near-parallel facing arrangement. Specifically, Kauff and Schreer [17] use four cameras mounted around a display to capture a conference scene as shown in Figure 2.1 (a). While in [18], as shown in Figure 2.1 (b), virtual views are synthesized using 5 cameras on an arc for immersive teleconferencing. On the other hand, for an all-around viewpoint freedom, a dome like arrangement is usually used, as done in [12, 14].

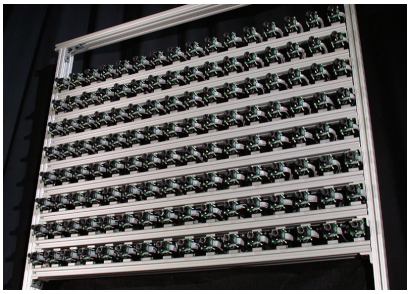


(a) (b)

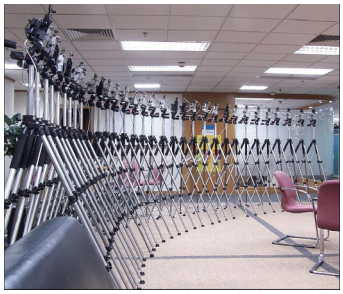
Figure 2.1: FVV based teleconferencing systems.

In addition to topological arrangement, the number of cameras used, which determines the sampling density of the captured scene, has a strong relationship with other FVV pipeline stages. As an example, it is possible to achieve extremely high sampling density by using a large number of cameras so that even direct linear viewpoint blending between adjacent cameras is enough for FVV rendering. But its high implementation cost makes this approach impractical in most cases. So there are always tradeoffs between system cost and the quality in rendering realism and the flexibility in selecting viewpoint.

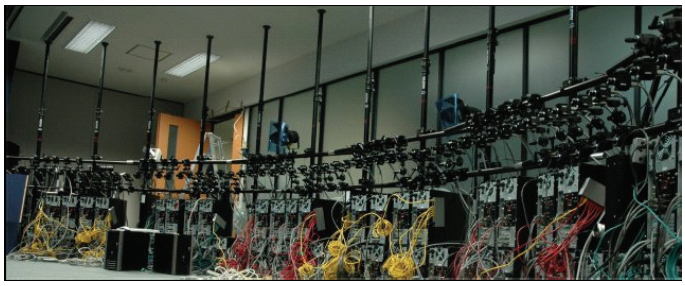
Usually the sampling density is determined by the 3D scene geometry representation and the reconstruction approach used by the system. For example, to use the "ray based" scene representation (Section 2.2), the sampling density must be high. In [16], as show in Figure 2.2 (a), a large scale camera array using 100 CMOS cameras is built and used for FVV rendering, high frame-rate video and video super-resolution. In [19], as shown in Figure 2.2 (b), up to 100 high definition cameras are used to enable viewer free control of the viewpoint of a real time dynamic 3D scene. In [20], a 32-camera array is used to provide a real-time interactive multi-view video service.



(a) Stanford camera array



(c) MSR camera array



(b) Camera array from [15]

Figure 2.2: Examples of large scale camera arrays.

If more processing overhead is allowed for reconstructing the scene geometry, much lower sampling density can be used. For example, as shown in Figure 2.3 (a), Zitnick et al. [15] use 8 cameras arranged on a roughly circular arc for dense depth reconstruction and high quality view interpolation is achieved based on layered based rendering. Shown in Figure 2.3 (b), Carranza et al. [21] use 7 cameras for 3D human model and motion capture based FVV synthesis.

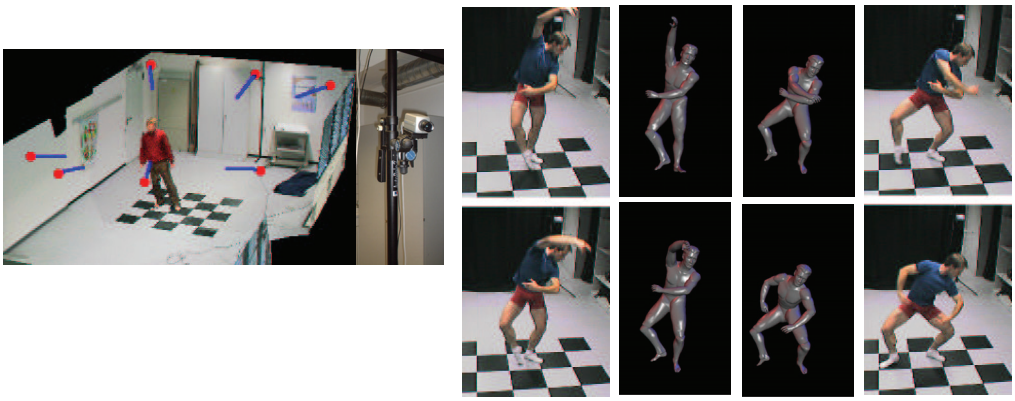
(2) Video Streaming

Based on the working mode and type of the cameras used, the captured video data can be streamed into local media such as video tapes or non-local media such as hard disks/RAMs of the controlling computers through high-speed cables or wireless network. The first approach is suitable only for offline FVV applications, while the second one is more flexible and has the potential for real time FVV applications.

Since the amount of data that needs to be captured is usually huge, even just for a small-scale camera-array, most FVV systems rely on using multiple computers. However, to lower the cost and deployment difficulties, on-the-fly

MPEG video compression can be done using embedded compression hardware. For example, as done in the Stanford camera-array [16], it manages to stream compressed video data from over 100 cameras to a hard disk array by using as few as one computer per 25 streams. With the cameras integrated with even more powerful and general computing capabilities in the future, the number of computers required can be further reduced.

(a) 8-camera based FVV system from [15]



(b) 7-camera based FVV system from [21]

Figure 2.3: FVV systems using lower sampling density.

(3) Camera Synchronization

Since most FVV applications need to establish correspondences between video frames, synchronization between cameras are very important, especially for scenes with fast moving objects. When a camera-array works in the integration mode, the synchronization can be achieved by using hardware based mechanism. Usually such approaches need a specialized control unit to broadcast external signals to trigger the exposure shutters of cameras. For example, the synchronization unit from Point Grey Research [22] can synchronize multiple cameras on different 1394 buses. For cameras interconnected on the same 1394 bus or through TCP/IP network, by means of the 1394 bus sync-signal or TCP/IP

signaling [23], hardware-software hybrid camera synchronization approaches can be used. For example, a series of 1394 cameras can be daisy chained together on the same 1394 bus and the exposure of the cameras can be synchronized using synchronization software [24]. In [25], a server–client architecture with a special error-checking technique is used for synchronization. By estimating and accounting for the network latency, synchronization between the times for each camera to receive the trigger signal is achieved.

On the other hand, when the cameras have to work in the standalone mode, offline camera (video) synchronization can be done using software-only mechanism, which does not require the special setup procedure or the equipment necessary for hardware based synchronization.

(4) Multiple camera calibrations

In additional to correlating cameras temporally through synchronization, most FVV applications also require to recover the geometric and photometric correlations between cameras which are mainly used in the scene reconstruction and rendering stages.

(a) Pin-hole camera model

In this thesis, we use the pin-hole camera model to describe the mathematical relationship between the coordinates of a 3D point and its projection onto the image plane. In particular, as shown in Figure 2.4, the projected 2D point $\boldsymbol{x}$ of a 3D point $\boldsymbol{X}$ is the intersection of the image plane of the camera with the line joining its optical center $\boldsymbol{C}$ and point $\boldsymbol{X}$ in question. The line that passes through the camera optical center $\boldsymbol{C}$ and is perpendicular to the image plane is called the ***principal axis***, and the intersection with the image plane is called the ***principal point***. The distance between the optical center and the principal point is the camera's ***focal length*** f.

For a pin-hole camera, the relation between $\boldsymbol{X}$ and $\boldsymbol{x}$ can be described by the ***perspective projection***. That is,

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \boldsymbol{x} \simeq \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]\boldsymbol{X} \tag{2.1}$$

wherein $\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$ and $\boldsymbol{X} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$ are homogenous coordinate of $\boldsymbol{x}$ and $\boldsymbol{X}$ respectively and $\simeq$ means "equal up to a non-zero scale factor." $(\boldsymbol{R}, \boldsymbol{t})$, which represents the camera's rotation matrix and translation vector w.r.t a reference coordinate frame, is defined by 6 extrinsic parameters.

Furthermore, $\boldsymbol{K} = \begin{bmatrix} f_u & \alpha & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$ is called the ***calibration matrix*** and $f_u = \frac{f}{s_x}$, $f_v = \frac{f}{s_y}$, $\alpha = \frac{f}{s_x}\cot\theta$, u_0 and v_0 are the camera's five ***intrinsic parameters***, which models the camera's optics properties. s_x and s_y are the physical width and height of a pixel on the image plane, which are determined by the camera's film size or its electronic sensor dimensions. α is the skew factor, which is determined by the cosine of the angle θ between the *x* and *y* axes of the image plane, and is usually 0 for orthogonal axes. u_0 and v_0 are, respectively, the *x* and *y* coordinates of the principal point in pixels. In this thesis, we always assume $\alpha = 0.0$ for simplicity. Equation (2.1) can also be expressed in another simpler form as

$$\boldsymbol{x} \simeq \boldsymbol{P}\boldsymbol{X} \tag{2.2}$$

where the 3×4 matrix $\boldsymbol{P}$ is called the ***projection matrix***. In many computer vision and graphics research, using such a linear projection transform based expression can simplify many derivations of the relations between 3D and 2D coordinates.

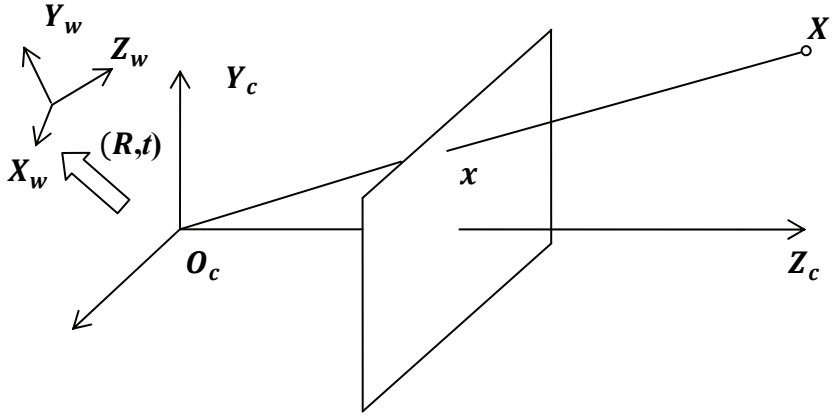


Figure 2.4: The pin-hole camera model.

(b) Geometric calibration

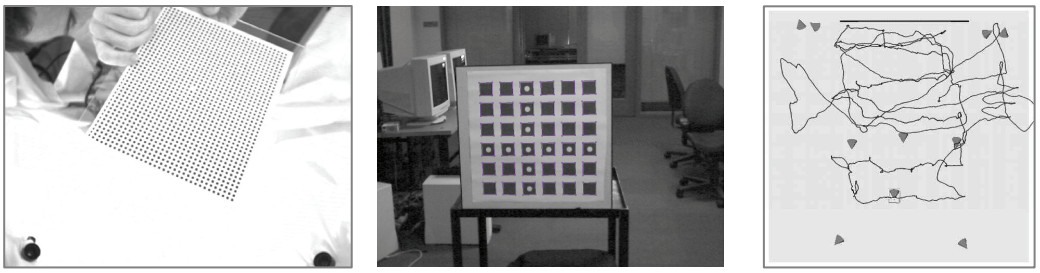
Geometric calibration of a camera is an indispensable step for FVV applications that require explicitly extracting 3D geometric information from 2D video frames. The purpose of geometric calibration of a camera is to recover the camera's intrinsic and extrinsic parameters from the captured image information. With such parameters known, the 3D-to-2D projection model of the camera can be established. The accuracy of these geometric parameters have to be high enough otherwise errors and artifacts may show up in subsequent scene geometry reconstruction and rendering steps.

Extensive research has been done for single camera calibration. Existing techniques can be roughly grouped into two categories: the intrusive ones [26-28] which rely on using artificial calibration patterns with known geometric information to facilitate feature extraction; and the non-intrusive ones [29, 30] which are based on the automatic natural feature matching using self-calibration techniques. In the first category, the plane based algorithm [28] is the most popular due to its simplicity of use, flexibility and high accuracy. As well, its many publicly available implementations [31, 32] also help its popularity.

On the other hand, automatic geometric calibration for camera arrays is challenging. Due to application-specific requirements on the topological arrangement of the cameras, their common field of views could be very small. Moreover, the use of heterogeneous or low-cost non-synchronized cameras may further complicate the geometric calibration process due to inaccurate synchronization.

Most of the existing camera array geometric calibration methods reply on the multiple-view geometric constraints on the spatially (and temporally) matched features (if dynamic). The often used constraints include plane-induced homography [33, 34], the fundamental matrix [35] or the tri-focal tensor [36]. In particular, some efforts have been made to extend [28] for camera array applications in a direct-scaling way [37] or by customizing the calibration pattern used [34, 36, 40]. Instead of using a planar pattern, the authors in [35, 38, 39] use

the so-called virtual calibration object idea. Indeed, it is based on the observation that, for synchronized stationary cameras, the trajectory of a dynamic point in a time span is equivalent to a snapshot of many static points at a time instant. Then such virtually static points can be used as a virtual rigid object whose matching between different cameras is considerably easy and robust. To generate virtual calibration objects, in [35, 38], the active illumination technique is used, while in [39] a real moving people or object is tracked using the extended Kalman filter (EKF) approach.



(a) Customized calibration pattern based (b) Virtual object based method []

Figure 2.5: Examples of camera array geometric calibration methods.

(c) Photometric calibration

It is very difficult, if not impossible, to guarantee that all the cameras in a camera array have exactly the same color response characteristics, even for the ones with the same model purchased at the same time from the same manufacturer. Therefore in many FVV applications, photometric calibration is also needed to correct the deviation of the sensed or reproduced colors from the "true" ones due to the non-linear color-response function of the camera, the on-board color processing artifacts or other sources. Ensuring such color consistency is very important for obtaining accurate FVV scene reconstruction and getting rid of view dependent color variations in FVV rendering.

Accurate photometric calibration is challenging and it is common practice to use the Macbeth color checkerboard which has 24 color pigment chips for calibration. To enforce color consistency among different cameras, the sensed pigment colors of the same color by different cameras should be as close as

possible after calibration. Usually it is done by adjusting the camera's gain and offset settings or post-processing of color of the captured video frames [41, 42].

2.2 Scene Representation and Reconstruction for FVV

To achieve the viewpoint navigation freedom, an FVV application usually has to know the 3D scene geometry to some extent for presenting viewers synthesized images with realistic viewpoint transitions through changes such as parallax, object shading and lighting. Therefore, reconstructing 3D dynamic scene from multiple video streams in a suitable representation format is of crucial importance to all FVV applications.

The FVV scene representation formats can be roughly categorized into the ray-based, depth-based and object-based representations.

2.2.1 Ray based Representation

The ray-based representation originated from IBR research [43], which as a convergence of computer vision and computer graphics disciplines aims to render photo-realistic images directly from real world photographs. Compared to the traditional 3D geometry/appearance description based approach, IBR achieves photo-realistic rendering with much less computation load and human effort, which is the main reason why IBR has attracted much attention since the early 90's.

In particular, in IBR, a 3D scene is implicitly modeled by a collection of images, or equivalently light rays filling the scene defined by the so-called 7-dimensional plenoptic function [44]. The plenoptic function represents the radiance of each light ray with 7 parameters, namely, the wavelength λ, the camera position (C_x, C_y, C_z) and the direction defined by the azimuth angle θ and the elevation angle ϕ, and time t. So the image formation process can be regarded as intersecting a camera's image plane with the ray space, with each pixel's color defined by the corresponding light ray passing the camera's optical center. Ray based scene reconstruction is equivalent to reconstruct the plenoptic function based on the input image samples. With the plenoptic function known,

synthesizing novel images is straightforward as function evaluation using the corresponding parameters.

However, fully reconstructing a 7-dimensional plenoptic function, if not impossible, needs a huge amount of sampling storage and workload. In practice, the simplified lower-dimensional plenoptic function variants are often used. In the following, we review only the ones that are most relevant to FVV.

(a) 5D parameterization

By assuming that wavelength λ and time t are fixed, the plenoptic function can be reduced into a 5D function. For example, in [44], a panning video camera is used to sample a static scene at several viewpoints. The captured images are re-projected onto a cylindrical surface. By recovering the optical flow fields and cylindrical epipolar geometry between the re-projected images, novel views can be synthesized.

(b) 4D parameterization

By further assuming light intensity remains constant along a ray, Levoy and Hanrahan [45] simplify the plenoptic function into a 4D one. This is the well-known light field rendering using two-plane parameterization for the plenoptic function. When synthesizing novel views, the two intersection points between each light ray to be synthesized and the two planes are used to interpolate the ray intensity using the captured images. Similarly, in the work of lumigraph [46], for each face of the bounding-box of the captured scene, two such planes are used as well. Different from light field rendering [45], in lumigraph, the cameras are allowed to be in general position. However, the object must be placed in a box with patterns for performing geometric calibration at each camera position. Furthermore rough volume reconstruction is done for depth-corrected rendering.

For ray-based FVV applications, light field rendering has also been extended to dynamic scenes in [47-50].The corresponding scene geometry reconstruction is equivalent to a ray indexing process. Based on the dynamic light field parameterization, each pixel of all video frames at a time instant is indexed as a

light ray with the corresponding discrete plenoptic parameters, all together forming the so-called "ray database." To render a novel view at a time instant according to a queried viewpoint, for each pixel in the image, the parameters of the queried ray are estimated and used to lookup in such "ray database." The pixel's color is then determined by directly using the closest ray or integrating/re-sampling using the nearby rays.

The main challenge of using the ray-based representation is in the required high sampling density for reconstructing the original continuous plenoptic function from discrete video frames and the corresponding storage and transmission overhead. Since there is no explicit (or very rough if any) scene geometry involved to compensate for view parallax effect of non-planar scenes, the sampling density has to be high enough or the viewpoint freedom must be limited otherwise artifacts will appear. Unfortunately, though building large scale camera-array is possible, the physical dimension of the cameras make it difficult to satisfy the space density requirement. To address this issue, researchers invent the approach of using multiple micro-lenses with a single camera [51]. However, the image resolution/quality and viewpoint navigation range is much limited.

2.2.2 Object based Representation

In this category, a dynamic scene is represented as dynamic 3D objects with geometry and appearance information. To synthesis a novel view, it just needs to match the virtual CG camera to the requested viewpoint and viewing direction and render the reconstructed scene as in classic CG rendering. In this way, full navigation freedom can be trivially achieved.

In the following, four popular 3D object representations are briefly reviewed.

(a) Mesh based representation

Polygonal mesh based representation has been the most universal 3D object definition format used in the computer graphics industry [52] and many mesh simplification and compression schemes have been developed [53,54] to meet the continuously advancing storage and rendering challenges. Mesh-oriented

hardware acceleration is now ubiquitously available and very complex 3D geometry can be rendered in real-time on low-end desktop or even handheld devices.

To model time varying or deforming objects using polygons for FVV, mesh animation and morphing techniques [55, 56] are used. Instead of using separate mesh for each time instant, static mesh connectivity is usually used for all video frames and only vertex position information is updated between frames for better storage efficiency. To avoid FVV rendering artifacts due to the use of static mesh, Kircher and Garland [57] proposes a progressive scheme to efficiently produce incremental level of detail (LoD) approximation for all video frames.

Reconstructing 3D surface meshes accurately from multiple video streams is nontrivial and has been a heavily researched topic in computer vision. Many algorithms such as ***image based modeling*** (IBM) [58-59] can reconstruct 3D models from videos by establishing sparse or dense correspondences between multiple views.

While for FVV rendering, the view-dependent video texture mapping technique [60] is used for realistic mesh appearance rendering, by which, for each time instant, textures from all available views are mapped onto the mesh and weighted in relation to a given geometric criterion. This weighting ensures a seamless fading between the object projection appearances from different viewpoints, resulting in better rendering quality matching the real lighting condition and reflectance changes when transitioning between viewpoints.

(b) Voxel based representation

Voxel based representation uses a set of so-called voxels as the minimal 3D space units to parameterize the scene reconstruction volume [61]. Each voxel has a binary visibility flag to indicate if it is empty. A non-empty voxel is associated with its appearance properties of the surface segment of the object occupying it.

To reconstruct a scene surface using voxels, the well-known space carving [62,63] or voxel coloring approaches [64-66] are often used, which starts with a completely solid volume and iteratively carves away invisible voxels based on

photo consistency. In the work of Snow et. al. [67], the volume reconstruction problem is reformulated as a voxel visibility labeling optimization problem and solved using graph cuts.

To render voxel based FVV content, usually it still needs to convert the volumetric geometry into textured meshes.

(c) Point based representation

A 3D object can also be represented using the primitive of points, particles or surfels [68,69]. In point based representation, there is no explicit topology or connectivity information and each 3D point is independent and has its own coordinates, color, normal and other rendering-oriented attributes [14, 70]. Currently, the point based representation has become more popular due to its better ability and efficiency in handling highly detailed complex 3D geometry. Progressive schemes have been proposed using hierarchical data structure such as the octree [68] for better performance. Compared to mesh based representation, the point based representation can easily implement surface dynamics since there is no connectivity change for geometry animation or morphing.

One example FVV system using the point based scene representation is in the work of Gross et al. [14]. Its rendering consists of the steps of reconstructing the continuous surface from surface samples, surface filtering and re-sampling according to viewing parameters. During synthesis, the splatting approach [71] is used, which projects each point with an orientation-adapted disc or ellipsoid onto the desired image plane. By properly choosing the splat shape and distribution, trade-off can be made between rendering quality and performance.

(d) Parametric model based representation

For a specific class of scene objects such as the human body, a parametric model with highly detailed geometry given as a priori can be used to ameliorate the difficulty of general object reconstruction [21]. By matching and fitting the parametric model to the captured videos with much fewer degrees of freedom compared to general reconstruction, more robust and convincing reconstruction

can be achieved. On the other hand, temporal coherence can be maintained by limiting only the maximum range of parameter changes between time instants. However, since it is difficult, if not impossible, to obtain the exact geometry model for general scene objects, special process has to be performed to correct the possible geometry error when projecting the reference video frames as textures onto the model during rendering.

2.2.3 Dense Depth based Representation

Dense depth based representation is often called 2.5D (not true 3D) representation [72]. The scene geometry is encoded by depth maps with the same spatial-temporal resolution as the video streams so that each pixel in each video frame has the associated depth information.

As a tradeoff between the object based and ray based representations, a depth based representation needs more explicit scene reconstruction than the ray-based one, but requires much lower sampling density. Compared to the object based representation, the per-pixel depth information is usually easier to obtain.

Existing FVV systems using depth based representation include the work of Zitnick et al., Gong and Yang, and Fehn [15, 73, 74]. The accuracy of depth map and the strategy to handle occlusions and dis-occlusions due to viewpoint changes are crucial to the rendering quality.

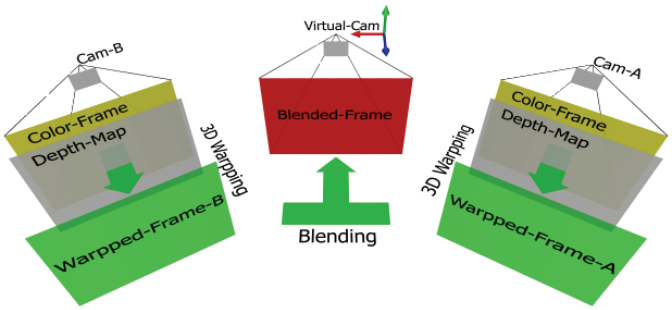
To recover accurate per-pixel depth information, hardware-based or software-based approaches can be used. In particular, specialized hardware such as range scanners [75], Z-Cam [76], structure light projectors [77] etc. have proved their effectiveness. On the other hand, stereo vision techniques provide a low-cost and general software-based solution.

Most depth based FVV rendering techniques used in practice are based on 3D image warping [78-80], which is also originated from IBR. Based on the pixel warping direction between the reference image and the destination image, the methods can be classified into two categories: forward and backward 3D image warping.

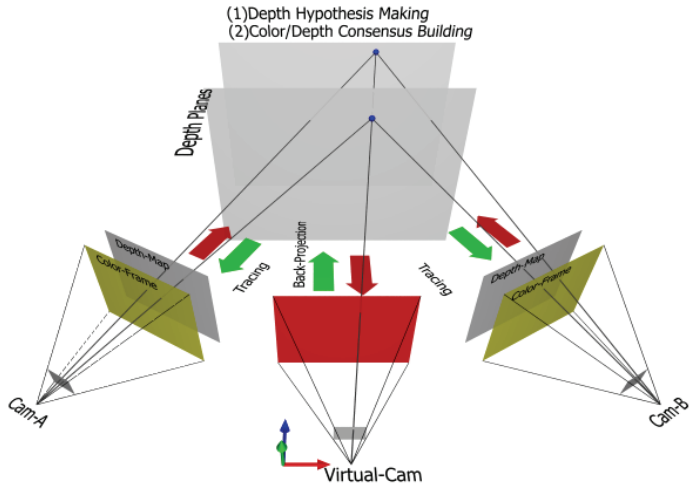
(a) Forward 3D image warping

As shown in Figure 2.6, in forward 3D image warping, each pixel in the warped image is back-projected as a 3D point which is the intersection of its back-projection ray and a depth plane corresponding to the pixel's depth. Given a destination camera, all the back-projected 3D points are re-projected onto its image plane, forming a geometrically-consistent warped virtual view. Since more than one source pixels can be warped to the same destination pixel, a Z-buffering mechanism is commonly used to resolve such conflicts. Moreover, due to parallax effects between the source and the destination cameras and pixel rasterization rounding errors, usually there will be some uncolored pixels in the warped image, forming holes. Specific to FVV rendering, a common solution is to warp from more than one nearby viewpoint and then weigh or blend all the warped views together. If necessary, a hole filling post-processing can be done by interpolating neighboring pixels of each remaining hole.

The advantage of forward warping is its high efficiency because its computation complexity depends only on the image dimensions and not on the scene depth complexity. Its main disadvantage is its higher requirement on depth accuracy. Zitnick et al. [15] show a practical example using forward 3D warping approach for FVV rendering. Different from conventional methods, it is assumed in [15] that the reference image consists of two layers, i.e., the main layer and the alpha-matted boundary layer. In warping, these two layers are warped independently and then blended, through which the visual quality greatly improves.



(a) Forward 3D warping



(b) Backward 3D warping

Figure 2.6: Comparison of forward & backward warping.

(b) Backward 3D image warping

In contrast to the forward one, backward 3D image warping works in a ray-tracing-like way [80]. For each pixel to be synthesized in the destination image, the back-projected ray is built similar to the forward method. However, since there is no depth map information available for the synthesized novel view, all the possible depths will be checked to find an optimal one which the projected pixels in all the reference images of the corresponding 3D point are most consistent with in the color and depth.

The advantage of this approach is that the above mentioned hole problem will be greatly ameliorated. However, it is more computationally intensive, especially when the scene depth complexity is high and there is no prior knowledge on what

the depth range a "being traced" pixel will be in. As a practical example of using backward 3D image warping approach for FVV rendering, the zero-crossing of the difference between the observed and the known disparity is used to determine the color [73].

In this thesis, the depth based FVV scene representation and forward warping based rendering is preferred. How to accurately and efficiently reconstruct the dynamic scene as dynamic depth maps from multiple video streams for FVV rendering is one of the main topics of this work. For this, the stereo vision based approach is applied. Since stereo has been one of the most heavily investigated topics in computer vision research and a huge number of formulations and algorithms have been proposed in the literature, an exhaustive survey on this topic is beyond the scope of this chapter. Instead, only a brief overview is provided here. In the corresponding chapters, more detailed review on the most relevant methods will be given.

2.3 Stereo Vision

In the following, we roughly categorize the dense stereo matching techniques based on the data complexity involved, that is, binocular stereo, multi-ocular stereo and temporal stereo.

2.3.1 Binocular Stereo

Although not being directly applicable to multiple-view based FVV applications, the classic binocular stereo matching forms the foundation of the design and development of any FVV oriented depth recovery method.

(a) Epipolar Geometry

Binocular stereo matching uses the epipolar geometry constraint to find point correspondences between two images. As the simplest and most widely used model of multiple view geometry, the epipolar geometry is the intrinsic projective geometry between two views.

In particular, as shown in Figure 2.7, for a 3D point $\boldsymbol{X}$ imaged by two cameras at $\boldsymbol{x}$ and $\boldsymbol{x'}$, respectively, it can be easily seen that image points $\boldsymbol{x}$ and $\boldsymbol{x'}$, 3D point $\boldsymbol{X}$, and camera centres $\boldsymbol{C}$ and $\boldsymbol{C'}$ are coplanar. That is, the rays back-projected from $\boldsymbol{x}$ and $\boldsymbol{x'}$ intersect at point $\boldsymbol{X}$ and such rays lie on the same plane $\boldsymbol{\pi_X}$. Then to search for the correspondence of $\boldsymbol{x}$ in view $\boldsymbol{C'}$, since the corresponding plane $\boldsymbol{\pi_X}$ can be determined by the baseline and the ray defined by $\boldsymbol{x}$ and also it is known that the ray corresponding to the (unknown) point $\boldsymbol{x'}$ must lie in $\boldsymbol{\pi_X}$, hence the point $\boldsymbol{x'}$ must lie on the intersection line $\boldsymbol{l'}$ of $\boldsymbol{\pi_X}$ with the image plane of $\boldsymbol{C'}$. Line $\boldsymbol{l'}$ is just the image of the ray back-projected from $\boldsymbol{x}$ in view $\boldsymbol{C'}$ and it is called the ***epipolar line*** corresponding to $\boldsymbol{x}$. In terms of stereo matching, such a geometric relation restricts the candidates of the point corresponding to $\boldsymbol{x}$ to the points on the epipolar line of $\boldsymbol{x}$ instead of the whole image, which greatly decreases the computation complexity required.

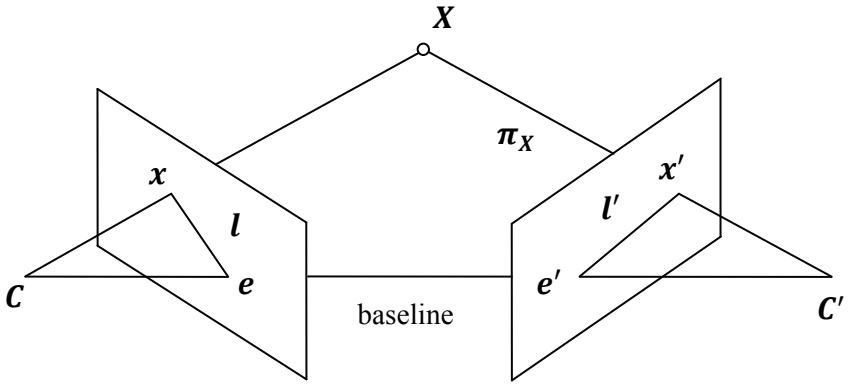


Figure 2.7: Epipolar geometry.

Algebraically, the epipolar geometry is encapsulated by the so-called ***fundamental matrix***, which is a rank 2 3×3 matrix. That is, for any pair of corresponding points $\boldsymbol{x}$ and $\boldsymbol{x'}$ in two views, the fundamental matrix $\boldsymbol{F}$ satisfies the following condition

$$\boldsymbol{x'^T F x} = 0. \tag{2.3}$$

Based on the algebraic constraints provided by Equation (2.3), the *fundamental matrix* for two uncalibrated images can be calculated from at least 7 corresponding point pairs. Both linear or robust non-linear algorithms have been

proposed [81] and their implementations are commonly available in many computer vision libraries such as the OpenCV [31].

(b) Classic Binocular Stereo

In classic binocular stereo, the left and right images are captured by two identical cameras arranged in parallel so that all the epipolar lines become horizontal. The matching is simplified to a 1D correspondence search on the corresponding scanline and the relative depth information is encoded as ***disparity***, that is, the 1D displacement between matching points. The closer a 3D scene primitive is to the camera, the larger its disparity. That is, as shown in Figure 2.8, suppose the corresponding image points of $\boldsymbol{X}$ is $\boldsymbol{x} = \begin{pmatrix} u \\ v \end{pmatrix}$ and $\boldsymbol{x}' = \begin{pmatrix} u' \\ v \end{pmatrix}$, we have $v = v'$ and disparity $d = u' - u$. Then the 3D point can be reconstructed as

$$\boldsymbol{X} = \begin{pmatrix} b(u + u')/2d \\ bv/d \\ bf/d \end{pmatrix} \tag{2.4}$$

where b is the length of the baseline between the two cameras and f the camera's focal length. The matching result is a ***disparity map*** which encodes per-pixel disparity information for the left or right image.

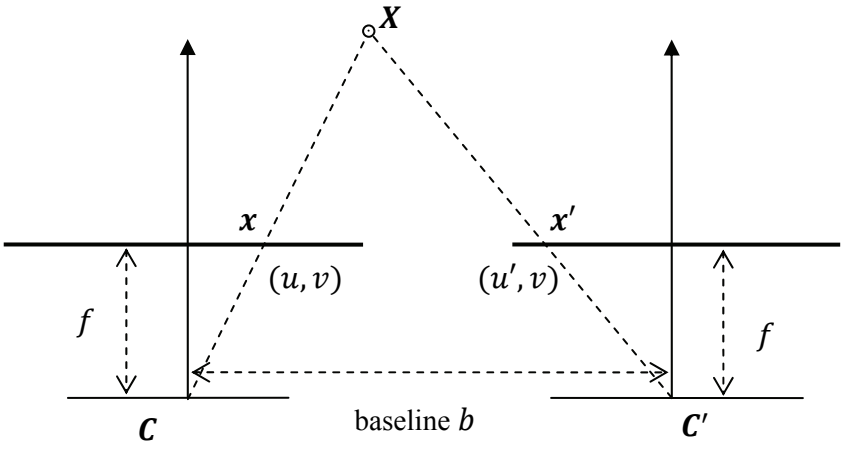


Figure 2.8: Illustration of binocular stereo.

2.3.2 Multi-ocular Stereo

Different from the classic binocular stereo setup, in many scenarios such as in our FVV application, using more cameras could provide more information about the scene and lessen problems due to occlusions. Based on the camera arrangement, multi-ocular stereo can be classified into two types: grid-positioned and general positioned.

In the former case, the cameras having a same focal length are arranged on a grid structure such that the baselines (spans) between cameras in the horizontal and vertical directions are all equal. Such special arrangement enables using disparity to encode depth information as in classic binocular stereo so that many binocular stereo methods can be directly used. The use of more images (usually the 4-connected neighbors) results in more accurate depth evaluations and better occlusion handling.

While in the latter case, the cameras are in general positions. Since there is no simple disparity-to-depth mapping any more, the true 3D depth information has to be recovered instead. The above simple displacement based inter-image mapping is replaced by a depth-based 3D forward warping procedure as introduced in Section 2.2.3. Specifically, as show in Figure 2.9, the 3D depth space is discretized into different depth planes. To check how well a matching primitive is matched between two images w.r.t a specific depth, its corresponding 3D primitive is found by intersecting the back-projected line or cone with the corresponding depth plane first and then re-projecting the intersection point to the corresponding cameras. In this way, most of the above binocular stereo matching techniques can be extended to multi-ocular stereo [82].

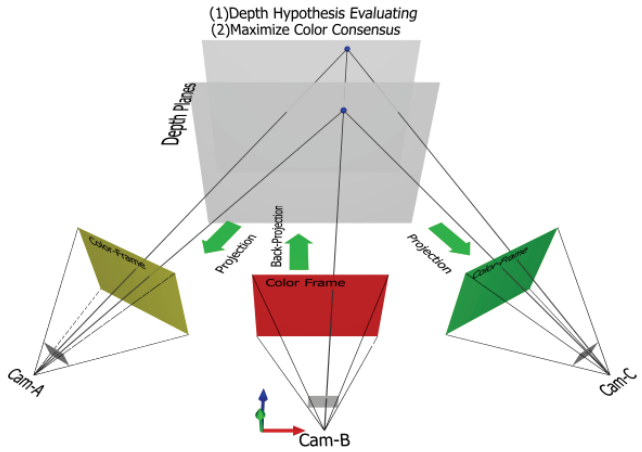


Figure 2.9: Illustration of multi-ocular stereo.

2.3.3 Temporal Stereo

In particular to FVV applications, in addition to improving the spatial accuracy of depth maps of different views, the temporal consistency between depth maps of consecutive video frames is also very important for better rendering quality. To this end, many temporal stereo techniques integrating motion analysis have been proposed.

Most of the representative temporal stereo methods [83-86] formulate the temporal stereo problem as a combination of static stereo matching and conventional monocular video sequence optical flow estimation (or motion analysis). For example, in 1986, Waxman and Duncan [83] proposed to fuse binocular stereo and motion estimations at a very early level by observing that a proper combination of them could help each other to overcome each other's inherent difficulties. Nasrabadi et al. [84] propose a cooperative method by modeling the input data as coupled Markov random fields (MRF's). The stereo problem is generalized so that not only pixel intensity but also its optical flow information is matched. As well, the Markov random fields (MRF) are used to combine stereo and motion estimations so that depth or motion discontinuities can help predict the other and the preservation of stereo matching is enforced through motion analysis results [85]. Similarly in the work of Isard and MacCormick [86], a single MRF probabilistic framework integrated with a multi-frame temporal

filtering mechanism is proposed. Motion and depth are estimated simultaneously with explicit handling for occlusions, depth discontinuities and motion discontinuities, which results in better performance than estimating either in isolation.

However, the performances of all the above attempts in integrating motion analysis with stereo matching have not been extensively verified with challenging real-world FVV data. Moreover, with the recent impressive progress in both motion analysis and stereo matching research, new integration schemes based on novel mathematical models and optimization frameworks should be investigated.

2.4 Discussions

Based on above review on FVV related works, it can be seen that building a practical FVV system requires many technical and scientific issues to be addressed. In the next chapter, we first deal with the hardware and software architecture design of our depth based FVV system. Then more detailed descriptions of the related algorithms are given in the remaining chapters.

Chapter 3
Design of a Computer Cluster Based FVV System

One of the main goals of this thesis is to develop a prototype FVV system. The system is intended to be self-contained so that it can undertake all the indispensible workflow processes ranging from acquiring the raw input videos to presenting the final FVV rendering with as little as possible human interventions. In this chapter, a scalable and extensible cluster based FVV system design is presented. Although the system is mainly intended for FVV applications, it is designed more as a general framework so that it can also be used for developing other cluster based camera array applications.

3.1 Previous Work in FVV System Design

Most existing FVV systems use their own in-house frameworks. Such frameworks are usually not developed for a general purpose FVV application and have strong coupling with the hardware configurations or the application contexts. Moreover, most of the published literatures do not elaborate their implementation details, which render it very difficult to reproduce or extend the corresponding FVV systems.

In the following some representative FVV system designs are first briefly reviewed. Then by comparing and trading-off the corresponding advantages and disadvantages, the system design specifications suitable for this thesis work are defined.

3.1.1 CMU's Virtualized Reality System

The Virtualized Reality™ system [12] built by CMU is arguably regarded as the first FVV system. Up to 49 widely spaced cameras arranged in a dome or a cube environment are used to create FVV using object based scene representation. Camera synchronization is done using external sync signal and time code

generators. Although a large scale cluster is used to stream all the video data in real-time, the bandwidth bottleneck still limits the system to only streaming very short video sequences. The calibration of cameras is done using dot patterns on the in-scene floor and the FVV scene reconstruction is done offline using multi-view stereo and silhouette information.

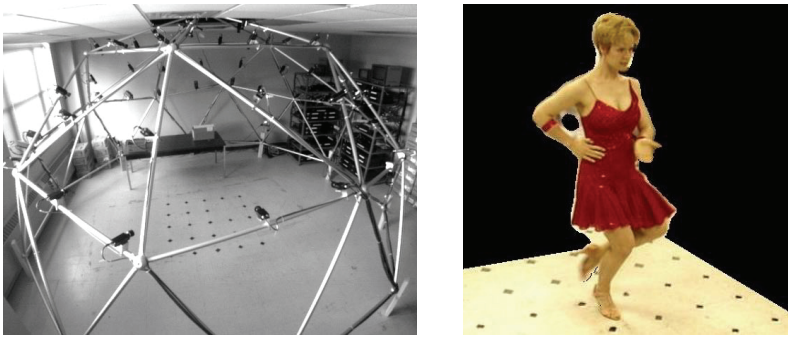


Figure 3.1: The "Virtualized Reality" System.

3.1.2 Surrey's FVV System for Human Actor

In the work of Starck et al. [87], an interactive FVV system is presented which uses up to 10 cameras arranged in a circle. The public domain calibration toolbox [32] is used for geometric calibration of the camera array. Background subtraction and view-dependent visual hull approaches are used to generate the FVV content offline and OpenGL is used for real-time rendering. Since its system design and underlying techniques have strong reliance on the studio environment for clean background, the portability and scene applicability are relatively limited.

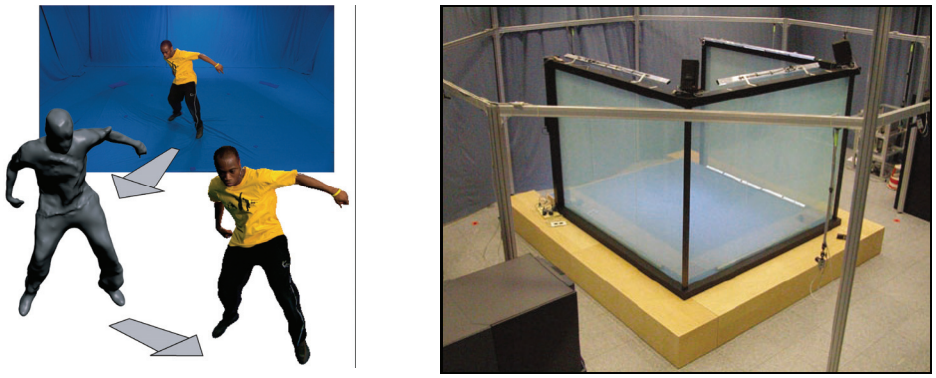


Figure 3.2: Surrey's FVV System (left) and the Blue-C system (right).

3.1.3 Blue-C System

The Blue-C system [14] is a real-time FVV portal specifically devised for telepresence application. It consists of a room-sized environment with a real-time 16-camera capture system and spatially immersive displays. Complex synchronizations between cameras, active display panels, illumination lights and user shutter glasses are precisely managed using specialized hardware for video capturing, user silhouette/texture extraction and 3D video display. Control information and non-real-time data are transmitted using traditional remote method invocations, while for the real-time data, traditional transfer protocols such as UDP or RTP are used. The visual hull based approach is used to reconstruct the FVV scene using point based representation. Parallel computation is done on a PC cluster for real-time performance. In its disclosed framework API [88], basic components such as distributed scene graph, synchronization manager, node manager, and built-in services such as graphics/audio rendering and networking are provided for building other remote collaboration and presentation applications. However, the API does not include FVV specific components or services, which render it not very suitable for FVV application development.

3.1.4 Stanford Camera Array

The Stanford camera array [16] intends to provide a general purpose and low cost camera array system. By using specially designed cameras instead of off-the-shelf ones, only 4 PCs are needed to stream 100 MPEG2 compressed videos. By chaining up all the cameras with external triggering lines and using the FPGAs and dedicated clock, precise timing control with arbitrary phase shifts among camera triggers can be achieved. This system enables many new camera array applications such as high speed videography [16]. Its main limitation, however, is in its reliance on special cameras and the required high-rate video compression may degrade the FVV rendering quality.

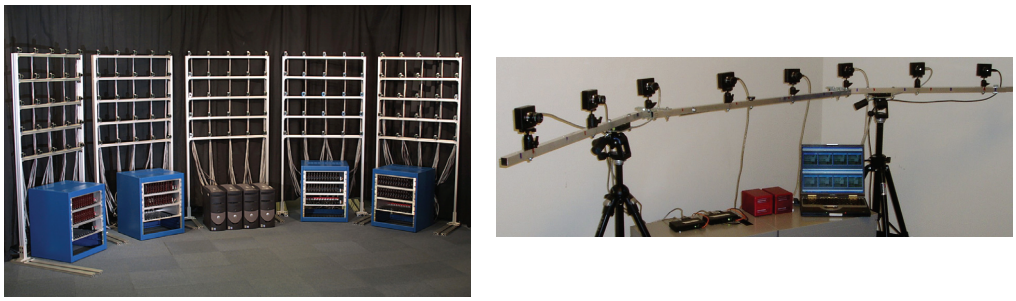


Figure 3.3: The Stanford's camera array system (left) and The MS's FVV system.

3.1.5 Microsoft FVV System

As the last representative FVV system, Microsoft Research has developed an 8-camera array system [15]. The cameras can be arranged on a 1D arc horizontally or vertically for FVV data acquisition. Geometric calibration is done using Zhang's method [28]. No explicit photometric calibration is done. To handle real-time storage of all the uncompressed input videos, the camera manufacturer (Point Grey Research) was commissioned to build specialized concentrator units which synchronize the cameras and store the video data directly to a bank of hard disks connected to the cameras by fiber optic cables. The concentrators themselves are further synchronized via a Firewire cable. In this way, the video acquisition can be managed by using only one laptop, greatly increasing the system portability. However, the system is closely coupled to the specific hardware devices, making it difficult to scale up or to reproduce.

As can be seen from all the above reviewed FVV systems, there are mainly two design strategies. The first one is to separate multiple video acquisition from other workflow stages. Its advantage is that the hardware system can be made very portable by using a minimal number of controlling PCs and specialized cameras. Its main limitation is that all the FVV-specific computations have to be done offline, which renders real-time FVV applications impossible. On the other hand, the second strategy is to integrate video acquisition with other workflow stages together by further providing powerful computation capability through a cluster of computers. Its advantage is that implementing computation-intensive

real-time FVV applications is supported and the management of the whole application workflow can be greatly simplified. Its main disadvantage is that the system could be bulky due to the use of the cluster, especially when the number of required cameras is large.

3.2 Design Specifications

In this thesis, we design our FVV system using the above cluster based strategy. That is, the cameras are managed by a cluster of networked PC nodes. Each node not only controls a subset of the cameras for multi-view video acquisition but also participates and collaborates in the application-logic computations. This design decision is made based on the following considerations.

First, as the FVV applications using dense depth based scene representation is the focus of this thesis, the number of cameras used is relatively small compared to applications using ray based scene representation.

Secondly, to satisfy long-term research goals, using cluster based design renders the system more versatile and flexible for satisfying different FVV application hardware requirements in future research.

As for the software architecture, in this thesis, our FVV system is designed more as a general framework so that the number of constraints on applicable algorithms or workflow patterns is as few as possible. In addition to our interested FVV authoring and rendering application, it is also intended to be as capable as possible for prototyping and developing other camera array based algorithms and applications.

In the following, we summarize the specifications that such a system or framework should fulfill with higher priorities.

(1) Centralized Workflow Management

The system should enable centralized managements of most FVV workflow stages. It should be able to perform all the required configurations and managements for different workflow stages on a single node. Otherwise to set up individual node or camera could be tedious and error-prone.

(2) Distributed data management and parallel computation

To best take advantage of the cluster's capacity, the system should manage the data in a distributive fashion. Also all the parallelizable computations involved in the application logic should be distributed to the nodes that control the involved cameras, resulting in camera-level parallelism.

(3) Transparent and low latency data access

With the data/computation distributed, the system has to enable location transparency of data accesses. That is, each node should be able to transparently access any other local or remote cameras for their video data or other relevant information when needed. Moreover, the access latency of remote data should be as low as possible. This is crucial for real-time services such as FVV rendering.

(4) Workflow modularization and customization

The system should be modularized. Application-specific algorithms are implemented as services. Some basic services commonly required in different camera array applications such as video acquisition, calibration should be built in and the developer can customize the workflow and integrate application-specific services.

(5) Hardware independency

The system should only use off-the-shelf devices and be independent of specific hardware configurations as much as possible. Different types of cameras should be supported directly without the need of other hardware or software changes. Camera synchronization should not reply on specialized hardware. And the number of cameras or computer nodes should be scalable for different application needs.

3.3 Hardware Architecture Design

Based on the specifications summarized above, the hardware architecture of our FVV framework designed is shown in Figure 3.4. In particular, a cluster with 8 nodes is used. The cluster is mounted on a mobile cart that can be moved around.

Each node is equipped with two AMD dual-core Opteron CPUs, two Nvidia 8800 GTX GPUs. The high-end CPUs and GPUs provide exceptional computing power for general purpose and graphics intensive computations. Each node is also equipped with two network interface cards (NIC). One is an on-board 1Gb Ethernet interface and the other a 10Gb high-speed Ethernet NIC from Myricom [89], which are interconnected by the corresponding 1Gb and 10Gb switches, respectively. As to be elaborated later, the 1G star topological network is mainly for master-worker based non-real-time data communications. The high speed 10G mesh topological network is specific for point-to-point real-time data communication, which is crucial for low latency remote data access.

Firewire-b cameras are used for multi-view video capturing due to their high data throughout. Each node in the cluster is also equipped with a 3-port Firewire-b interface card so that up to three cameras can be handled directly by each node. To scale up the camera array, Firewire-b hubs or extra nodes can be used. However, due to the hard-disk IO bandwidth limit, the camera's resolution or frame-rate may have to be lowered appropriately. Currently, we use Flea 2 Firewire-b color cameras manufactured by Point Grey Research, each of which can operate with a maximum resolution of 1024x768 and maximum frame-rate of 15 FPS. Off-the-shelf 8 mm lenses are used for less lens distortion. Moreover, all the cameras are mounted on a mobile frame to ease transportation. By reconfiguring the frame's structure, different 1D or 2D camera arrangements can be achieved easily.

For centralized management, the cluster works in the master/worker mode. One node is selected as the master one and all the other nodes work as workers commanded by the master node. Each node controls the video acquisition, stores and processes the captured video data locally, enabling distributed data management and parallel computation. The high speed networking makes it possible to implement transparent and low latency data access. The system also supports other types of cameras such as camcorders which can work in the integration mode or standalone mode.

Figure 3.4: Our FVV System hardware setup.

3.4 Software Architecture Design

As mentioned before, our FVV system is developed as a specific implementation of a general camera array application framework. The software architecture of our framework is designed using the object-oriented and event-driven paradigms and the service provider/consumer model. It includes a number of foundation components and services, which can be used for quick development of a prototype FVV or camera array application. The developer only needs to focus on the development of application specific components and services.

3.4.1 Foundation Components

The following loosely-coupled foundation components are implemented to facilitate development of different FVV or camera array applications. Shown in Figure 3.5 is an illustration of the software component architecture of our framework.

(1) Node

The node component abstracts the functionalities of a cluster node. Each node component is tagged with its 1Gb network IP address as a unique ID and can work in either worker or master mode.

Each node component hosts a message communicator and a data communicator for inter-node message and data communications respectively. In particular, via the message communicators, the worker nodes are connected in a star topology network with the master node serving as the hub. This is because

most message communications are related to camera manipulation and workflow configuration and they mainly occur between the master node and the worker nodes. For message communications between two worker nodes, which mainly occur for establishing their direct point-to-point data communication connections, the master node will act as a message router.

In addition to message communication network, the nodes are connected into a mesh topology network via the data communicators. In this way, direct data communication is made possible between any two nodes.

The node component also hosts a camera manager component, which for a worker node, manages all of its local cameras, and for the master node, registers and manages all the cameras in the cluster, with a proxy camera created to represent each remote one. Anytime a worker node is connected or disconnected, or anytime a camera is attached or detached, the camera registry will be updated automatically through the corresponding message communications.

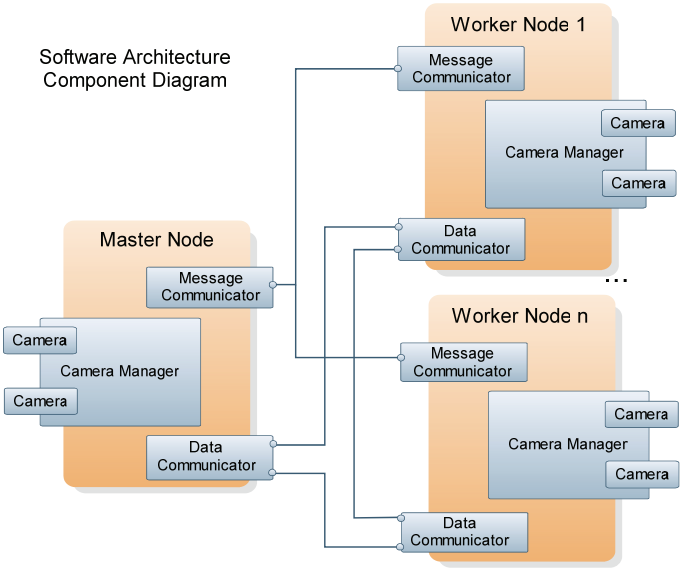


Figure 3.5: Software architecture component diagram.

(2) Communicator

As mentioned above, the responsibility of the communicator component is to perform bi-directional communications between different nodes. Based on the hardware configuration, two types of communicators are implemented: TCP/IP protocol based message communicator and MXoE protocol (proprietary to

Myricom) based data communicator. The former one is used for passing camera/node event messages between master and worker nodes, whose packet sizes are usually small and fixed. The latter one is used for real-time transfer of large packets of data, such as images or various computation results, between any two nodes directly. Using independent networks for message and data communications can better adapt to their different flow patterns and performance requirements. This design also allows convenient extension to use future high-speed networks.

(3) Camera manager

The camera manager component is provided mainly for centralized and layered management of cameras on each node. In addition to its management routines of inserting/removing cameras, it also has the responsibility to relay messages from the hosting node to the destination camera(s).

(4) Camera

The camera component is one of the key components of our framework. It generally abstracts any objects being able to provide a sequence of temporally sorted images. In our framework, we define three categories of cameras: online, offline and proxy.

An online camera corresponds to a physical video camera capable of streaming images sequences. By using the Microsoft DirectShow framework, a wide range of video cameras are supported. However, our framework also provides sub-classed component for cameras using non-DirectShow drivers such as cameras from Point Grey Research [90] for more control options and better performance.

An offline camera abstracts a pre-recorded video file, which is useful in our current FVV application needing to perform offline processing on video files.

Finally a proxy camera is mainly used by the master node as a proxy of a remote camera connected to a worker node. It can relay video frames from the remote camera in full frame-rate through the high-speed network, making accessing remote video data by the master node indistinguishable to that from

accessing a local one. Also it will automatically synchronize itself with its remote peer, making it possible to achieve transparency in both camera property/data query and in the occasion to invoke camera related services.

The camera component has various properties such as serial number, hosting node ID, resolution, frame-rate, intrinsic and extrinsic parameters. It provides basic playback control to start, play, pause, stop, record, seek, rewind or forward a video. It also provides some local services such as calibration pattern tracking, intrinsic parameter calibration, geometric un-distortion, color correction and so on. All of these services can be invoked remotely and asynchronously using the corresponding event messages.

3.4.2 Messaging and Parallelism

As mentioned before, the message passing scheme is used in our framework to organize and to coordinate all the components for efficiently fulfilling the application-specific tasks. The underlying principle of our implementation is similar to that of the Message Passing Interface (MPI) protocol [91], but in a lightweight way tailored for better performance and OS deployment requirements.

Specifically, a message is packed into a packet together with the information of the message source/destination node ID, the source/destination camera's serial number, the message ID and a length variable message buffer. The message packets are sent or received through the inter-node socket [92] connections. Each node spawns a thread in which the message communicator continuously dispatches out or receives in messages to or from another node. By parsing the message ID information in the received packet, the destination node may further route the message to the destination camera(s) via the camera manager, for handling the corresponding event. Similarly, each node also spawns a pumping thread in which its data communicator communicates bi-directionally with all the other active nodes to receive or send data through respective channels. With the use of the 10Gb network, the remote data access latency is very similar to the local data access latency.

Moreover, since many services involved in the camera array or FVV applications are naturally data-parallelizable at the camera level, e.g. calibrating the intrinsic parameters of the cameras can be done independently in parallel, our framrwork has built-in support for such parallelism. For this, multiple threading techniques are heavily leveraged.

For example, to invoke the intrinsic calibration service on a remote camera, a unique task ID is generated and sent together with other necessary configuration information as a message to a remote camera. Simultaneously, the corresponding proxy camera will be informed to spawn a result-waiting thread in which the task ID is used to check if the corresponding service computation has been done remotely. While for the remote camera, upon receiving the corresponding message, it creates a calibration service execution thread to track feature points of the calibration pattern and then to do the calibration. When the calibration is done, the calibrated intrinsic parameters will be sent back through the data communicator with the task ID tagged as the destination identifier. By keep probing the data with the task ID as a filter, the waiting proxy camera finally receives the updated intrinsic parameters. It then notifies the service requesting thread through inter-thread communication mechanism to indicate that the service request has been fulfilled and that the data is ready to use.

In this way, multiple services can be executed on multiple cameras simultaneously, which greatly improves the performance.

3.4.3 Service Customization and Extension

To satisfy different FVV application requirements, our framework also supports service customization and extension.

When developing a new FVV application, a skeleton project is first created which the node/camera managements and communications are enabled automatically. The developer only needs to focus on the application logic implementation by using/customizing the built-in utility and computation services or integrating new services for fulfilling the corresponding workflow. Such

customization or extension is done through an application logic callback mechanism.

Specifically, the application can register callback functions on the framework's event-handler components such as Node or Camera, which will be called first when an event registered is to be handled. For customization, the application logic can augment or override the built-in event handler code with its own handler. For extension, the application registers corresponding application domain events and handles them in the corresponding callback functions. Please note that the same event may need to be handled differently on the master and the worker nodes. In this way, the framework components can be reused and the application logic can be modularized and loosely coupled to the framework.

3.5 Implementation

Based on the above designed framework, our FVV system has been implemented in C++ on Windows XP Professional 64® OS.

The system workflow is highly automatic and needs very little user intervention. In particular, there are 5 steps for creating FVV content.

(a) System Initialization

The hardware system initialization mainly involves setting up the cluster and camera frame properly on the FVV capturing site. For software system initialization, the same application program starts up on each node with different options to run either as the master or as the worker. The worker starts to run as an OS service in GUI-less mode automatically upon booting up the operating system and keeps listening to master, while the master is manually started by the user as a normal GUI application. After the master starts up, the workers automatically register themselves to the master and establish the bi-directional message and data communication connections. Then all the available cameras are probed and registered. Currently our system can handle 16 cameras. One example on-site setup and the corresponding system GUI are shown in Figure 3.6 and 3.7, respectively.

(b) Video Acquisition

The master node manages the video acquisition process using the GUI, by which the user can select the online cameras to capture videos. The actual multiple view video acquisition is triggered by sending the video recording event to all the selected cameras. Currently we require each video to also capture a part of movements of our customized plane pattern in the scene for calibration purpose.

(c) Calibration

In this stage, our system executes different built-in services to geometrically, photometrically and temporally calibrate the camera array through the captured calibration pattern in the videos. The details of the algorithms implemented in the calibration services are elaborated in Chapter 5.

(d) FVV Scene Reconstruction

After the calibration is done, depth based FVV scene reconstruction is performed in this step. Simply speaking, the system notifies each involved offline camera (video file) to spawn a stereo matching service execution thread in which all the synchronized video frame sets are collected and matched in turn with the corresponding camera as the reference. During the process, each camera also needs to communicate with other ones for accessing their depth maps or other intermediate results. The details of the specific algorithm implemented in the service are explained in Chapter 9.

(e) FVV rendering

After the FVV scene is reconstructed as multiple depth sequences, the user can invoke the built-in FVV rendering service to view the created FVV content. The depth-based warping method is used to synthesize novel views for viewpoint transition. The involved computations are done on the GPUs in real-time. For each synthesized view, the required corresponding video frames and depth maps could be fetched remotely from different nodes. High-speed data communication enables real-time rendering. More details about the FVV rendering algorithm are given in Chapter 9. Shown in Figure 3.8 is one example synthesized view and the corresponding depth map.

With the centralized workflow management, automatic calibration and distributed computation, our FVV system makes the whole process from capturing multiple view videos to the final FVV rendering very convenient and efficient, which provides a practical solution for FVV content authoring and rendering. On the other hand, because the system is highly modularized, different algorithms can be easily tested and integrated. Therefore, our system can also be used as a general platform for different camera array applications.

Figure 3.6: One example on-site setup of our FVV hardware system.

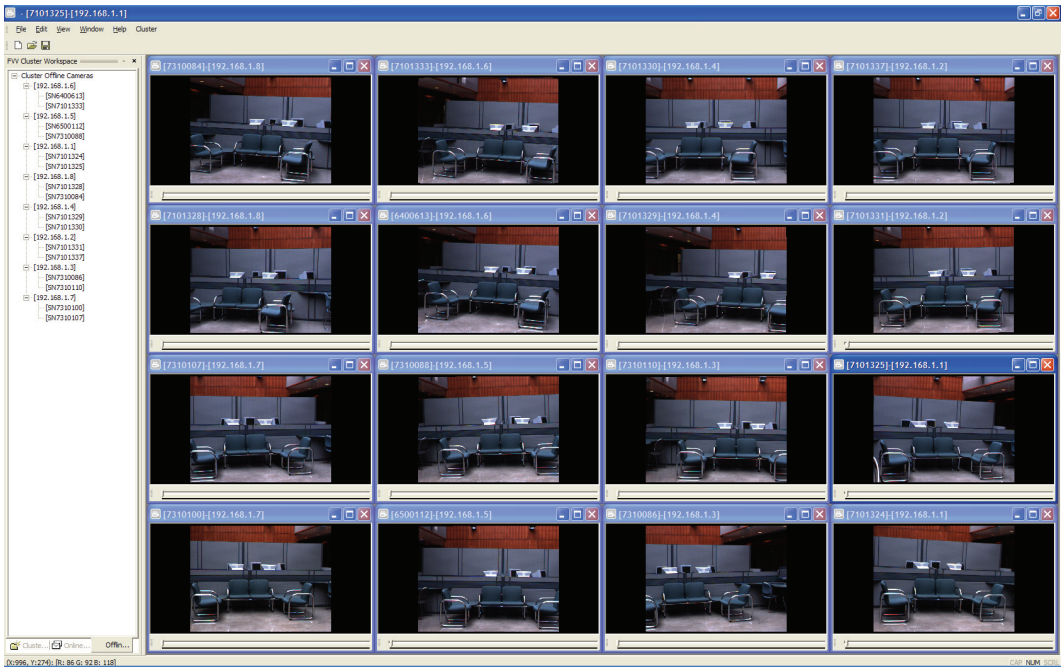
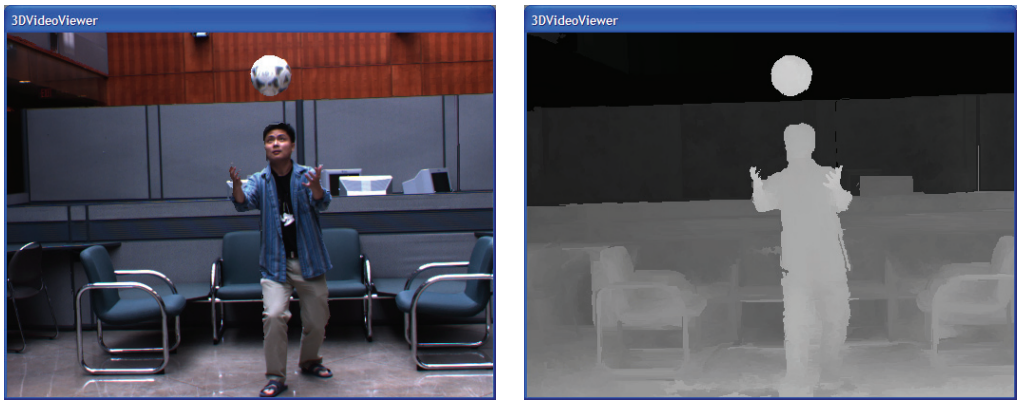


Figure 3.6: Example screenshot of the main GUI of our FVV software system.



(a) One synthesized view (b) Corresponding depth map (synthesized)

Figure 3.7: Example FVV rendering using our system.

Summary

In this chapter, the hardware and software designs and implementation of our cluster based FVV system are presented. The system is designed not only for specific research needs of this thesis, but also as a general framework for other camera array applications. By taking advantage of the built-in utility and computation services, new applications can be easily prototyped and tested. In the following chapters, details of the underlying algorithms implemented in our FVV system are elaborated.

Chapter 4
Tri-Focal Tensor based Multiple Video Synchronization

In this chapter, a novel method for synchronizing multiple (more than 2) un-calibrated video sequences recording the same event by free-moving full-perspective cameras is presented. Although this method is not directly used in the current FVV system, it is a general method to temporally correlate data from multiple simultaneously recorded video sequences, which may be needed in many offline camera array applications. Furthermore, since the equivalent underlying synchronization mechanism is used in the camera array total calibration method to be elaborated in the next chapter, this method is included for completeness.

4.1 Previous Work

Our proposed multiple video synchronization method is a software based one. Based on the inter-video temporal correlation constraint used, most software based video synchronization methods can be roughly categorized as: feature based [93-100], intensity based [101] and camera movement based [102].

Specifically, the feature based methods [93-100] usually require tracking features in videos and implicitly or explicitly matching such features across the sequences. The most commonly used features are the points, which can be treated locally as single points or globally as trajectory curves. Such feature based methods are usually based on the fact that, for the exactly synchronized frames, the corresponding dynamic 3D scene features can be regarded as a stationary rigid configuration at the corresponding time instant. Therefore, some form of multiple view geometric alignment constraints must hold for their 2D projections in the synchronized frames. Commonly exploited geometric alignment constraints include the binocular epipolar geometry constraint [94-97], the plane-induced

homography [93, 95], rankness properties arose from special projection models [98], feature movements [99] and so on.

The intensity based methods try to minimize the sum of squared differences (SSD) between the sequences that can be spatially and temporally warped through a parametric model. As described in the representative work [101], the homography based spatial transform and the 1D affine temporal transform are usually used. All the pixels can provide constraints to such a model, while not just a limited number of salient features so that feature tracking and matching can be avoided. Moreover, the temporal and spatial information in the sequences can be utilized in the unified framework for simultaneous spatial and temporal alignments.

As an example of camera movement based synchronization methods, in [102], the sequences with no overlap in their visual field of views can be aligned spatially and temporally under the assumption that the cameras are fixed rigidly sharing a common optical center and moved together. The synchronization is done by using the homography induced constraint between the frame-to-frame transformations across the sequences.

Effort has been made for handling multiple free moving cameras. In [99], a 5-point method is proposed for synchronizing 2 video sequences captured by affine moving cameras in 3D instead of in 2D by evaluating the line-to-line distance of the back-projection lines of the matching points as the geometric alignment measure. While in [94], with fixed inter-camera epipolar geometry recovered using stationary feature points, for each moving feature point detected in the reference sequence, if the corresponding epipolar line intersects with any tracked feature trajectory across consecutive frames in the other sequences, a timeline map voter voting for the corresponding tentative synchronization offset is formed. With enough voters collected, a RANSAC procedure is applied to synchronize more than 2 sequences with the robust feature matching done implicitly.

Different from most previous work, our proposed method is for synchronizing more than 2 video sequences. As a feature based one, instead of using the

traditional bi-view geometric constraint, trifocal tensor based tri-view geometric constraint is used for its better robustness and less ambiguity in identifying cross-view correspondences. Moreover, similar to [94], our method is voting based. That is, for each synchronization relation hypothesis between video sequences, a collection of putatively synchronized video frame sets as voters make vote based on the corresponding geometry alignment measure. However, instead of fitting the optimal synchronization in the voting space as done in [94], our method further resorts to a final non-linear optimization to achieve sub-frame accuracy, with the voting results used as its initial values.

4.2 Problem Formulation

Synchronization of a set of $M(\geq 3)$ video sequences $\widehat{\boldsymbol{S}} = \{\boldsymbol{S}_m | m = 0, \dots, M-1\}$ can be formulated as a timeline mapping recovery problem and solved through optimization. The video capturing cameras do not have to be stationary and free moving cameras can also be handled by this method. Specifically, each video sequence $\boldsymbol{S}_m$ defines a local timeline TL_m sampling the captured dynamic event spatially and temporally with a rate of Fps_m (i.e. frame rate), with each sample being a captured video frame $\boldsymbol{I}_m^t (t \in TL_m)$. W.l.o.g., taking $\boldsymbol{S}_0$ as the reference sequence, the synchronization problem can be stated as: given a timeline sample $\boldsymbol{I}_0^{t_0}(t_0 \in TL_0)$ in $\boldsymbol{S}_0$, find in the other sequence $\{\boldsymbol{S}_m | m = 1, \dots, M-1\}$ the corresponding timeline sample $\boldsymbol{I}_m^{t_m}(t_m \in TL_m)$ that is captured at the exact same time instant, i.e., to recover a one-to-one timeline map $\mathbb{M}_{TL_0 \to TL_m}$ from $\boldsymbol{S}_0$ to $\boldsymbol{S}_m (m = 1, \dots, M-1)$ respectively.

Various forms of the timeline map can be used. The simplest one is the "offset-only" form given as $\mathbb{M}_{TL_0 \to TL_m}: t_0 + \Delta_m = t_m$ when $Fps_0 = Fps_m$. The more general "1D-Affine" form defined as:

$$\mathbb{M}_{TL_0 \to TL_m}: \frac{Fps_m}{Fps_0} t_0 + \Delta_m = \alpha_m t_0 + \Delta_m = t_m \tag{4.1}$$

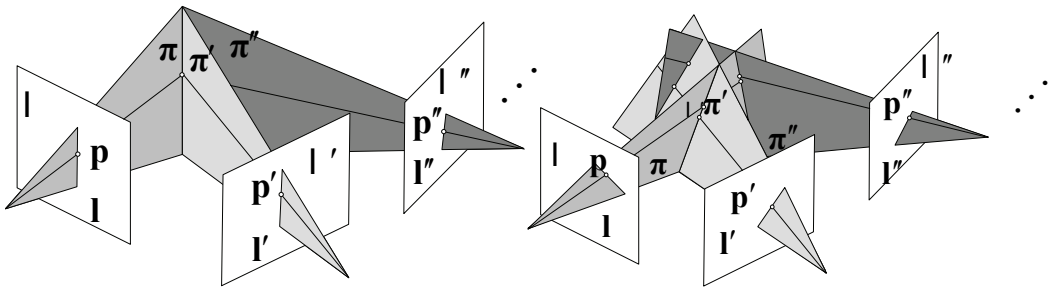
can be used when $Fps_0 \neq Fps_m$. However, if the frame rates are constant and known, it only needs to recover the offset Δ_m. In some special cases, a dynamic timeline map is required [103], which is beyond the scope of this thesis.

Given a specific group of hypothesized synchronization timeline maps $\{\mathbb{M}_{TL_0 \to TL_m}\}$, each set of video frames that satisfies such mapping relations is called a "supporting voter," which votes on how well the inter-sequence synchronization is achieved according to $\{\mathbb{M}_{TL_0 \to TL_m}\}$. In the following, we denote such a "voter set" as $\boldsymbol{\mathcal{V}}$.

4.3 Overview

The new proposed video synchronization method is a feature based method. That is, the video synchronization problem is addressed as a geometric alignment problem for the set of matching features among multiple video sequences. How well a set of video frames are synchronized is evaluated using a geometric alignment metric measure of the matched features.

Instead of using the binocular epipolar constraint of point feature as in many previous methods, the proposed method uses the multiple-view geometric incidence constraint of point and line features. As shown in Figure 4.1, for the matched 2D point/line features on exactly synchronized frames, their corresponding back-projected lines/planes must intersect at/in a single 3D point/line, i.e., forming a sheaf (or pencil) of the back-projected lines/planes. Otherwise, no such line/plane sheaf can be formed in general. Therefore, how the back-projected lines/planes are aligned at/in a single 3D point/line can be used as the metric measure for evaluating how well the video frames are synchronized.



(a) Synchronized (b) Unsynchronized

Figure 4.1: Illustration of the difference between synchronized and unsynchronized cases when 2D points/lines are back projected.

The proposed method works as follows.

Firstly, the point/line features are matched across sequences automatically or manually and then tracked automatically within each sequence. It is assumed that the features can be tracked throughout the whole sequence for simplicity, i.e., there is no missing data.

Secondly, all the possible integral synchronization offset combinations are evaluated to find the best one which maximizes the feature geometric alignment measure w.r.t all available support voters (*M*-frame tuples).

Specifically, w.r.t. the reference sequence $\boldsymbol{S}_0$, the verifiable integral synchronization offset Δ_m corresponding to sequence $\boldsymbol{S}_m$ is in the range of $\phi_m = [-T_0 + 1, T_m - 1]$. That is, it includes the two extreme cases of potential synchronization between $\boldsymbol{S}_0$ and $\boldsymbol{S}_m$ when the first frame of $\boldsymbol{S}_0$ is synchronized with the last frame of $\boldsymbol{S}_m$ or when the last frame of $\boldsymbol{S}_0$ is synchronized with the first frame of $\boldsymbol{S}_m$.

For each integral offset combination candidate$(\Delta_1, \cdots, \Delta_k, \cdots \Delta_{M-1})$, the support voters $\boldsymbol{\mathcal{V}}_{(\Delta_1,\cdots,\Delta_k,\cdots\Delta_m)}$ satisfying the timeline maps defined by $(\Delta_1, \cdots, \Delta_k, \cdots \Delta_{M-1})$ is collected so that each voter $\Gamma \in \boldsymbol{\mathcal{V}}_{(\Delta_1,\cdots,\Delta_k,\cdots\Delta_m)} = \{\boldsymbol{I}_0^t, \cdots, \boldsymbol{I}_{M-1}^{t+\Delta_{M-1}}\}$ with $t \in \boldsymbol{\mathcal{K}}_{\boldsymbol{\mathcal{V}}_{(\Delta_1,\cdots,\Delta_k,\cdots\Delta_{M-1})}} = \{i | 0 \le i < T_0, 0 \le i+\Delta_m < T_0, m = 1, \dots, M-1\}$.

Then all of the support voters are checked with the corresponding feature geometric alignment measure evaluated and stored in an *M*-dimensional evaluation tensor $\boldsymbol{E}$ as $\boldsymbol{E}(t, \alpha_1 t+\Delta_1, \cdots, \alpha_{M-1} t+\Delta_{M-1})$ with $t \in \boldsymbol{\mathcal{K}}_{\boldsymbol{\mathcal{V}}_{(\Delta_1,\cdots,\Delta_k,\cdots\Delta_{M-1})}}$, whose $Median_{t\in\boldsymbol{\mathcal{K}}_{\boldsymbol{\mathcal{V}}_{(\Delta_1,\cdots,\Delta_k,\cdots\Delta_{M-1})}}}\big(\boldsymbol{E}(t, \alpha_1 t+\Delta_1, \cdots, \alpha_{M-1} t+\Delta_{M-1})\big)$ in turn is used as the overall synchronization fitness evaluation of $(\Delta_1, \cdots, \Delta_k, \cdots \Delta_{M-1})$. Here the median value is used instead of the mean value for better robustness. Note that each entry in $\boldsymbol{E}$ is initialized to zero.

Therefore, after checking all possible integral synchronization offset combinations, the best integral synchronization offsets $(\tilde{\Delta}_1, \cdots \tilde{\Delta}_k, \cdots \tilde{\Delta}_{M-1})$ can be recovered such that the synchronization fitness evaluation is maximized, that is, $(\tilde{\Delta}_1, \cdots \tilde{\Delta}_k, \cdots \tilde{\Delta}_{M-1}) =$

$$\text{argmax}_{(\Delta_1,\cdots,\Delta_k,\cdots\Delta_m)}\left(Median_{t\in\mathcal{K}_{\boldsymbol{v}_{(\Delta_1,\cdots,\Delta_k,\cdots\Delta_{M-1})}}}(\boldsymbol{E}(t,\alpha_1 t+\Delta_1,\cdots,\alpha_{M-1}t+\Delta_{M-1}))\right) \quad (4.2)$$

Finally, by using the recovered integral synchronization offsets $(\tilde{\Delta}_1, \cdots \tilde{\Delta}_k, \cdots \tilde{\Delta}_{M-1})$ as initial values, a post-optimization using the Levenberg-Marquardt (LM) method [81] is performed to further achieve sub-frame synchronization accuracy.

In the following, the implementation details of our method are presented.

4.4 Implementation Details

4.4.1 Issue of Computational Complexity

Synchronization of *M* video sequences with a maximum frame number of *T* using an exhaustive search strategy has computation complexity of (T^M) , which has poor scalability in practical applications. To address this problem, the following strategies are incorporated.

Instead of processing all the sequences at once, the synchronization process starts with three bootstrapping sequences and incrementally synchronizes the remaining ones by adding one or two sequences at a time. That is, after the first three video sequences are synchronized, the remaining ones are synchronized incrementally step by step.

Specifically, in each step, two unsynchronized sequences are grouped with one synchronized sequence as a new group of three sequences which are then synchronized using the tri-focal tensor based method to be discussed in Section 4.4.2. Moreover, by specifying a global reference sequence (usually it is one of the bootstrapping sequences), all the synchronization results can be integrated into a global temporal system, which can be further globally optimized as explained in Section 4.4.3.

As for the synchronization process of each 3-sequence group, it can be further speeded up through a coarse-to-fine hierarchical strategy. In particular, assuming two hierarchical levels are used, the three sequences are first temporally down-sampled appropriately so that the total number of possible integral synchronization offset combinations decreases dramatically. Then with the down-sampled sequences synchronized, three much shorter sub-sequences are each extracted from the original sequences so that their first frames are coarsely synchronized according to the synchronization result obtained and are further synchronized. For long sequences, multiple hierarchies could be used as well.

By using the above strategies, the computation complexity of our method can be lowered to $\boldsymbol{O}(M \cdot T^3)$. Although the complexity is still higher than the traditional pair-wise approaches, which have a complexity of $\boldsymbol{O}(M \cdot T^2)$, the increase in computation cost is usually acceptable in practice, because the use of tri-view based geometry alignment measure instead of the two-view based one greatly improves the synchronization accuracy as can be seen from the experimental results.

4.4.2 Tri-focal Tensor based Geometric Alignment Measure

Similar to epipolar geometry for two views, there is a similar geometric constraint for three views. As the generalization of the fundamental matrix in three views, the trifocal tensor incorporates all the projective geometric relations of three views, which is independent of the scene structure and depends only on the relative motion among the views and their intrinsic parameters.

The trifocal tensor can be calculated in closed form from the projection matrices of the three views. In practice it is estimated from point or line matches across the three views. In particular, as shown in Figure 4.1, suppose a 3D line $\boldsymbol{L}$ is imaged in three views, then the planes back-projected from its 2D projection lines in each view must all intersect in line $\boldsymbol{L}$ in the 3D space. Since in general three arbitrary 3D planes in space do not meet in a single line, this geometric ***incidence*** condition provides a geometric constraint on the corresponding 2D lines in the three views. Similarly for a 3D point, there is also such an incidence

constraint for its projected points in the three views. From such incidence constraints, the tri-focal tensor representation is derived as its algebraic constraint.

One of the most important properties of a tensor is that it can be used to transfer corresponding points or lines in two views to the corresponding points or lines in the third view.

In particular, there are three trifocal tensors for the three given views, each of which uses one of three views as the reference. In matrix notation, the tri-focal tensor corresponding to view $\boldsymbol{C}$ can be represented by a set of three 3×3 matrices $\boldsymbol{T}_i^{jk} = \{\boldsymbol{T}_0, \boldsymbol{T}_1, \boldsymbol{T}_2\}$, with $\{\boldsymbol{T'}_0, \boldsymbol{T'}_1, \boldsymbol{T'}_2\}$ for view $\boldsymbol{C'}$ and $\{\boldsymbol{T"}_1, \boldsymbol{T"}_2, \boldsymbol{T"}_3\}$ for view $\boldsymbol{C"}$. For three corresponding lines $\boldsymbol{l}$, $\boldsymbol{l'}$ and $\boldsymbol{l"}$ in $\boldsymbol{C}$, $\boldsymbol{C'}$ and $\boldsymbol{C"}$, respectively, we have

$$\boldsymbol{l}^T = (\boldsymbol{l'}^T\boldsymbol{T}_0\boldsymbol{l"} \quad \boldsymbol{l'}^T\boldsymbol{T}_1\boldsymbol{l"} \quad \boldsymbol{l'}^T\boldsymbol{T}_2\boldsymbol{l"}) \tag{4.3}$$

Similar constraints hold for $\boldsymbol{l'}$ and $\boldsymbol{l"}$. This means that, with any two of $\boldsymbol{l}$, $\boldsymbol{l'}$ and $\boldsymbol{l"}$ and the corresponding tri-focal tensor known, the remaining one can be written in closed form using the so-called line transfer. Similar point transferring relation exists for three corresponding points in the three views. For more details, please refer to [81].

Using the bootstrapping (first 3) video sequences as an example, to synchronize them, each potential synchronization offset combination is evaluated on how well the induced timeline maps can synchronize the sequences in question by checking all of its supporting frame triplets (support voter). The evaluation of each supporting frame triplet is done individually by measuring the feature geometric alignment based on the point/line incidence relations encapsulated by the tri-focal tensor.

In particular, suppose that we are evaluating the synchronization of a video frame triplet $\Gamma(\boldsymbol{I}, \boldsymbol{I'}, \boldsymbol{I"})$ among which there exist matching points $\{\boldsymbol{p}_i^k | k = 0, \cdots n_p - 1, i = 0, \cdots 2\}$ and line features $\{\boldsymbol{l}_i^k | k = 0, \cdots n_l - 1, i = 0, \cdots 2\}$, where $n_p(n_l)$ is the number of matching point (line) features. However, for synchronization, the feature points and lines must not all be stationary or move rigidly.

Then from the matching feature points or lines, the tri-focal tensor $\boldsymbol{T}_i^{jk}$ for the 1st view can be calculated to encoding the tri-view geometry between video frames $(\boldsymbol{I}, \boldsymbol{I}', \boldsymbol{I}")$. Then given $\boldsymbol{T}_i^{jk}$ and the matched lines in the 2nd view $\boldsymbol{l}' = (l'_0, l'_1, l'_2)^T$ and 3rd view $\boldsymbol{l}" = (l"_0, l"_1, l"_2)^T$, the corresponding line in the first view $\boldsymbol{l} = (l_0, l_1, l_2)^T$ can be obtained by transferring from $\boldsymbol{l}'$ and $\boldsymbol{l}"$ through the tensor operation of $l_i = l'_j l"_k \boldsymbol{T}_i^{jk}$. Similarly, the above property also holds for the line transfer process of $(\boldsymbol{l}, \boldsymbol{l}") \rightarrow \boldsymbol{l}'$ and $(\boldsymbol{l}, \boldsymbol{l}') \rightarrow \boldsymbol{l}"$ w.r.t. the different transferred destination frames (with the trifocal tensors calculated accordingly).

When $(\boldsymbol{I}, \boldsymbol{I}', \boldsymbol{I}")$ are not synchronized, the estimated trifocal tensors will be invalid and hence, the alignment constraint of the matched feature points/lines will be violated, which is reflected in the large distance between the transferred and the observed feature points/lines. This enables the use of the point/line transfer distance errors for evaluating the synchronization fitness of the three frames.

Specifically, part of the available features (hereafter we call them as "***geometry features***") are used to recover the tri-focal tensors of 3 views using the point based perspective factorization method [104]. Other methods [81] can be used as well. Then, the transfer distance errors are evaluated for the remaining point/line features (hereafter we call them as "***synchronization features***"). For simplicity and w.l.o.g., in the following, we assume that all the matching point features are used as "geometry features," while all the line features are used as "synchronization features."

With the tri-focal tensors recovered, the corresponding line transfer distance error of the *i*-th line triplet $(\boldsymbol{l}, \boldsymbol{l}', \boldsymbol{l}")$ is calculated as

$$\boldsymbol{e}_{Line}^{(\Gamma,i)}(\boldsymbol{l}, \boldsymbol{l}', \boldsymbol{l}") = d_{\perp}^2(\boldsymbol{p}_A, \tilde{\boldsymbol{l}}) + d_{\perp}^2(\boldsymbol{p}_B, \tilde{\boldsymbol{l}}) + d_{\perp}^2(\boldsymbol{p}'_A, \tilde{\boldsymbol{l}}') + d_{\perp}^2(\boldsymbol{p}'_B, \tilde{\boldsymbol{l}}') + d_{\perp}^2(\boldsymbol{p}"_A, \tilde{\boldsymbol{l}}") + d_{\perp}^2(\boldsymbol{p}"_B, \tilde{\boldsymbol{l}}") \quad (4.4)$$

where $(\tilde{\boldsymbol{l}}, \tilde{\boldsymbol{l}}', \tilde{\boldsymbol{l}}")$are the transferred lines; and $\boldsymbol{p}_{A(B)}(\boldsymbol{p}'_{A(B)}, \boldsymbol{p}"_{A(B)})$ are the end points of the observed lines $\boldsymbol{l}(\boldsymbol{l}', \boldsymbol{l}")$; and $d_{\perp}^2(\boldsymbol{p}, \boldsymbol{l})$ denotes the 2D Euclidean

distance from point $\boldsymbol{p}$ to line $\boldsymbol{l}$. Then based on the transfer distance errors of all matched line triplets, we define the line alignment measure for support voter Γ as

$$\boldsymbol{E}(\Gamma) = 3n_l/(\sum_{i=0}^{n_l-1} \boldsymbol{e}_{Line}^{(\Gamma,i)} + \mu) \tag{4.5}$$

where μ is a small positive value for avoiding the divide-by-zero problem. The point alignment measure can be defined similarly based on the point transfer distance errors. Therefore, the smaller the average of the point/line transfer distance errors, the larger the point/line alignment measure, and in turn the better the frames in question are synchronized. To maximize the alignment measure for finding the best synchronization, it is equivalent to minimizing the average of feature transfer distance errors.

It is noteworthy that the degenerate cases of the trifocal tensor based point/line transfer can be detected as shown in [81] and are excluded in the alignment measure.

4.4.3 Sub-frame synchronization and global optimization

With the integral synchronization offsets obtained, the sub-frame accuracy can be further achieved through a LM based non-linear optimization. It is recommended that the sub-frame refinement is performed for each group of three sequences synchronized during the incremental synchronization procedure to minimize error propagation.

Specifically, an error function

$$f(\Delta_1, \cdots, \Delta_k, \cdots \Delta_{M-1}) = \frac{1}{2}\sum \Gamma \in \boldsymbol{\nu}_{(\Delta_1,\cdots,\Delta_k,\cdots\Delta_{M-1})} \varepsilon(\Gamma)^2 \tag{4.6}$$

is minimized w.r.t. sub-frame synchronization offsets $(\Delta_1, \cdots, \Delta_k, \cdots \Delta_{M-1})$ by the LM optimization using finite difference. The optimization is initialized with the previously recovered integral synchronization offsets and $f(\Delta_1, \cdots, \Delta_k, \cdots \Delta_{M-1})$ is evaluated by linear interpolation for non-integral synchronization offsets. For the case of refining with only three sequences, the function of sum of feature transfer distance errors as defined in Section 4.3.2 is used as $\varepsilon(\Gamma)^2$, while for the case of globally optimizing with more than 3 video sequences, $\varepsilon(\Gamma)^2$ is defined as the sum of squares of the re-projection errors of all synchronization features.

4.4.4 Algorithm workflow

For better clarity, the workflow of our proposed algorithm is summarized as follows (List 4.1).

Step 1: Match and then track point and line features across and along sequences

Step 2: Incrementally process all sequences in groups of three. While for each 3-sequence group, the best integral synchronization offsets are hierarchically recovered by evaluating the support voters based on tri-focal tensor transfer as elaborated in Section 4.4.2. Sub-frame accuracy is further achieved using the LM optimization method as detailed in Section 4.4.3 to avoid possible error propagation due to incremental integration.

Step 3: The globally integrated synchronization offsets for all the sequences are refined using the same LM based optimization.

List 4.1: Workflow of our proposed algorithm

4.5 Experiments and Evaluations

To test the effectiveness of our proposed method, extensive experiments using synthetic and real video sequences have been done. Additionally, two other pairwise synchronization methods, i.e. the 5-point method [99] and the affine factorization measurement matrix rank based method [105] are also implemented and compared. To achieve sub-frame accuracy, the 5-point method is repeated for different 5-point configurations for ten times and the results are averaged. For the rank based method, the frame window size [105] used is 10 and the sub-frame resolution is set as 0.01. For each frame of the reference video sequence, the best sub-frame matching is searched in the to-be-synchronized sequence in the range of [-2.0, 2.0] centered with respect to the integer synchronization offset. We use a RANSAC based outlier elimination process before doing the least squares fitting the sub-frame offsets to avoid large sub-frame matching inconsistencies. Moreover, the comparison is also made in using point or line features as synchronization features when testing our new framework, in which the same

geometry features are used to avoid possible impact caused by the difference in tri-focal tensor calculations.

For simplicity, the following notations are used to describe the experimental details. In particular, $\boldsymbol{S}_m^{[u:v]\%n}$ denotes a sequence obtained by temporally down-sampling with a rate n to the $[u:v]$ sub-sequence of $\boldsymbol{S}_m$. Suppose $\left\{\boldsymbol{S}_0^{[u_0:v_0]\%n_0}, \cdots, \boldsymbol{S}_{M-1}^{[u_{M-1}:v_{M-1}]\%n_{M-1}}\right\}$ are the actual video sequences being synchronized at the *j*-th hierarchical level and the corresponding result $(\Delta_1, \cdots, \Delta_k, \cdots \Delta_{M-1})$ means that the corresponding timeline map between sequence $\boldsymbol{S}_0$ and $\boldsymbol{S}_m$ is

$$\mathbb{M}_{TL_0 \to TL_m}: \alpha_m(t_0 - u_0) + \Delta_m = t_m - u_m \tag{4.7}$$

i.e., all the frame-tuples $\Gamma\left(\boldsymbol{I}_0^{t_0}, \cdots, \boldsymbol{I}_{M-1}^{\alpha_{M-1}(t_0-u_0)+(\Delta_{M-1}\cdot n_{M-1}+u_{M-1}}\right)$in the original sequences are in synchrony.

Moreover, the 3D evaluation graph in the following illustrations shows the point/line geometric alignment measurements surface $z(x, y)$ for three sequences being synchronized, where $z(x, y)$ is the median of the alignment measurements of all the voters (frame-tuples) supporting the timeline maps based on specific synchronization offset relations $(\Delta_1 = x, \Delta_2 = y)$ as described above. Therefore, the offsets $(\tilde{x}, \tilde{y})$ corresponding to the highest peak in the 3D evaluation graph is just the integer offsets best synchronizing the sequences in question.

In the following experiments, the features are matched manually across sequences in the first frames and then tracked automatically. For point features, the KLT [106] point tracker is used. While for the line features, a line tracker based on the work of [107] is used. Here some remarks should be made on feature selection. First the selected features should be as spatially distributed as possible in the frames. Second, features with large relative position changes between frames are better because the difference of the recovered geometry due to slight off-synchronization but large movement may still be detectable. Hence, contrary to intuition, camera movements may even help in the synchronization. Finally, all the features are normalized [81] for better robustness in computing perspective factorization and tri-focal tensor transferring.

4.5.1 Synthetic videos

The synthetic video experiments are done to quantitatively evaluate the performance of our method with known ground truth. The synthesized dynamic scenes are constructed and rendered using POV-Ray [108] with the virtual cameras moving rectilinearly and uniformly. The frame rates are constant but need not be the same. Both the perspective and affine videos of the same scene are rendered for comparison. As an example shown in Figure 4.2, 5 sequences are synthesized, each of which starts at a different global time instant and spans for a different duration with frame rate $Fps_{(0,1,2)} = 25$ and $Fps_{(3,4)} = 50$. Figure 4.2 depicts the global timeline relations.

For our method, the 5 sequences are processed incrementally by synchronizing two sequence triplets $\{\boldsymbol{S}_0, \boldsymbol{S}_1, \boldsymbol{S}_2\}$ and $\{\boldsymbol{S}_0, \boldsymbol{S}_3, \boldsymbol{S}_4\}$. As for the other two methods, the synchronization is done pairwise w.r.t. the reference sequence $\boldsymbol{S}_0$.

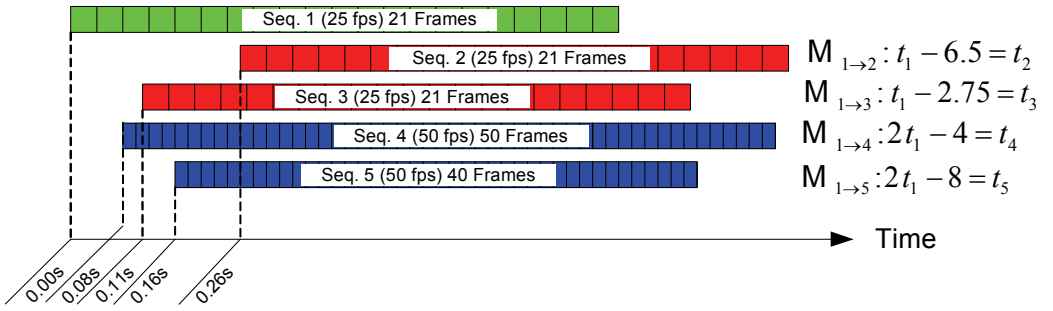


Figure 4.2: Global timeline relations between 5 synthetic videos.

In Figure 4.3, one can see that the line based method recovers the timeline maps correctly. The corresponding peak of the 3D line alignment evaluation is rather dominant and the sub-frame synchronization result is quite accurate $(\Delta_1 = -6.503, \Delta_2 = -2.712, \Delta_3 = -3.977, \Delta_4 = -8.030)$ as compared with the ground truth: $(\Delta_1 = -6.50, \Delta_2 = -2.75, \Delta_3 = -4.0, \Delta_4 = -8.0)$. In contrast, the other two methods incurred fairly large errors. For the 5-point method, the best recovered synchronization offsets are $(\Delta_1 = -8.60, \Delta_2 = -10.30, \Delta_3 = -0.70, \Delta_4 = -6.85)$ and for the rank-based method, they are$(\Delta_1 = -8.660, \Delta_2 = -12.665, \Delta_3 = 1.224, \Delta_4 = -7.180)$. As shown in the figure, both of our line and

point based implementations give comparable results, though there are some noisy peaks in the second group of results using the line-based method. The exact reason for this will be investigated in the future. While for the counterpart videos rendered using affine cameras, the results of the three methods (Note: In our method, we use affine factorization for the tri-focal tensor calculation.) all give very accurate results, with the two pair-wise ones showing much higher speeds. Although different features must be used in the affine video test due to the perspective difference, it is safe to conclude that large perspective distortions in the videos to be synchronized will have negative impact on the performance of the two pair-wise methods. Therefore, in the context where the affine camera model assumption is satisfied, these methods are still worthy for consideration for their low computation cost.

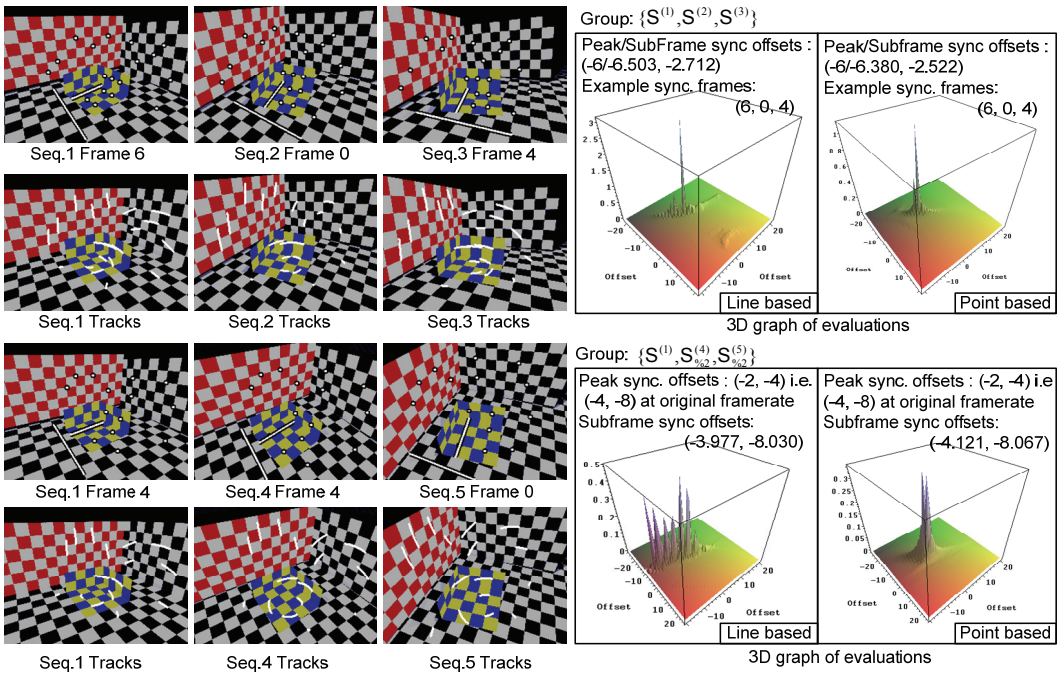


Figure 4.3: Synchronization result summary of 5 synthetic videos.

We further investigate the performance sensitivity of our synchronization algorithm. In particular, the tracked features are further distorted with different level of Gaussian noise to simulate feature tracking errors. At each noise level, 20 tests are performed, with the mean and standard deviation of the resultant synchronization offsets used for performance profiling. Please note here that the

noise is added to the features tracked using real trackers, which themselves may already include small errors.

Shown in Figure 4.4 is a typical profiling curve for synchronization results of $\boldsymbol{S}_1$ in the above simulation setup. As we can see, our algorithm can achieve pretty high synchronization accuracy when the tracking noise level is lower than 0.5 pixels. But for larger noise levels, larger synchronization errors of up to 2 frames may appear. One possible reason could be due to the number of features used is too small. Using more features and RANSAC based tri-focal tensor computation algorithm [81] may improve the robustness.

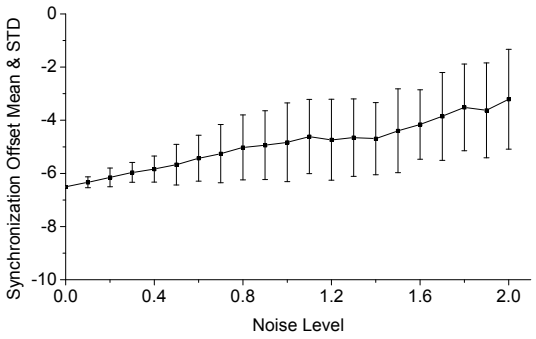


Figure 4.4: Example profiling curve of synchronization accuracy to tracking noise levels.

4.5.2 Real videos

Experiments are also conducted using real video sequences captured in the lab and outdoor.

First as shown in Figure 4.5, a lab scene with the electronic stopwatch shown on the wall is captured by two stationary and one hand-held moving Sony TRV120 DV camcorders. For synchronization, 2 lines and 2 points are used as "synchronization features," respectively, in the line and point based implementation, with the other 9 points used as the "geometry features." Since the sequences are a little long, they are synchronized hierarchically for better efficiency. Specifically, the temporally down-sampled sequences $\{\boldsymbol{S}_0^{\%4}, \boldsymbol{S}_1^{\%4}, \boldsymbol{S}_2^{\%4}\}$ are first synchronized at level L_1 with the line-based result of $(\Delta_1 = 5, \Delta_2 = -5)_{L_1}$, namely $(\Delta_1 = 20, \Delta_2 = -20)$ after transforming to the original frame-rate

context. Then guided by this initial result, three sub-sequences $\{\boldsymbol{S}_0^{[20:45]}, \boldsymbol{S}_1^{[40:65]}, \boldsymbol{S}_2^{[0:25]}\}$ are extracted and further synchronized at level L_0. The resulting synchronization offsets are $(\Delta_1 = 3, \Delta_2 = 4)$ which means that all frame tuples $\Gamma(\boldsymbol{I}_0^{t_0}, \boldsymbol{I}_1^{t_0+23}, \boldsymbol{I}_2^{t_0-16})$ are synchronized.

From the timing information transitions of the synchronized frames shown in the middle column of Figure 4.5 and by further considering the sub-frame optimization result $(\Delta_1 = 3.226, \Delta_2 = 4.127)_{sub}$ (namely, the sub-frame synchronization offsets of the original sequence $\{\boldsymbol{S}_0, \boldsymbol{S}_1, \boldsymbol{S}_2\}$ is $(\Delta_1 = 23.226, \Delta_2 = -15.873)_{sub}$), it can be seen that our result is quite accurate. However, our point-based implementation failed to get good result at the hierarchical level L_0, though more accurate result could be obtained by increasing the sub-sequence length. A possible reason might be due to the small inter-frame movement of the features. In contrast, for the other two compared methods, the 5-point based one gives the result of $(\Delta_1 = 25.60, \Delta_2 = -9.20)$ while the rank-based one gives the result of $(\Delta_1 = 26.176, \Delta_2 = -7.893)$, both of which incur large errors.

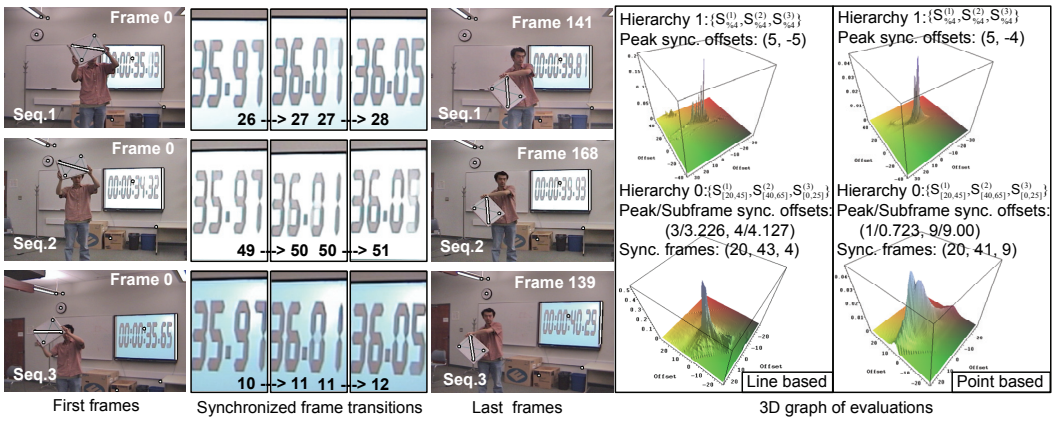


Figure 4.5: Synchronization result of the stop-watch sequence experiment.

Figure 4.6 shows another experiment in which three ping-pong sequences captured in a gymnasium using three stationary DV camcorders are synchronized. In total 10 point geometry features and 2 line or 2 point synchronization features are used. Since all the sequences are quite short, they are directly synchronized without building the hierarchical sequence pyramid. For the

sequences$\{\boldsymbol{S}_0, \boldsymbol{S}_1, \boldsymbol{S}_2\}$, the integral synchronization offsets are recovered as $(\Delta_1 = 10, \Delta_2 = 8)$. Although such result is quite close to the ground truth identified manually by checking the trajectory of the moving ping-pong ball bounced by the person walking from the right to the left, the off-synchronization error is still fairly large. Then by applying sub-frame synchronization optimization, more accurate sub-frame synchronization offsets are obtained, that is, $(\Delta_1 = 10.498, \Delta_2 = 9.830)_{sub}$. By carefully checking the timing instant (frame) at when the ping-pong ball hits the pad and its corresponding height from the pad in the next instant (frame), we can see that the sub-frame synchronization offsets are quite accurate. Specifically, the height of the ball from the pad in frame $\boldsymbol{I}_0^{15}$ of sequence$\boldsymbol{S}_0$ is obviously higher than that in frame $\boldsymbol{I}_1^{25}$ and lower than that in frame $\boldsymbol{I}_1^{26}$ of sequence $\boldsymbol{S}_1$. Therefore, the synchronization offset Δ_1 must be in the range of $(25-15, 26-15)$, that is, $(10,11)$. By further considering the kinetics characteristics of the ball movement, we estimate that the ground truth of Δ_2 should be very close to 10.5 which coincides with our result very well. While for sequence $\boldsymbol{S}_2$, we can also see that the inaccurate integral result is successfully corrected by the sub-frame optimization. On the other hand, our point-based implementation get the integral-frame result of $(\Delta_1 = 9, \Delta_2 = 8)$ and sub-frame result of $(\Delta_1 = 10.148, \Delta_2 = 10.542)_{sub}$, which is quite similar to the line-based one. While in the comparison, the 5-point and rank-based methods also get quite accurate results of $(\Delta_1 = 10.0, \Delta_2 = 9.0)$ and $(\Delta_1 = 10.018, \Delta_2 = 9.573)_{sub}$, respectively.

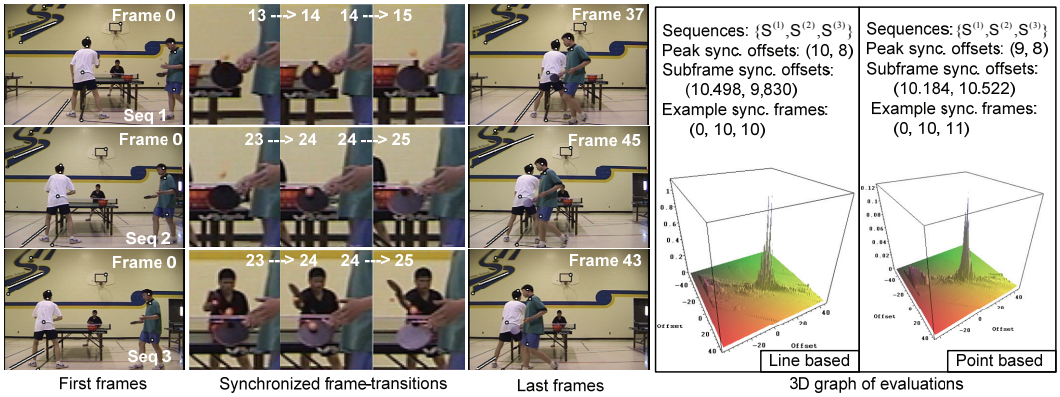


Figure 4.6: Synchronization result of the ping-pong sequence experiment.

Some wide-baseline videos experiments are also performed. In one example, shown in Figure 4.7, three toy car sequences captured by 2 stationary and 1 hand-held circularly moving DVs from different viewpoints are synchronized. In particular, 11 point geometry features and 2 line or point synchronization features are used. Our line-based implementation gets the integral-frame result of $(\Delta_1 = -2, \Delta_2 = 4)$ and sub-frame result of $(\Delta_1 = -1.729, \Delta_2 = 4.257)_{sub}$, while our point-based implementation gets the integral-frame result of $(\Delta_1 = -2, \Delta_2 = 5)$ and sub-frame result of $(\Delta_1 = -2.196, \Delta_2 = 4.705)_{sub}$. By inspecting the on-off transitions of the ornament LED lights on the car wheels or body, and the collision event of the toy car with the toy bricks, as shown in the transition column in Figure 4.7, we can see that such synchronization results are quite accurate, although the line-based result seems a little better. While for the other two methods, the 5-point method gets $(\Delta_1 = -3.30, \Delta_2 = 1.80)_{sub}$ and the rank-based one $(\Delta_1 = -1.978, \Delta_2 = 2.329)_{sub}$. Both methods incur synchronization errors of 2~3 frames.

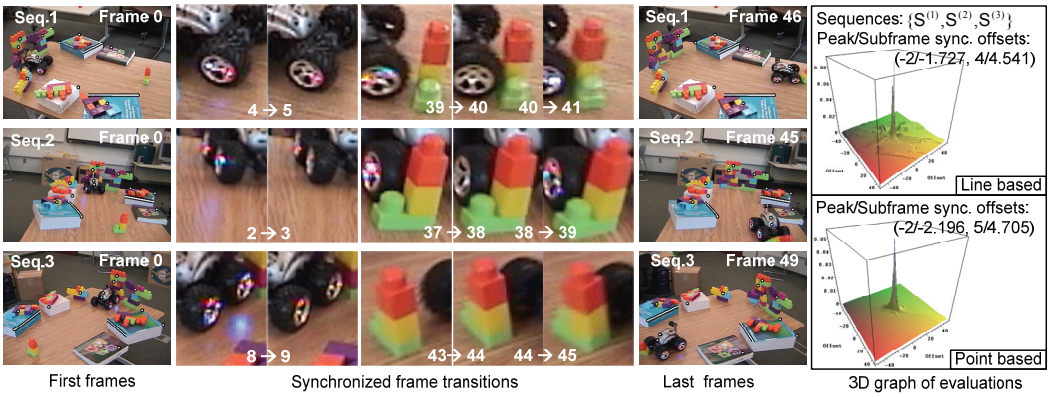


Figure 4.7: Synchronization result of the toy-car sequence experiment.

As shown in Figure 4.8, we also adopt the neat idea in [98] to test the accuracy of sub-frame optimization. Specifically, we first capture three hardware-synchronized sequences using the Point-Grey Digiclops® stereo vision camera system. All sequences are of length 53 frames and the known synchronization

offset ground truth is (0, 0). Then by taking interleaved frames starting from different starting frame, we create target sequences $\left\{\boldsymbol{S}_0^{[0:53]\%3}, \boldsymbol{S}_1^{[1:53]\%3}, \boldsymbol{S}_2^{[2:53]\%3}\right\}$ whose sub-frame synchronization offset should be $(\Delta_1 = -0.33, \Delta_2 = -0.66)_{sub}$. Applying our line-based implementation using 16 point geometry features and 2 line synchronization features, we get the integral synchronization offset of $(\Delta_1 = -1, \Delta_2 = -1)$ and sub-frame result of $(\Delta_1 = -0.417, \Delta_2 = -0.764)_{sub}$, which are quite close to the known ground truth $(\Delta_1 = -0.33, \Delta_2 = -0.66)_{sub}$. Our point based implementation also gets a similar result of $(\Delta_1 = -0.435, \Delta_2 = -0.778)_{sub}$.

Figure 4.8: Synchronization result of the hardware synchronized sequence experiment.

Summary

In this chapter, a novel tri-focal tensor based video synchronization has been presented. By using the tri-view geometry based point/line transferring distance as the synchronization alignment measure, high accuracy and robustness in synchronization are achieved. Even though its accuracy can not really compete with the hardware based approaches, it provides a more general and flexible solution for many application scenarios in which hardware based approach may not be applicable due to various practical constraints. Furthermore, the underlying idea in this work provides a solid foundation for the automatic total calibration algorithm to be presented in the next chapter.

Chapter 5
Efficient Geometric, Photometric, and Temporal Calibration of Unsynchronized Camera Array

This chapter addresses the camera array calibration problem, which is indispensable for all the FVV applications that require to reconstruct the 3D scene geometry explicitly. However, most of existing methods for camera array calibrations have limitations in some aspects.

For example, different from geometric camera calibration which has been heavily researched in the literature, photometric calibration is often assumed to be done separately or even ignored. Little effort has been made to provide an integrated solution for geometric and photometric calibration, in particular, for camera arrays. The few existing integrated geometric and photometric calibration methods usually need complicated procedures and take long processing time. Possible efficiency improvement is very much desired, especially for mobile camera array based applications, for which the calibrate-once approach may not work anymore.

On the other hand, most of the existing camera array calibration methods assume that the cameras being calibrated are synchronized. Although this assumption may be reasonable in some specialized cases, it does restrict the use of heterogeneous type of cameras or the diversity of their configurations. For unsynchronized cameras, many existing methods will fail to obtain accurate results or may require more user interactions. Even though the geometric calibration itself may be feasible by using snap-shots of a static pattern in different positions/orientations [42], FVV scene reconstruction and rendering do need the cameras to be temporally calibrated, i.e. synchronized, especially for scenes with fast moving objects.

Therefore, to address the above mentioned limitations and practical needs, in this chapter we present a novel integrated camera array calibration solution which is capable of performing geometric, photometric and temporal calibrations for an array of unsynchronized cameras. This new method extends the classic planar pattern based calibration method [28] to geometric and photometric calibrations and incorporates some of the ideas used in the last chapter for temporal calibration.

5.1 Problem Formulation

The problem of interest in this chapter is to calibrate $M(\geq 2)$ unsynchronized stationary cameras with similar formulation to the one discussed in Chapter 4. Each camera $\boldsymbol{C}_m(m \in [1, M))$ captures a video sequence $\boldsymbol{S}_m$ at frame rate Fps_m of the same dynamic scene. For geometric calibration, the cameras' intrinsic and extrinsic parameters are estimated w.r.t the reference camera $\boldsymbol{C}_0$.

Our new integrated calibration solution is based on Zhang's work [28] for its verified robustness, efficiency and accuracy. As a plane pattern based calibration method, it is assumed that in the scene there is a moving calibration pattern plane $\boldsymbol{\pi}$. Plane $\boldsymbol{\pi}$ is defined by $N(N \geq 4)$ 3D points $\boldsymbol{X}^j$ $(j \in [0, N))$ in the local object coordinate frame. Its location and orientation at time instant t_k defined in the local coordinate system of camera $\boldsymbol{C}_m$ are denoted by, respectively, the translation vector $\boldsymbol{T}_{\boldsymbol{\pi}}^m(t_k)$ and the rotation matrix $\boldsymbol{R}_{\boldsymbol{\pi}}^m(t_k)$. Then the undistorted projection of 3D point $\boldsymbol{X}^j$ to $\boldsymbol{x}_m^j(t_k)$ in camera $\boldsymbol{C}_m$ at time instant t_k can be defined as:

$$\boldsymbol{x}_k^j(t_k) = \boldsymbol{K}^m \begin{bmatrix} \boldsymbol{R}_{\boldsymbol{\pi}}^m(t_k) & \boldsymbol{T}_{\boldsymbol{\pi}}^m(t_k) \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{pmatrix} \boldsymbol{X}^j \\ 1 \end{pmatrix} \tag{5.1}$$

The actual observed point projection may be further distorted by the lens radial and tangent distortions.

Furthermore, since the cameras are not synchronized, the temporal calibration is needed to further recover the synchronization offset parameter Δ_m for each camera $\boldsymbol{C}_m$ as defined in the timeline mapping equation (4.1).

Then given M sequences capturing a moving calibration plane $\boldsymbol{\pi}$ using M unsynchronized cameras, the task of full camera array calibration is to estimate each camera's geometric parameters, both intrinsic and extrinsic, and its synchronization offset with respect to the reference camera. In addition to geometric and temporal calibrations, the photometric calibration is also done to recover a 3×3 color transformation matrix for each camera to normalize its color response characteristics w.r.t other cameras.

5.2 Algorithm Overview

The new calibration workflow includes the following steps.

At the first step, the intrinsic parameters of each camera are calibrated independently using the plane-based method [28]. However, instead of using the conventional checkerboard calibration pattern, a redesigned one which is capable of providing features for both geometric and photometric calibrations is used.

Then in the second step, the cameras are coarsely synchronized up to integer frame index and the initial estimations of extrinsic parameters are obtained for each camera using the video frames synchronized according to the integral synchronization result.

In the third step, the LM method based non-linear optimization is performed to find the cameras' optimal extrinsic parameters and sub-frame synchronization offsets by minimizing the projection errors of the moving 3D plane feature points in the video frames in which the calibration pattern is visible.

Finally, based on the photometric calibration patterns tracked together with the geometric calibration patterns, a color-transform based photometric normalization is done to minimize the color-inconsistency among the cameras.

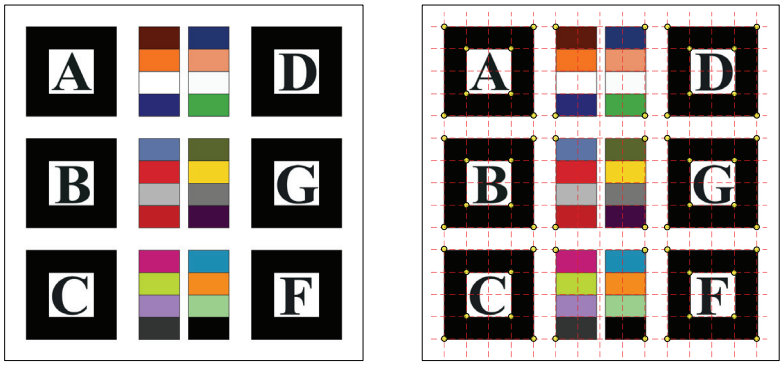
5.3 Implementation Details

5.3.1 Integrated Calibration Pattern Tracking

Similar to Zhang's work [28], our new method also requires to capture a number of 2D projections of the calibration plane $\boldsymbol{\pi}$ at different orientations and locations, which will provide the plane-induced homography constraints required for geometric calibration. For this, a 3D plane $\boldsymbol{\pi}$ is usually defined by several

geometry-known 3D feature points, whose 2D projections are easy to detect in the video frames. It is common practice to use a checkerboard pattern to facilitate plane detection.

However, to calibrate a large number of cameras as efficient as possible, it is desired to collect all the features required for geometric/photometric calibrations in just one pass. For this, a more versatile calibration pattern which is capable of integrating geometric and photometric features is preferred. One of the many possible patterns is shown in Figure 5.1.



(a) One redesigned calibration pattern (b) Implicit grid structure

Figure 5.1: One redesigned calibration pattern and the calibration features on an implicit grid structure.

In particular, we take advantage of the well-tested tracking robustness and flexibility of the black-and-white patterns (AR-pattern) from the well-known toolkit for augmented reality research - ARToolkit [109]. As can be seen in Figure 5.1 (a), each AR-pattern has its own "signature" so that it is easily distinguishable from the rest, resulting in better identifiably and reliability. Pattern tracking is done using binary template matching and history based tracking prediction. One advantage of using such patterns over using the traditional checkerboard pattern is that the complete detection of the calibration pattern is not required. Indeed, when any one of the AR-patterns is accurately detected, the detection of the remaining ones can be inferred and facilitated based on the known inter-pattern co-planar transformations. This feature is especially advantageous for calibration widely

spaced camera array for which the calibration pattern is often not entirely visible in all the video frames.

Also we integrate the rearranged Macbeth color checker patterns for photometric calibration in the middle columns, each of which can be automatically detected and tracked using the detected AR-patterns.

As shown in Figure 5.1 (b), each AR-pattern can provide up to 8 feature points (outer and inner corners of the black square ring), while each pigment-chip strip pair can provide 4 more feature points. Furthermore, we arrange the AR-patterns and color checker patterns so that such feature points are on an implicit grid structure. Just as in the case of using the traditional checkerboard pattern, all the feature point coordinates and inter-pattern transformations can be defined by using only one parameter, that is, the size of the implicit grid square block. Hence, possible errors due to inaccurate pattern printing and layout arrangements can be accommodated. By trading-off the pattern size which may impact the ease and accuracy of pattern detection, a total of six AR-patterns are used. Therefore in total, there are up to $N = 6 \times 8 + 3 \times 4 = 60$ feature points available for defining the calibration plane.

However, it should be noted that the specific drawing, number and arrangement of the geometric calibration patterns or color patterns are not restricted to the one shown above. They can be customized according to the user's specific requirements.

With the co-planar features tracked, the intrinsic parameters and distortion coefficients can be calibrated independently [28]. Then the multiple cameras are further geometrically and temporally correlated through calibrating the extrinsic parameters and temporal offsets w.r.t a user-specified reference camera. It is done in two steps: linear initialization and non-linear optimization.

5.3.2 Linear Initialization

As the step to bootstrap the following non-linear optimization, in linear initialization, the integer-frame synchronization is done first incrementally, which is similar to the synchronization procedure explained in the last chapter. However,

in here the synchronization is done together with extrinsic parameter calibration, the cameras are synchronized and calibrated in groups of two instead of three for better efficiency.

In particular, initially all the cameras except the reference one $\boldsymbol{C}_0$ are flagged as unsynchronized. For each unsynchronized camera $\boldsymbol{C}_m (m \in [1, M-1))$, it is first locally synchronized to a synchronized camera based on the synchronization-hypothesis-verification step to be detailed below. Then the local synchronization result is then upgraded to a global one w.r.t $\boldsymbol{C}_0$ via chained temporal transformations. Then camera $\boldsymbol{C}_m$ is flagged as synchronized and can be used as a peer camera for other unsynchronized ones. This incremental synchronization process will stop until all cameras are synchronized.

Together with this incremental initial synchronization process, the extrinsic parameter estimation process (in the following, we call it as "positioning" for simplicity) is done simultaneously since local synchronization of two cameras implicitly solves the local positioning problem as can be seen later. Similarly, an un-positioned camera is always first locally positioned w.r.t to a positioned (also synchronized) one while doing initial synchronization and then upgraded to the global coordinate frame by chaining the spatial transformations.

W.l.o.g, in the following, we only focus on how the initial synchronization and positioning are done for two cameras. The generalization to the remaining cameras is straightforward.

(a) Initial synchronization

Similar to what is described in the last chapter, to locally synchronize a specific camera pair, a range of integer-frame offsets are hypothesized and each possible offset is evaluated by the correspondingly deduced multiple view geometry alignment error of dynamic scene objects.

This range can be defined in an exhaustive way based on the sequence lengths as done in the integral offset synchronization procedure in Chapter 4. If prior knowledge on the rough synchronization offset range is known, a smaller range can be used instead for better efficiency.

For each hypothesized temporal offset, a number of "synchronization hypothesis voter" video frame pairs are first collected, that is, those video frames in which the calibration pattern features have been detected successfully and the frame index satisfying the timeline mapping deduced from the currently evaluated hypothesized temporal offset.

Instead of using the tri-focal tensor based geometric alignment error for uncalibrated cameras as in Chapter 4, we take advantage of the fact that the intrinsic camera parameters are known already and the 3D projection error is used as the synchronization measure.

Specifically, for each collected "synchronization hypothesis voter" video frame pair, since the calibration plane orientation and position in the respective camera coordinate frame are known through intrinsic parameter calibration, the inter-camera rotation and translation can be recovered by solving the corresponding coordinate system transformation. For all the video frame pairs, the "optimal" inter-camera rotation and translation corresponding to the temporal offset in question are obtained (see the next sub-section for more details).

In summary, for a hypothesized temporal offset, the rotation and translation between two cameras being synchronized are calculated. For a wrong temporal offset hypothesis, such inter-camera geometric transform will be wrong in general. This provides a new way for synchronization evaluation.

In particular, given an inter-camera geometric transform between two cameras, for each video frame pair, the 3D co-planar points in one view is transferred into another view and their corresponding projection errors are calculated. Then the average projection error for all the collected synchronization frame pairs is used to evaluate the fitness of the corresponding hypothesized temporal offset. This is based on the observation that for a moving calibration plane, unsynchronized frame pairs will capture the plane at different orientations and positions, which in general cannot be aligned by the same inter-camera geometric transform. Therefore, incorrect synchronization offset hypotheses will in general result in large projection errors. So the one resulting in the smallest error will be chosen as the initial integer temporal offset, which will be further

optimized. Meanwhile, the corresponding extrinsic (rotation and translation) parameters will be used for initial local positioning.

(b) Local positioning

Regarding the local positioning process of a camera pair, suppose we are recovering the relative rotation $\boldsymbol{R}_c^{k\to i}$and translation $\boldsymbol{T}_c^{k\to i}$ between an un-positioned camera $\boldsymbol{C}_k$and a positioned camera $\boldsymbol{C}_i$ whose global extrinsic parameters are already known as $\boldsymbol{R}_c^i$ and $\boldsymbol{T}_c^i$.

As a by-product of the plane based single camera calibration of camera $\boldsymbol{C}_i$, local translations $\boldsymbol{T}_{\boldsymbol{\pi}}^i(t_k)$ and rotations $\boldsymbol{R}_{\boldsymbol{\pi}}^i(t_k)$ of the calibration plane $\boldsymbol{\pi}$ at different time instants t_k can also be recovered w.r.t $\boldsymbol{C}_i$'s local coordinate frame. So the projection[1] of 3D plane point $\boldsymbol{X}^j$ in the frame captured by another camera $\boldsymbol{C}_k$ at time instant t_k should be

$$\boldsymbol{x}_k^j(t_k) = \boldsymbol{K}^k\,[\boldsymbol{R}_c^{k\to i} \quad \boldsymbol{T}_c^{k\to i}]\begin{bmatrix}\boldsymbol{R}_{\boldsymbol{\pi}}^i(t_k) & \boldsymbol{T}_{\boldsymbol{\pi}}^i(t_k)\\ \boldsymbol{0} & 1\end{bmatrix}\begin{pmatrix}\boldsymbol{X}^j\\ 1\end{pmatrix} \tag{5.2}$$

Meanwhile, since

$$\boldsymbol{x}_k^j(t_k) = \boldsymbol{K}^k\begin{bmatrix}\boldsymbol{R}_{\boldsymbol{\pi}}^k(t_k) & \boldsymbol{T}_{\boldsymbol{\pi}}^k(t_k)\\ \boldsymbol{0} & 1\end{bmatrix}\begin{pmatrix}\boldsymbol{X}^j\\ 1\end{pmatrix}$$

therefore we have

$$[\boldsymbol{R}_c^{k\to i} \quad \boldsymbol{T}_c^{k\to i}]\begin{bmatrix}\boldsymbol{R}_{\boldsymbol{\pi}}^i(t_k) & \boldsymbol{T}_{\boldsymbol{\pi}}^i(t_k)\\ \boldsymbol{0} & 1\end{bmatrix} = \begin{bmatrix}\boldsymbol{R}_{\boldsymbol{\pi}}^k(t_k) & \boldsymbol{T}_{\boldsymbol{\pi}}^k(t_k)\\ \boldsymbol{0} & 1\end{bmatrix} \tag{5.3}$$

Then for a given temporal offset, each synchronized frame pair between $\boldsymbol{C}_i$ and $\boldsymbol{C}_k$ will provide constraints for recovering the unknown $\boldsymbol{R}_c^{k\to i}$and $\boldsymbol{T}_c^{k\to i}$. To accommodate for possible inconsistencies between the results of $\boldsymbol{R}_c^{k\to i}$and $\boldsymbol{T}_c^{k\to i}$ obtained by different synchronized frame pairs, the median of all available results is picked. If the temporal offset is not close to the true one, such $\boldsymbol{R}_c^{k\to i}$and $\boldsymbol{T}_c^{k\to i}$

[1] Here for simplicity, we use the undistorted projections and the induction hereafter applies to the distorted projections too.

will in general result in large deviation between the 2D projections transformed from camera $\boldsymbol{C}_i$, i.e.,

$$\boldsymbol{x}^j_{transformed}(t_k) = \boldsymbol{K}^k \begin{bmatrix} \boldsymbol{R}^{k\to i}_c & \boldsymbol{T}^{k\to i}_c \end{bmatrix} \begin{bmatrix} \boldsymbol{R}^i_{\boldsymbol{\pi}}(t_k) & \boldsymbol{T}^i_{\boldsymbol{\pi}}(t_k) \\ \mathbf{0} & 1 \end{bmatrix} \begin{pmatrix} \boldsymbol{X}^j \\ \mathbf{1} \end{pmatrix} \tag{5.4}$$

and the actual observed ones, i.e.,

$$\boldsymbol{x}^j_{oberved}(t_k) = \boldsymbol{K}^k \begin{bmatrix} \boldsymbol{R}^k_{\boldsymbol{\pi}}(t_k) & \boldsymbol{T}^k_{\boldsymbol{\pi}}(t_k) \\ \mathbf{0} & 1 \end{bmatrix} \begin{pmatrix} \boldsymbol{X}^j \\ \mathbf{1} \end{pmatrix} \tag{5.5}$$

Therefore, as mentioned in the last section, the average projection error $\frac{\sum_{j=0}^{N-1} \left| \boldsymbol{x}^j_{oberved} - \boldsymbol{x}^j_{transformed} \right|}{N}$ provides a good measure for evaluating a hypothesized temporal offset.

Finally, to upgrade the local positioning to a global one, the initial estimation of $\boldsymbol{C}_k$'s global rotation and translation are obtained as:

$$\begin{cases} \boldsymbol{R}^k_c = \boldsymbol{R}^{k\to i}_c \boldsymbol{R}^i_c \\ \boldsymbol{T}^k_c = \boldsymbol{R}^{k\to i}_c \boldsymbol{T}^i_c + \boldsymbol{T}^{k\to i}_c \end{cases} \tag{5.6}$$

5.3.3 Non-linear Total Optimization

In this step, an LM-based non-linear optimization is performed to further improve the accuracy of the cameras' geometric parameters and synchronization offsets by minimizing the sum of squared projection errors of the feature points of the 3D calibration plane in the synchronized video frames. It can be regarded as a variant of the traditional bundle-adjustment [81] augmented with temporal parameters. Similar to the method presented in Chapter 4, the involved derivative calculation is done using finite difference and the projection error evaluation for frames synchronized up to sub-frame accuracy is done by linear interpolation.

In particular, the total number of optimized parameters is $(4+4) \times M$ for intrinsic parameters and lens distortion, plus $6\times (M-1)$ for extrinsic parameters, and plus $(M-1)$ for synchronization offsets. To improve optimization efficiency, the intrinsic parameters and lens distortion parameters can be dropped because the single camera calibration results are usually very accurate already.

5.3.4 Multiple Camera Photometric Normalization

In addition to the geometric and temporal calibrations, photometric calibration is also done automatically.

In particular, we follow the video post-processing based color normalization approach [42]. As shown before, for each camera, 24 Macbeth color pigment chips can be tracked based on their relative position transformations to the detected geometric calibration pattern. The samples of each pigment color are collected throughout each video and averaged (temporally) for modeling the corresponding camera's color response characteristics. It should be noted that for better accuracy, the camera's built-in features such as automatic exposure/gain/white balance adjustments should be disabled when capturing the videos.

Then the above temporally averaged colors are further averaged spatially for all the views and used as reference colors. To ensure color consistency among different cameras through normalization, a 3×3 RGB transform matrix is recovered for each camera as described in [42] to minimize the color difference between the camera's pigment response colors and the reference ones. Finally the color normalization is done by applying the corresponding RGB color transformation as a post-processing step.

Please note that as the Macbeth patches are printed, not painted, their real colors are not exactly the same as their original designs. Furthermore, the illuminant color is not factored out during the photometric calibration. Therefore the transformed video colors are usually not the same as the true color of the scene. That is, the above photometric calibration is only suitable for applications in which color consistency is more important than color fidelity, such as our interested FVV ones.

5.4 Experiments

We tested our new algorithm using several camera array configurations, with an illustrative one shown in Figure 5.2. In particular, 7 heterogeneous cameras are used, two of which are Sony camcorders (TRV-120/230) with Fps of 29.97 and

interlaced resolution of 720×480 and the other five cameras are Point Grey Research Flea2 cameras with FPS of 15 and progressive resolution of 1024×768. Although coarse synchronization among cameras can be achieved via TCP/IP communications, as can be seen later, systematic synchronization error could exist due to varying factors. Also due to the heterogeneity of cameras, the variation of colors among cameras is large.

The calibration procedure is fully automatic and the only user intervention is to first move the calibration pattern plane around in the observing volume of the cameras for 15 to 20 seconds. As can be seen from one example calibration scene shown in Figure 5.3, the view and color of each video stream are noticeably different from the rest. Additionally, the noise in the videos is quite apparent and makes accurate geometric and photometric calibrations challenging. Also to evaluate synchronization accuracy, an electronic stopwatch is put in the scene to provide synchronization ground truth information up to an observable accuracy limited by the highest camera frame rate.

Then for calibration, the geometric calibration features (as shown by the red circles) and color checker patterns (as shown by the yellow blocks) are tracked automatically and accurately, based on which all the cameras are calibrated successfully and efficiently.

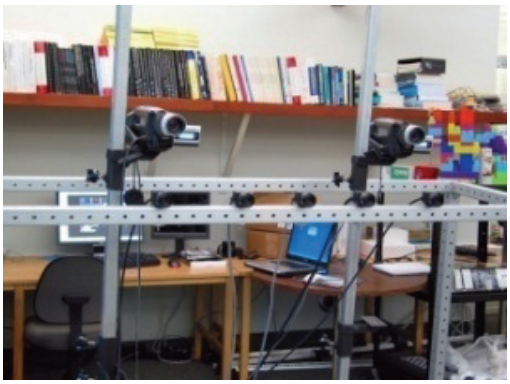


Figure 5.2: An example camera array setup.

For the temporal calibration result, we achieve the sub-frame synchronization offsets of $\boldsymbol{C}_m (m \in [1,7))$ w.r.t to the reference camera $\boldsymbol{C}_0$ as $[\Delta_1 = 0.00067, \Delta_2 = -0.00259, \Delta_3 = 0.614, \Delta_4 = 0.633, \Delta_5 = -5.474, \Delta_6 = -6.128]$. Figure 5.3

illustrates one group of synchronized frames of 4 cameras $\boldsymbol{C}_m (m \in \{0,1,3,5\})$ based on the above synchronization result. Specifically, according to the computed temporal offsets and Equation (4.2), we can deduce that frame 3 of camera $\boldsymbol{C}_0$ should be synchronized with frame 3.00067 of $\boldsymbol{C}_1$, frame 3.614 of $\boldsymbol{C}_3$ and frame 0.52 of $\boldsymbol{C}_5$ (please note that $Fps_0 = Fps_1 = Fps_3 = 15$ and $Fps_5 = 29.97$). From the stopwatch reading of the transitions in the consecutive frames shown in Figure 5.3 (c), we can see that these sub-frame offset results are quite accurate.

Fairly high accuracy is also achieved in geometric and photometric calibrations. In particular, the average projection error of the calibration pattern points after optimization is less than 0.3 pixels. The accuracy of calibration without performing synchronization is also compared, which results in an average projection error of 5.43 pixels. So it can be seen that the synchronization errors between cameras cannot be ignored. Accurate temporal calibration does help to improve the overall geometric calibration performance. While for photometric calibration, we evaluate its performance by the average (R, G, B) color difference and standard deviation of each camera's pigment response color to the reference color. As can be seen from the statistics listed in Table 5.1, the accuracy is comparable to the results reported in the work of Litos et al, [23] for similar heterogeneous camera configuration. The calibration error is mainly due to the automatic exposure and gain adjustment of the camcorders, which make the pigment color sampling slightly inaccurate when the large and bright calibration plane is moving in the scene. As a comparison, for the 5 Flea2 cameras, since their automatic exposure and gain adjustment features are disabled, a much higher accuracy is obtained.

More extensive calibration experiments are also done for different camera array setups with similar performance observed.

Color Channel	All 7 Cameras				5 Flea2 Cameras			
	Before normalization		After normalization		Before normalization		After normalization	
	Max Mean Error	Max STD.	Max Mean Error	Max STD.	Max Mean Error	Max STD	Max Mean Error	Max STD.
R	17.792	16.298	7.402	4.812	12.179	7.142	2.208	1.489
G	12.190	12.190	6.510	5.245	10.981	5.142	1.874	1.253
B	12.036	12.036	6.626	6.033	3.8139	5.218	2.259	1.142

Table 5.1: Photometric calibration error statistics.

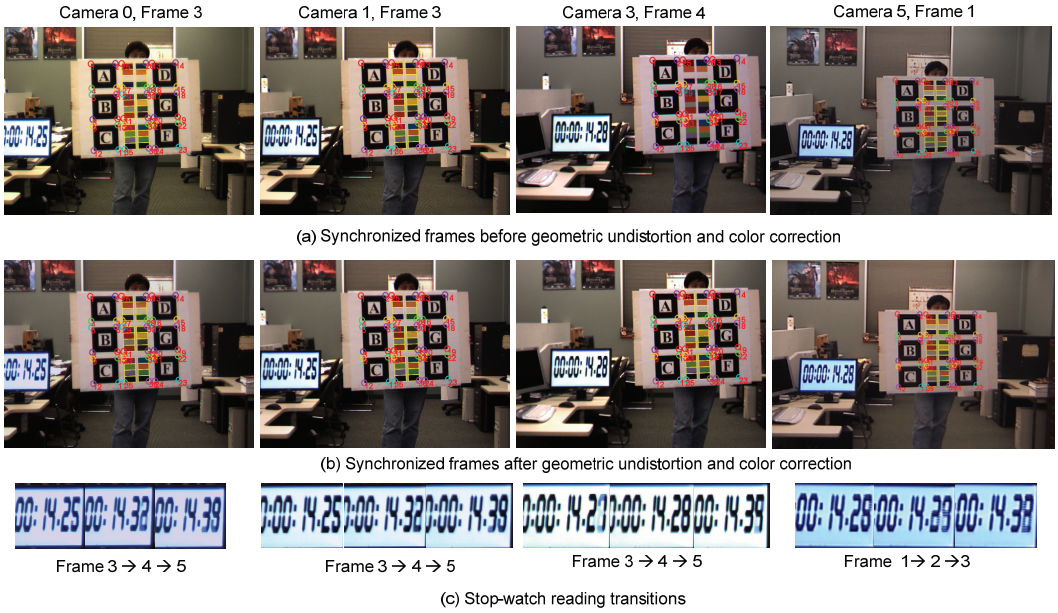


Figure 5.3: Results of 4 cameras $\boldsymbol{C}_m(m \in \{0,1,3,5\})$ (columns from left to right).

The proposed camera array total calibration method has been implemented as a built-in service in our cluster based FVV system and extensively tested and successfully used for calibrating many different configurations. Its automatic workflow, high efficiency and no requirement on hardware based camera synchronization enable fast scene acquisition for FVV rendering and applications. From the high quality depth scene reconstruction and rendering results to be presented in Chapter 9, it can be seen that the proposed method can provide accurate calibrations for our FVV system.

Summary

In this chapter, we have presented an automatic and efficient method for performing complete (geometric, photometric and temporal) calibrations of an array of unsynchronized video cameras by extending the classic plane based single camera geometric calibration method. It enables the use of heterogeneous types of cameras and supports more portable camera array configurations.

Chapter 6
A Region-Tree Based Image Labeling Framework

As we know that many important tasks in low level vision, image analysis and pattern recognition such as image restoration and texture modeling can be formulated as a discrete labeling problem, where the labels represent some task-specific local attributes. How to solve this class of labeling problems efficiently and accurately has attracted a significant amount of computer vision research over the past few years. In this thesis, we address the dense depth based FVV scene reconstruction task as an image labeling problem and propose a novel and general region-tree based image labeling framework.

6.1 Image Labeling Problem Formulation

Image labeling is essentially a discrete optimization problem, which is to assign each pixel in an image an optimal label from a limited number of candidates. To formulate the original task into a image labeling problem, a so-called label discretization process is usually needed if the original solution space is continuous. By sampling such continuous solution space at discrete points with fine enough sampling resolution, the original task can then be solved through discrete optimization. Most popular and successful approaches for addressing such image labeling problems all use the discrete energy minimization formulation [110].

Specifically, suppose a labeling target image $\boldsymbol{I}$ is represented as a spatial structure $\boldsymbol{\Omega}$ of a set of image primitives $\boldsymbol{\mathcal{P}}$ spanning the whole image, the task of image labeling is to assign each image (labeling) primitive $p \in \boldsymbol{\mathcal{P}}$ an optimal label ℓ_p from a label set $\boldsymbol{\mathcal{L}}$. Usually a weighted influence graph $\boldsymbol{\mathcal{G}} = (\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{E}})$ is also defined, wherein an edge $e_{pq} \in \boldsymbol{\mathcal{E}}$ represents the pair-wise influence between two

labeling primitives p and q and its corresponding edge weight w_{pq} encodes the strength of such influence.

Each image labeling solution $\boldsymbol{L}(\boldsymbol{\Omega})$ defines a mapping function $\boldsymbol{\mathcal{P}} \rightarrow \boldsymbol{\mathcal{L}}$ and is associated with an energy cost evaluating its fitness. Such energy cost usually consists of two terms – a data term and a smoothness term. On the one hand, assigning label ℓ_p to a labeling primitive p incurs the so-called data cost $\mathcal{D}(p, \ell_p)$ which measures the likelihood of such label assignment based on given observations. On the other hand, for each pair of labeling primitives p and q that are related, (i.e. connected by an edge $\in \boldsymbol{\mathcal{E}}$), there is the so-called smoothness cost $\mathcal{S}(p, q, \ell_p, \ell_q, w_{pq})$ which measures the compatibility of assigning primitives p and q with labels $\{\ell_p, \ell_q\}$, respectively, based on the corresponding edge weight w_{pq} and task-specific smoothness prior. The justification of introducing such a smoothness cost is that two strongly related labeling primitives are more likely to be assigned with similar labels. In this way, spatial piecewise smoothness can be preserved in the final labeling solution.

Then the labeling problem is solved by finding a labeling solution $\boldsymbol{L}(\boldsymbol{\Omega})$ which minimizes an energy function $E(\boldsymbol{L}(\boldsymbol{\Omega}))$ in the general form of

$$E(\boldsymbol{L}(\boldsymbol{\Omega})) = E_{data}(\boldsymbol{L}(\boldsymbol{\Omega})) + \lambda \cdot E_{smooth}(\boldsymbol{L}(\boldsymbol{\Omega})) \tag{6.1}$$

where the positive constant λ represents the relative weight.

The data energy term $E_{data}(\boldsymbol{L}(\boldsymbol{\Omega}))$ sums the data cost over all the labeling primitives as

$$E_{data}(\boldsymbol{L}(\boldsymbol{\Omega})) = \sum_{p \in \Omega} \mathcal{D}(p, \ell_p) \tag{6.2}$$

The smoothness term $E_{smooth}(\boldsymbol{L}(\boldsymbol{\Omega}))$ sums the smoothness cost over all related labeling primitive pairs as

$$E_{smooth}(\boldsymbol{L}(\Omega)) = \sum_{\forall e_{pq} \in \boldsymbol{\mathcal{E}}} \mathcal{S}(p, q, \ell_p, \ell_q, w_{pq}) \tag{6.3}$$

This energy minimization based formulation is closely connected to the theory of Markov Random Fields (MRFs). Specifically, the optimizing energy function (6.1) is essentially equivalent to finding the maximum posteriori estimate of a discrete MRF, with the potential function of the MRF defined as the distance function between labels [110]. Due to the equivalence to MRFs, solving an image

labeling problem is in general NP-hard and usually only approximate solutions can be found. Based on the underlying theoretical foundation, existing methods can be roughly categorized into two classes: those based on linear programming [111,112] and those based on discrete optimization [113-121]. The former class, though being more theoretically elegant, is not practical due to its excessive computational cost. Methods of the latter class are much more efficient and have been successfully applied in practice. Therefore, in this following, we focus only on the discrete optimization based techniques.

6.2 Workflow of Discrete Optimization based Image Labeling

Similar to the observations reported in work of Scharstein and Szeliski [122], most discrete energy minimization based image labeling algorithms follow a similar workflow pattern in performing the following four steps (or a subset of them). The specific execution order of such steps as well as the exact strategy used in each step is what distinguishes different algorithms.

(a) Data Cost Computation

In this step, for each label $\ell \in \boldsymbol{\mathcal{L}}$, the corresponding data cost is evaluated for each pixel in the labeling target image $\boldsymbol{I}$. Based on the specific application, different number of additional images can be involved. Usually a data cost value is calculated from the difference between the intensity of a pixel (x, y) in image $\boldsymbol{I}$ and the intensity of a relevant pixel induced from the label ℓ in image $\boldsymbol{I}$ itself or other involved images. For example, in image restoration, the induced intensity is just the label itself. While in stereo matching, the induced intensity is the intensity of the corresponding pixel(s) found using the label (disparity or depth) in the image(s) being matched.

The resultant data cost values over all pixels and all labels form a 3 dimensional image, which is called ***label space image*** $c(x, y, \ell)$.

(b) Data Cost Aggregation

Pixel-wise data cost evaluation usually cannot reflect the true label assignment fitness distribution due to noise or imperfect cost calculation. To enhance robustness, the initial label space image $c(x, y, \ell)$ is often "filtered" by summing

or averaging the data costs over an aggregation support region, which can be either two dimensional in the x-y space or three-dimensional in the x-y-ℓ space. This process is called data cost aggregation. When a fixed regular support region is used, the aggregation can be performed using 2D or 3D convolution. For example, in the case of a rectangular window based support region, the aggregation can be done using an efficient box filtering. After the aggregation, an aggregated label space image $c'(x, y, \ell)$ is obtained.

On the other hand, since the label space image is only pixel based, for image representations using non-pixel labeling primitives, further aggregation is needed to evaluate the data costs for the non-pixel primitives.

(c) Labeling Computation and Optimization

In this step, the optimal labeling solution minimizing energy function (6.1) is calculated using local or global optimization.

Computing the optimal labeling assignment for each labeling primitive using local optimization method is usually trivial and fast. In particular, the label associated with the minimum data cost values for each labeling primitive is chosen as the optimal one. That is, the local "Winner-Take-All" (WTA) optimization is performed for each labeling primitive. However, since the smoothness costs are ignored, the labeling solution obtained by local optimization usually contains many errors. Therefore, local methods are mainly used in efficiency critical scenarios.

In contrast, global optimization needs more intensive computation due to the enforcement of pair-wise smoothness costs and searching of the global minimum. A variety of algorithms such as continuation [123], simulated annealing [124], dynamic programming [113-116], graph cuts [117, 118] and belief propagation [119-121] have been applied to solving the underlying discrete optimization problem.

(d) Labeling Refinement

Due to efficiency considerations and computation resource limits, during the label discretization process, it is usually needed to constrain the size of the label set $\mathcal{L}$. However, in some cases, the corresponding discretization resolution may not be fine enough for the required accuracy. Therefore, in this step, labeling refinement can be performed to further enhance the resolution of the labeling solution obtained in the last step. In addition, for algorithms using non-pixel based image primitives, pixel-wise labeling refinement may also be done based on the labeling assignment of the corresponding container primitive.

As well, there are also other application-specific ways of post-processing the obtained labeling solution. For example, median filtering or diffusion can be applied to "clean up" spurious label assignments in many applications. In binocular stereo matching or motion estimation, cross-checking can also be done to detect and correct errors due to occlusions [125].

6.3 Motivation

Most image labeling algorithms focus on the above labeling computation and optimization step by designing better discrete optimization techniques. With powerful new discrete optimization algorithms such as graph cuts [117,118] and loopy belief propagation [119-121] proposed and popularized in the last few years, the performance improvement in solving various image labeling problems has been very encouraging, which in turn further enables many novel applications such as interactive photo editing [126]. Therefore, most popular image labeling frameworks are categorized according to the underlying discrete optimization technique.

However, in addition to discrete optimization techniques, image representation also plays an important role in solving the image labeling problem.

So far, two image representations are most often used in the existing image labeling frameworks, in particular, the traditional pixel grid and the layer (or plane) representations [127, 128]. The former one uses a single pixel as a labeling primitive and a regular grid as the spatial spanning structure $\mathbf{\Omega}$. It is simple to use

and can be applied to any rectangular shaped image. But it suffers from higher labeling ambiguities because it loses some useful scene-based smoothness prior, which is implicitly and easily enforced in the layer-based approaches. To address this issue, hypothesis label evaluations from local neighborhood pixels are usually used. The shape of such local neighborhood can be a rectangle or an irregular region. In particular, in the local optimization based labeling framework [129], for each pixel and each hypothesized label assignment, a region with spatially varying size is constructed as a connected component containing pixels for which the likelihood of such hypothesized label assignment is high. Then the label which gives the largest region is chosen as the final label assignment for that pixel.

The layer based representation uses a layer as a labeling primitive which results in better ambiguity discrimination performance. It has unique potentials in various image labeling problems, especially in the classic binocular stereo matching [127-128, 130] and motion estimation [131].

According to the de facto stereo benchmarking [122], the current top two ranked stereo methods [130, 127] are both layer based. In particular, in [[130], an inter-regional cooperative optimization scheme is proposed. Local window-based pixel matching is done first to fit a disparity plane for each color-segmented region. And then the parameters of all such disparity planes are iteratively optimized to obtain the final disparity map. A similar idea is also used in [127], in which the scene structure is modeled by a set of planar surface patches through color segmentation. Disparity planes are extracted by applying local window-based pixel matching and segmentation-based clustering. The disparity plane label assignment for each region is optimized using belief propagation.

One drawback of such layer based representation is its reliance on accurate segmentation and the difficulties of layer parameterization without prior knowledge. Errors in layer segmentation due to violation of the assumption that layers do not cross label discontinuities and in parameterization modeling due to over-fitting can incur errors in the labeling result. Also for non-structured scenes in which there are no dominant layers (planes), its performance could be degraded.

To address the above issues, researchers propose over-segmentation based approaches [132, 133]. An over-segmented region can be regarded as a trade-off between a pixel and a layer. Compared to a pixel, it enables a labeling primitive to contain enough information with a large support area. On the other hand, it also reduces the risk of violating the parameterization assumption with a relatively small region size compared to a layer. Due to such advantageous properties, representations using over-segmented regions have shown great potentials. Because there is much fewer number of regions than pixels, the computational complexity is reduced compared to pixel-based representations. Also given the smaller size of the regions, the chance that color segmentation errors propagate into the labeling process using over-segmented regions is reduced compared to those of using layers, which are usually much larger compared to over-segmented regions.

In particular, in [132], the image is first over-partitioned into a grid of equal sized square patches and then K-means clustering is applied to refine the over-segmentation through region merging and splitting. Depths of regions are computed using loopy belief propagation optimization over the corresponding region adjacency graph (RAG).

In the work of Taguchi et al. [133], over-segmentation is also used for simultaneous binocular stereo reconstruction and alpha matting of mixed region boundary pixels iteratively. The over-segmentation is done w.r.t. both input images for consistency and iteratively updated based on region color, depth and shape constraints. For a given segmentation, the depths of the regions are estimated using belief propagation optimization, which are in turn used for updating the segmentation in the next iteration. In this way, the segmentation and the depth map as labeling results are updated iteratively until convergence.

All of the above mentioned encouraging results of using over-segmented based image representations in stereo matching motivate our work in this thesis to be described in the next three chapters. In particular, we introduce a general region-based image labeling framework using a novel image representation – a spanning tree of over-segmented regions. Different from previous work on using over-

segmentation, using a tree structure instead of a more general graph structure enables the use of fast optimization techniques such as dynamic programming (DP). Also it can avoid unnecessary smoothness enforcement between regions across labeling discontinuities, which may incur errors in the labeling solution.

The most related previous works in this regard are Veksler's pixel tree [115] and Deng and Lin's line segment tree based stereo [134], both of which are proposed mainly for binocular stereo matching and not as a general framework. In particular, Veksler [115] extends the traditional DP based stereo matching by introducing the so-called pixel-tree structure. A pixel-tree is formed by keeping only the "most important" edges to 4 connected neighbor pixels in the pixel lattice graph based on color similarity. Applying DP to such a 2D pixel-tree structure enables the vertical consistency to be enforced between scanlines, which cannot be done in traditional 1D scanline based methods, resulting in truly global optimization since the label (disparity) assignment for each pixel depends on assignments of all the other pixels.

In the work of Deng and Lin [134], scan-lines are partitioned into segments and the resulting line segments are used as tree vertices instead of single pixels. The connections between adjacent pixels within the same segment are preserved as completely as possible, resulting in better stereo matching performance compared to pixel tree [115].

In our new proposed framework, similar to the above two methods, a spanning tree structure is also built, but with a more general and flexible tree vertex definition for over-segmented regions.

Specifically, the labeling target image is first over-segmented into a set of regions using the mean-shift algorithm [135]. From the resultant region adjacency graph, a weighted spanning region tree is extracted. Then the labeling problem is solved by optimizing the energy function (6.1) defined on such a region tree.

The proposed framework is versatile. By using different label definitions and data/smoothness cost functions, the same framework can be applied to solving a wide spectrum of image labeling problems such as the classic binocular, multi-

view stereo matching and optical flow estimation, all of which are very important by themselves and in our FVV applications.

The proposed framework is also extensible and can be adapted to different applications. For example, as to be elaborated later, by taking advantage of the bottom-up granularity controllable fusion based image segmentation procedure, the coarse-to-fine (C2F) scheme can be easily integrated and utilized. On the other hand, different region tree edge-weighting schemes or region tree extensions such as the temporal variant used in Chapter 9 may be used or made for different applications.

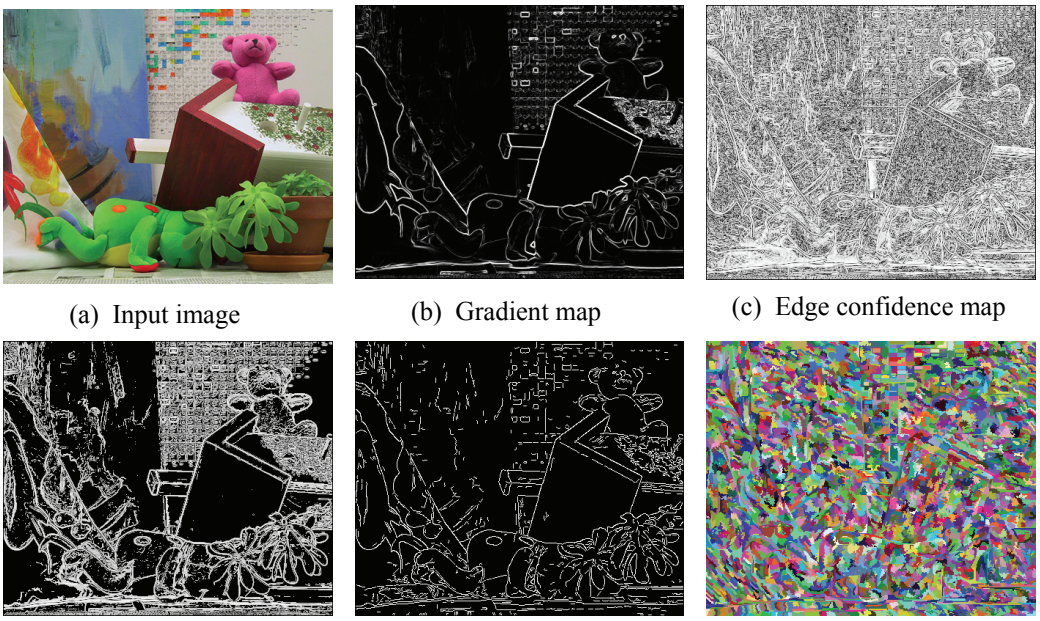
6.4 Region-tree based Image Representation

In the proposed framework, the labeling primitive set $\boldsymbol{\mathcal{P}}$ is a set of over-segmented regions $\boldsymbol{\mathcal{R}}$. The spatial structure $\boldsymbol{\Omega}$ is a spanning tree $\boldsymbol{\mathcal{T}}$ over $\boldsymbol{\mathcal{R}}$, which is extracted from the region adjacency graph (RAG) created as the segmentation result. The region-tree $\boldsymbol{\mathcal{T}}$ itself is used as the influence graph in solving the labeling program in question.

6.4.1 Mean-shift based Image Over-segmentation

To build the corresponding region-tree $\boldsymbol{\mathcal{T}}$, the labeling target image is first over-segmented using an improved variant [136] of the mean-shift algorithm [135].

In particular, the segmentation consists of two steps, namely, mean-shift filtering and pixel/region bottom-up fusion. In the filtering step, in addition to transforming its intensity/spatial information into feature space parameters, a weight is also assigned to each pixel. Such weight is defined using its normalized rank of the gradient magnitude and edge confidence. It is used as the edge-likelihood measure to enhance the contributions of edge-like pixels and to suppress the contribution of non-edge-like pixels when calculating the mean-shift moves in feature space. In this way, the discontinuity preserving property of the mean-shift filtering is further enhanced. Shown in Figure 6.1 (d) is one example of the weight map.

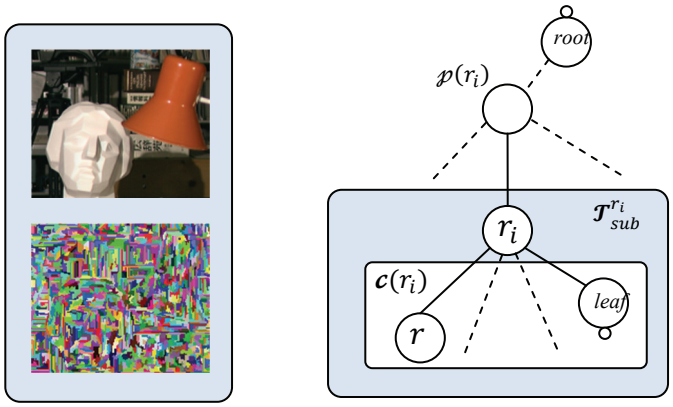


(a) Input image (b) Gradient map (c) Edge confidence map

(d) Weight map (e) Edge map (f) Over-segmentation

Figure 6.1: Over-segmentation illustration. Given an input image, its gradient and edge confidence maps are created first, from which the weight map is calculated. The weight map is then used for mean-shift based segmentation and edge detection.

After the filtering step is done, the fusion step iteratively fuses smaller regions (pixels in the first iteration) into larger ones by performing transitive closure operations to the corresponding RAG. Specifically, for each edge in the RAG, a boundary strength measure $\varkappa$ is computed as a "fusibility" measure between two regions by averaging the above mentioned weights of pixels on the boundary shared by the two regions. By thresholding the maximum boundary strength between any two to-be-fused regions, discontinuity-preserving region fusion can be achieved. Furthermore, different segmentation granularities can be achieved by controlling the number of fusion iterations and the minimum region size. An illustration of a typical over-segmentation result is shown in Figure 6.2.



(a) Over-Segmentation Example (b) Region-tree illustration

Figure 6.2: . Illustrations of region-tree generation.

6.4.2 Region-tree Generation

With the results of mean-shift based over-segmentation, the labeling target image has been partitioned into a set of regions $\boldsymbol{\mathcal{R}}$ and the corresponding RAG $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{E}}')$ is created**.** Each region $r_i \in \boldsymbol{\mathcal{R}}$ has N_{r_i} pixels $(x, y) \in r_i$ and corresponds to a mode m_{r_i}, which is a mean-shift clustering point in the multiple dimensional feature space. For color images, the dimension is 5 with the first three for the color channel values in the CIELUV color space and the other two for the *x* and *y* coordinates. For grey-level images, the dimension is 3 with the first one for the pixel intensity value and the other two for the *x* and *y* coordinates. Each vertex in $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{E}}')$ corresponds to a region r_i and each edge $e_{(i,j)} \in \boldsymbol{\mathcal{E}}'$ corresponds to a link between two adjacent regions r_i and r_j.

From the RAG $\boldsymbol{\mathcal{G}}(\boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{E}}')$, we further extract a region tree by keeping only the "most important" edges. Then an obvious question is on how to evaluate the importance of a given edge. An intuitive criterion is that, the more likely that two regions have the same label, the more desirable is to keep the edge between them. Therefore, each edge $e_{(i,j)}$ is assigned with a positive edge weight $w_{e_{(i,j)}}$ encoding the *un*-likelihood for regions r_i and r_j to share the same label. The larger the edge weight $w_{e_{(i,j)}}$, the less likely regions r_i and r_j to have the same label so that the less important is the edge $e_{(i,j)}$.

In this thesis, we define the edge weight based on the Euclidean distance $dis(m_{r_i}, m_{r_j})$ between two region's modes in the feature space of mean-shift segmentation and the strength of the common boundary between the two regions $\varkappa_{r_i \leftrightarrow r_j}$ as

$$w_{e_{(i,j)}} = dis(m_{r_i}, m_{r_j}) \cdot e^{\varkappa_{r_i \leftrightarrow r_j}} \tag{6.4}$$

It should be noted that different edge weighting schemes can also be used for specific applications, which may impact the resulting region-tree.

Then we construct a minimum spanning tree (MST) $\boldsymbol{\mathcal{T}}(\boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{E}})$ using the Prim greedy MST method [137] by traversing all region vertices in $\boldsymbol{\mathcal{G}}$ and by minimizing the sum of weights of the remaining edges $\sum_{e_{(i,j)} \in \boldsymbol{\mathcal{E}}} w_{e_{(i,j)}}$. In this way, each specific region is only linked to the regions which are most likely to have the same label so that the adaptive piecewise labeling smoothness can be enforced during optimization.

6.5. Energy Minimization over a Region-tree

With the labeling target image represented as a spanning tree of over-segmentation, the optimization of energy function (6.1) defined over such a region-tree $\boldsymbol{\mathcal{T}}(\boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{E}})$ can be done by any discrete optimization methods supporting non-grid labeling primitive adjacency system such as graph cuts or belief propagation. However, the unique tree structure also makes it possible to use the more efficient DP optimization. As we know, to apply DP, the problem in question must exhibit an optimal structure. Our region-tree based labeling optimization does have such an optimal structure. Indeed, the optimal solution (partial labeling) for any sub-region-tree must also be optimal for the solution for the whole region-tree.

Specifically, taking the sub-region-tree $\boldsymbol{\mathcal{T}}_{sub}^{r_i}$ rooted at vertex (region) r_i shown in Figure 6.2(b) as an example. Suppose vertex r_i is not the global root of the whole region tree $\boldsymbol{\mathcal{T}}$, then the minimum value of the energy as defined in Equation (6.1) on the sub-tree $\boldsymbol{\mathcal{T}}_{sub}^{r_i}$ plus the edge between vertex r_i and its

parent vertex $\mathcal{p}(r_i)$ can be recursively written as a function of the label $\ell_{\mathcal{p}(r_i)}$ assigned to $\mathcal{p}(r_i)$:

$$E_{r_i}(\ell_{\mathcal{p}(r_i)}) = \min_{\ell_{r_i}\in\mathcal{L}}\left(\mathcal{D}(r_i,\ell_{r_i}) + \lambda\cdot\mathcal{S}(r_i,\mathcal{p}(r_i),\ell_{r_i},\ell_{\mathcal{p}(r_i)}) + \sum_{r\in\mathcal{c}(r_i)} E_r(\ell_{r_i})\right) \quad (6.5)$$

where $\mathcal{c}(r_i)$ is the set of the children vertices of vertex r_i.

Then let's further investigate two types of special vertices in the whole region tree: the leaf and the root.

For a leaf vertex, since it has no child vertex, then Equation (6.5) can be directly evaluated for each possible label assignment for its parent vertex, that is,

$$E_{leaf}(\ell_{\mathcal{p}(leaf)}) = \min_{\ell_{leaf}\in\mathcal{L}}\left(\mathcal{D}(leaf,\ell_{leaf}) + \lambda\cdot\mathcal{S}(r_{leaf},\mathcal{p}(leaf),\ell_{leaf},\ell_{\mathcal{p}(leaf)})\right) \quad (6.6)$$

Therefore, for a leaf vertex r_{leaf}, if the label of its parent vertex $\mathcal{p}(leaf)$ has been assigned somehow, its optimal label ℓ_{leaf} can be easily determined.

While for the root vertex, it has no parent vertex so that Equation (6.5) can be simplified to

$$E_{root} = \min_{\ell_{root}\in\mathcal{L}}\left(\mathcal{D}(root,\ell_{root}) + \sum_{r\in\mathcal{c}(root)} E_r(\ell_{root})\right) \quad (6.7)$$

That is, with E_{root} known for each possible root vertex label assignment and for each child vertex of the root vertex, the best label ℓ_{root} can be directly recovered. Furthermore, it can be seen that the minimum energy E_{root} of the root vertex is just the minimum value of the energy (6.1) of the whole region tree. So the labeling assignment optimizing E_{root} is just the one optimizing Equation (6.1).

Based on the above observed properties, the energy function (6.1) can be optimized using DP through two recursive region-tree traversals. In the first bottom-up traversal starting from the leaf vertices, the optimal label that minimizes the energy (6.6) and the corresponding minimal energy are determined for each possible parent vertex label. Then upon reaching the root vertex, the optimal label of the root vertex is trivially determined and the region-tree is breadth-first traversed again starting from the root vertex. In this turn, for each visited non-root vertex, since the parent's optimal label has been set, its optimal

label can be easily determined accordingly. Therefore, in this way, all the optimal labels of the vertices (regions) can be recovered. Since we assume all the pixels in a region share the same label, then the original pixel-wise labeling problem can be solved.

6.6. Coarse to Fine (C2F) Region-tree

As mentioned before, when using discrete labeling formulation to solve real world problems, the original usually continuous solution space has to be quantized and mapped to a discrete set of labels. Direct brute-force discretization usually suffers from the so-called "discretization bottleneck" problem, which means that the number of labels required for sampling the whole possible solution space with fine enough resolution could be too large for efficient optimization. This issue is even more problematic when the label corresponding to multi-dimensional attribute of a labeling primitive such as 2D displacement in optical flow estimation.

To address this issue, we further extend our framework by enabling the use of a new C2F region-tree representation. In particular, during the mean-shift based image segmentation process, by using different minimal region size thresholds, multi-level of image over-segmentations with different segmentation granularities can be obtained. As shown in Figure 6.3, the bottom-up fusion based segmentation process as explained in Section 6.4.1 makes it possible to obtain such multiple level over-segmentations efficiently in one single pass. Also it can guarantee that each larger region in the coarser segmentation level consists of exactly the same smaller regions in its corresponding finer level segmentation. That is, each region in a coarser segmentation level consists of a group of adjacent regions of a finer segmentation level, such that the coarser region is the "container" region of these adjacent smaller regions. Then for each segmentation level, a corresponding region-tree can be built as explained in Section 6.4.2. Hence, all the segmentation levels together form a multiple-layer C2F region-tree. Its main difference from the single-layer region-tree is that there are implicit

edges between segmentation layers linking coarse-level container region to its finer-level "child" regions.

Then the labeling problem is solved level by level starting from the coarsest segmentation one, at each of which a similar region-tree based DP optimization is performed. In particular, at the coarsest segmentation level, a global label set is used for all the labeling primitives, while for each of the remaining finer segmentation levels, the labeling solution from the last coarser level will be used to define the label sets locally for each region. Hence, each region maintains its own label set for data/smoothness cost evaluation and optimization. In this way, C2F label discretization is achieved so that the original solution space can be explored by using only a relatively small number of labels. Such an incremental refinement scheme can greatly improve the efficiency and performance of a labeling algorithm to be shown in the following chapters.

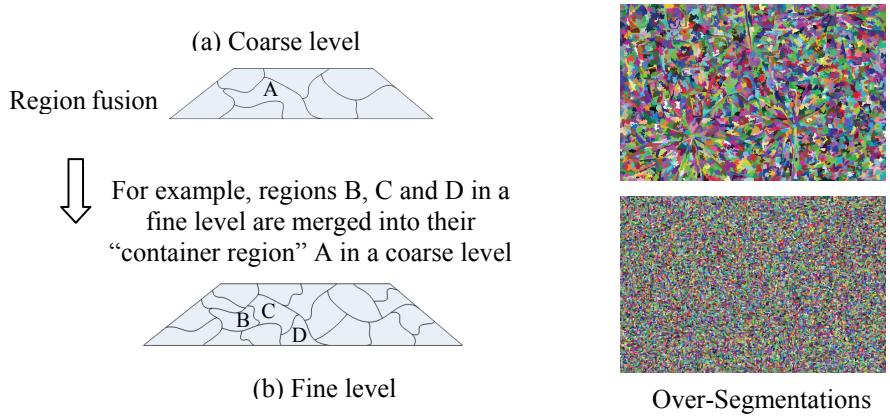


Figure 6.3: Two-level coarse-2-fine over-segmentations.

Summary

In this chapter, a novel region-tree based framework for image labeling problems has been presented. Such a framework forms the foundation of the dense depth based scene reconstruction module/service of our FVV system. In the following chapters, three new applications of the presented framework are discussed, which include binocular stereo matching, optical flow estimation and spatial-temporal consistent video depth recovery.

Chapter 7
Region-Tree Based Binocular Stereo Matching

Perhaps it is the most direct way of reasoning 3D information from image data, stereo vision has a wide range of applications in computer vision or graphics research. It has been an extensively researched topic for decades and a large number of new methods are published every year.

Generally speaking, the main purpose of stereo vision is to find matching points among a set of images and then recover the corresponding depth information based on triangulation. Most existing matching methods are based on the brightness constancy assumption that the scene objects are Lambertain so that object appearance does not very with viewpoint.

Based on the required density of correspondences, methods can be divided into the sparse and dense stereo matching. In the sparse methods, only a small number matching feature points, such as Harris [138] or SIFT corners [139], between images based on intensity cross correlation or affine invariant region properties are used. In contrast, for the dense methods, all the pixels in an image must be matched with the given images or flagged as occluded. Sparse stereo matching is relatively easy compared to the dense one since such feature points generally are more salient resulting in lower matching ambiguity. In this thesis, the focus is on dense stereo matching.

In this chapter, the image labeling framework presented in Chapter 6 is first applied to binocular stereo matching, for which the dense pixel correspondences can be recovered directly in 2D image space without calibration and encoded as disparity map. The same method can also be easily extended to grid-positioned multi-ocular stereo as reviewed in Chapter 2. In Chapter 9, the general-positioned multi-ocular stereo problem is further investigated, for which the cameras normally need to calibrated.

7.1 Formulation as a Labeling Problem

To apply our region-tree framework, the binocular stereo matching problem has to be formulated as a discrete labeling problem.

In particular, given two stereo images $\boldsymbol{I}_{left}$ and $\boldsymbol{I}_{right}$ and suppose the maximum and minimum disparities are known as d_{max} and d_{min}, the label set $\boldsymbol{\mathcal{L}}$ can be defined as a set of integers, each of which serves as an index into a disparity look-up-table (LUT) sampling the disparity range of $[d_{min}, d_{max}]$. For simplicity, most methods usually sample disparity at integral intervals only. Then the mapping of a label $\ell \in \boldsymbol{\mathcal{L}}$ to a disparity d_ℓ is

$$d_\ell = \lfloor d_{min} \rfloor + \ell \quad \text{with} \quad \ell = 0,1,\cdots,\lceil d_{max} - d_{min} \rceil \tag{7.1}$$

For methods requiring sub-pixel disparity precision, the mapping becomes

$$d_\ell = \lfloor d_{min} \rfloor + \Delta_d \cdot \ell \quad \text{with} \quad \ell = 0,1,\cdots,\left\lceil \frac{d_{max} - d_{min}}{\Delta_d} \right\rceil \tag{7.2}$$

where $\Delta_d < 1.0$ is the sub-pixel disparity sampling interval.

Using the above label to disparity mapping, the binocular stereo matching can be solved by optimizing the label assignment for each pixel, from which the disparity map as the final result is induced.

7.2 Data Cost Computation

Based on the binocular stereo vision fundamentals and label to disparity mapping explained above, we can see, for a pixel (x, y) in the left image $\boldsymbol{I}_{left}$, its matching point corresponding to a hypothesized label ℓ in the right image $\boldsymbol{I}_{right}$ is defined as $(x + d_\ell, y)$. Therefore, to evaluate the label space image $c(x, y, \ell)$ (or the so-called ***disparity space image*** specific to stereo matching), a pixel-based photo consistency function based on the intensity difference between each such matching point pair is used. The most commonly used functions include:

SD (Squared Difference),

$$c(x, y, \ell) = \left(\boldsymbol{I}_{left}(x, y) - \boldsymbol{I}_{right}(x + d_\ell, y)\right)^2 \tag{7.3}$$

and AD (Absolute Difference),

$$c(x, y, \ell) = \left|\boldsymbol{I}_{left}(x, y) - \boldsymbol{I}_{right}(x + d_\ell, y)\right| \tag{7.4}$$

For both of them, the Birchfield-Tomasi intensity difference measure [140] can be used to lower the sensitivity to image sampling.

Another popular one is ZNCC (Zero-mean Normalized Cross-Correlation) measure η, by which

$$c(x,y,\ell) = \eta(\boldsymbol{I}_{left}(x,y), \boldsymbol{I}_{right}(x+d_\ell, y)$$

$$\frac{\sum_{i=-W}^{W}\sum_{j=-W}^{W}(\boldsymbol{I}_{left}(x+i,y+j)-\bar{\boldsymbol{I}}_{left}(x,y))\cdot(\boldsymbol{I}_{right}(x+d_\ell+i,y+j)-\bar{\boldsymbol{I}}_{right}(x+d_\ell,y))}{\sqrt{\sum_{i=-W}^{W}\sum_{j=-W}^{W}(\boldsymbol{I}_{left}(x+i,y+j)-\bar{\boldsymbol{I}}_{left}(x,y))^2}\cdot\sqrt{\sum_{i=-W}^{W}\sum_{j=-W}^{W}(\boldsymbol{I}_{right}(x+d_\ell+i,y+j)-\bar{\boldsymbol{I}}_{right}(x+d_\ell,y))^2}} \quad (7.5)$$

where W is the half size of ZNCC window and $\bar{\boldsymbol{I}}_{left}(x,y)$ or $\bar{\boldsymbol{I}}_{right}(x,y)$ is the mean intensity of the pixels in the window centered at (x,y) in image $\boldsymbol{I}_{left}$ or $\boldsymbol{I}_{right}$. In this thesis, we have found ZNCC measure gives better results in most experiments of different applications.

When evaluating the disparity space images, for pixels close to the image borders, only partial disparity range can be evaluated since for larger disparities the corresponding matching pixels may be outside of the image. Incomplete evaluations could result in erroneous optimal disparity decision in the following optimization step for the involved pixels, which may further propagate to other pixels if global optimization is used. For example, as shown in Figure 7.1, without handling the issue related to the data cost of border pixels, the disparity results of [115] suffer from errors in the border areas.

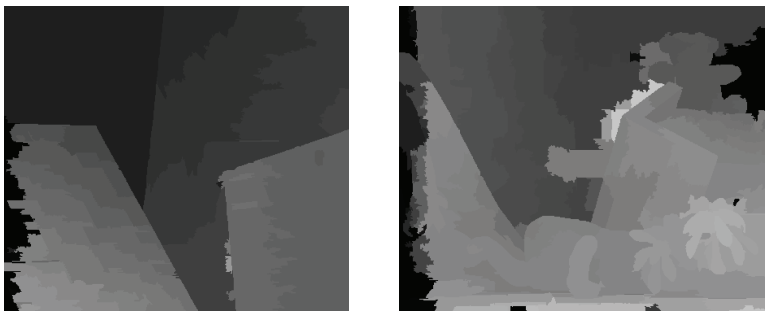


Figure 7.1: Disparity errors without handling border pixel datacost issue.

To address this issue, we propose a new approach based on the traditional occlusion filling heuristic, which has been previously used in [82] as a post-

processing technique to estimate unrecoverable disparities of occluded pixels. In particular, for a specific disparity, the boundary pixels matched to pixels outside of the image will leave "holes" in the corresponding disparity space image. For each pixel in the holes in the left image, the data cost value of the leftmost non-hole pixel on its right side (on the same scanline) is assigned as its data cost for the disparity in question, and a similar reasoning applies to holes in the right image. In this way, the missing data cost evaluations can be approximated. However, this approach works best when the border pixels have horizontally smooth disparity distribution.

7.3 Data Cost Aggregation

The most often used data cost aggregation schemes in binocular stereo matching are the window-based ones which average the costs of the neighboring pixels in a window positioned based on the pixel in question. To address the problem that symmetric and fix-sized aggregation windows straddling depth discontinuities may corrupt the data cost of corresponding pixels, more computationally intensive variants of window-based cost aggregation such as shiftable window [141] or size-adaptive window [142] have also been proposed.

For our region-tree based binocular stereo matching, each region itself is used as the corresponding aggregation support region for all the pixels in that region. That is, for a region r_i in the corresponding region-tree $\boldsymbol{\mathcal{T}}(\boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{E}})$, $\mathcal{D}(r_i, \ell_{r_i})$ can be defined as

$$\mathcal{D}(r_i, \ell_{r_i}) = \frac{\sum_{(x,y)\in r_i} c(x,y,\ell)}{N_{r_i}} \tag{7.6}$$

Based on the assumption that all the pixels in a region has similar disparities, the above depth discontinuity straddling problem can be ameliorated . Furthermore, the irregular aggregation regions are more adaptive compared to using regular windows.

7.4 Disparity Computation and Optimization

After the data cost computation, to apply our framework to binocular stereo matching, the smoothness costs are evaluated.

Specific to the binocular stereo matching, for each edge $e_{(i,j)} \in \mathcal{E}$, the smoothness cost function $\mathcal{S}\left(r_i, r_j, \ell_{r_i}, \ell_{r_j}\right)$ measures the assignment of two adjacent regions r_i and r_j with disparities $d_{\ell_{r_i}}$ and $d_{\ell_{r_j}}$, respectively. It can be any arbitrary non-decreasing function of absolute disparity difference $\left|d_{\ell_{r_i}} - d_{\ell_{r_j}}\right|$. In the current implementation, we define it as

$$\mathcal{S}\left(r_i, r_j, \ell_{r_i}, \ell_{r_j}\right) = \frac{\left|d_{\ell_{r_i}} - d_{\ell_{r_j}}\right|}{\left(w_{e_{(i,j)}} + \epsilon\right)} \tag{7.7}$$

where ϵ is a small positive value for avoiding division-by-zero and $w_{e_{(i,j)}}$ is the region-tree edge weight as defined in (6.4).

In this way, the general labeling energy function (6.1) has been specialized for binocular stereo matching. Optimizing such an energy function defined over the whole region-tree using DP is rather straightforward as explained in Section 6.5. From the labeling solution, the corresponding disparity map can be deduced. That is, for each region r_i assigned with a label ℓ_{r_i}, all the pixels $(x, y) \in r_i$ will have the same disparity $\lfloor d_{min} \rfloor + \Delta_d \cdot \ell_{r_i}$,where $\Delta_d \leq 1.0$ is the corresponding disparity range sampling interval.

7.5 Disparity Refinement

7.5.1 Cross Checking based Occlusion Handling

Since no explicitly visibility/uniqueness constraint is enforced during the above region-tree based disparity optimization, there may be possible disparity errors due to occlusions in the regions which are visible only in one image. To address this issue, the symmetric cross-checking technique is applied as a post processing step.

Specifically, two disparity maps are generated each for the left and right images. Then the occlusion reasoning and correction is done by symmetrically cross-checking for violations between these two disparity maps based on the weak disparity consistency constraint proposed by Gong and Yang [143]. That is, suppose a pixel (x, y) in $\boldsymbol{I}_{left}$ is assigned with a disparity of d_1 and the disparity

of its corresponding pixel $(x + d_1, y)$ in $\boldsymbol{I}_{right}$ is d_2. By reasonably assuming all the disparity errors are caused by occlusions, for an unoccluded pixel, d_1 should be equal or very similar to d_2. However, if d_1 is larger than d_2, then the pixel is most likely occluded since otherwise it should be matched to a pixel with a larger disparity d_2 instead, i.e. closer to the camera. Then the detected occlusions can be corrected by hole-filling [82].

7.5.2 C2F Region-tree based Sub-pixel Disparity Refinement

In our region-tree based labeling framework, all the pixels in a region are assumed to have the same label. This simple region-label distribution model prefers scene with fronto-parallel planes, especially for larger regions. Therefore, the disparity map obtained as above may not be able to capture gradual pixel-level disparity changes such as that observed for slanted planes. To address this issue, the C2F region-tree approach introduced in the last chapter is resorted.

In particular, a multiple-level C2F region tree is built using different segmentation granularities, with the finest level regions containing only single pixels. For the coarsest level region tree, the matching is done as above using a globally sampled disparity hypothesis set (label set) which is the same for all the regions. Then for any non-coarsest level, for each region, its disparity range of interest is defined based on the disparity result of its containing region in the last level. A locally sampled disparity hypothesis set is created and used for each region in the optimization of non-coarsest level. In this way, incremental spatial and disparity resolution improvement can be achieved, resulting in smoother disparity result.

7.6 Experiments and Evaluations

7.6.1 Middlebury Dataset Benchmarking

Four datasets, Tsukuba, Venus, Teddy and Cones in the new Middlebury stereo evaluation scheme [122] are used to quantitatively evaluate the algorithm. Specifically, the stereo matching quality is measured by the percentages of the number of bad pixels (whose absolute disparity error is larger than a threshold) to

the total number of pixels in the whole image, in un-occluded regions and in discontinuity regions. In all the experiments, we use a two-level C2F region tree with the minimal region size at the coarse level set as 10 pixels and the fine level segmentation granularity set to single pixel. The ZNCC window size used is 3×3.

The smoothness energy term weight λ is automatically calculated for a specific dataset in a similar way (adapted to region-based) as in [119] based on the Kullback-Leibler divergence $\overline{KL}$. That is, $\lambda = \rho\overline{KL}$. As can be seen in the following chapters, for different applications, the smoothness weight scale ρ can be different. Specific to binocular stereo matching, we set $\rho = 4/3$. Please note that this constant is dependent on the data and smoothness cost functions. When the data or smoothness cost function is changed for different applications, ρ should be changed accordingly.

Figure 7.2 shows the dense disparity maps produced by our algorithm and the ground truths for the above mentioned datasets. For comparison, the results of the pixel-tree based algorithm [115] are also included. It can be seen that the incorporation of the region-based approach using the region tree instead of using the pixel tree does improve the matching accuracy for most datasets, especially the more challenging Teddy and Cones datasets due to their fairly large disparity ranges. Shown in Table 7.1 is the ranking of the overall quality of our results. It can be seen that our algorithm performs quite well and is ranked among the state-of-the-art algorithms.

As for efficiency, our method is comparable to the two top-ranking ones, which are also region-based in which the mean-shift based segmentation is the most time-consuming step. Specifically, for the teddy dataset, with cross-checking and two level of C2F region-trees, the processing time is about 35s while the current No.1 method [127] requires 20s.

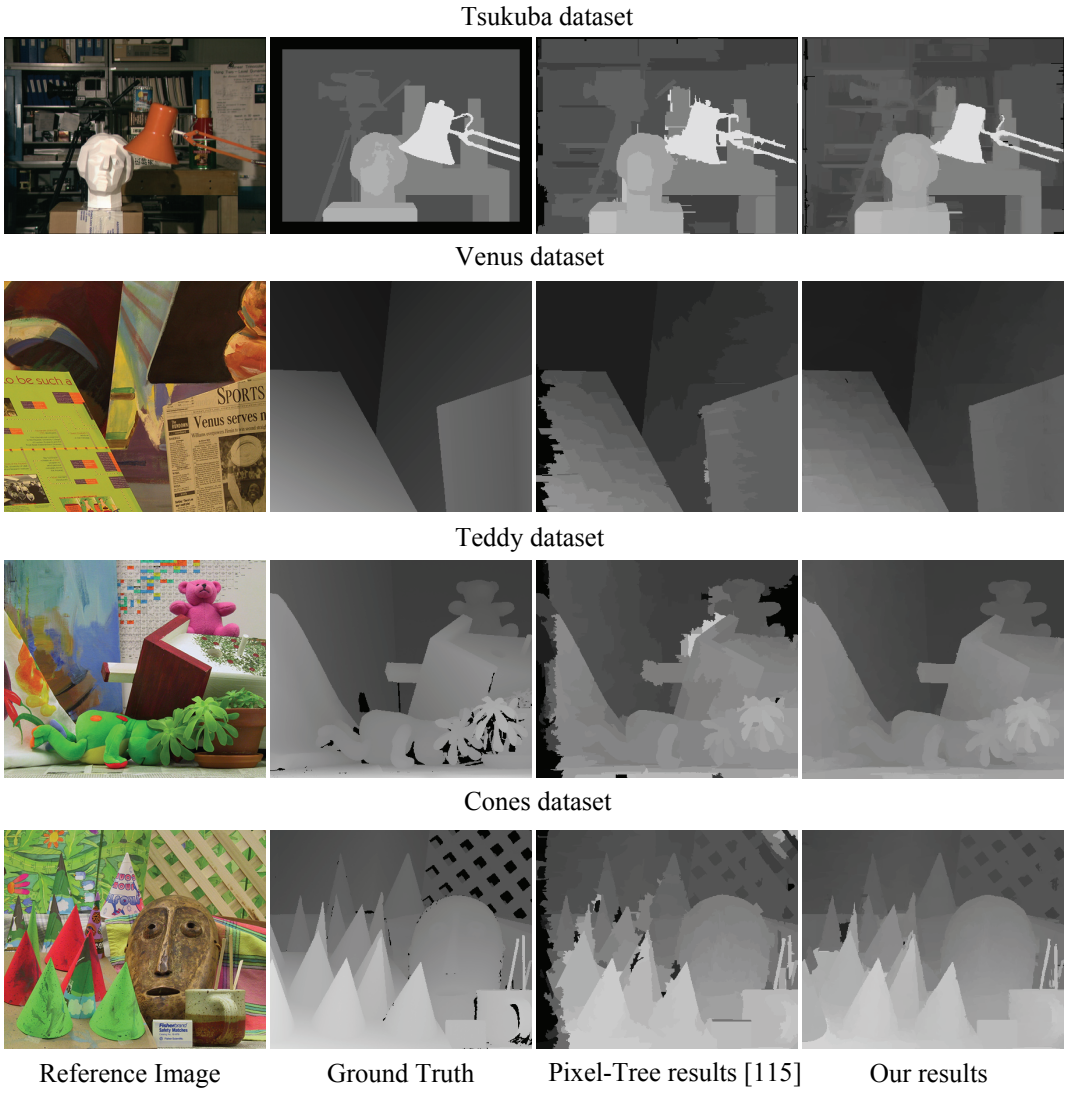


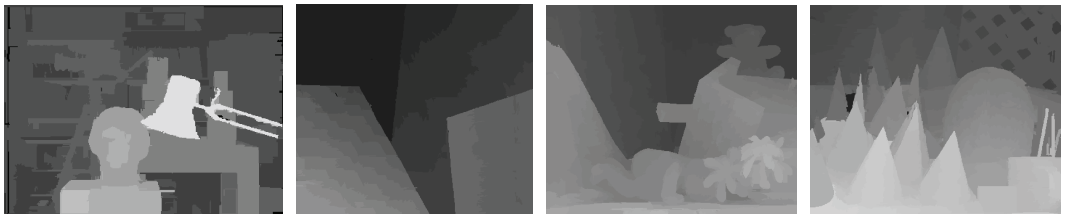
Figure 7.2: Results on Middlebury stereo benchmark dataset. Results of the pixel tree algorithm [115] is also included for comparison.

To illustrate the improvement obtained by using the C2F region tree based sub-pixel disparity refinement as explained in Section 7.5.2, Figure 7.3 shows the comparison of the dense disparity maps produced by the coarse level and the fine level region trees. As can be easily seen, the C2F region tree based refinement does produce smoother disparity maps. Through C2F refinement, the sub-pixel accuracy benchmark ranking (with the bad pixel threshold set as 0.5 pixel) is improved to 24.3 from 38.2.

Error Threshold = 1 (Error Threshold...)		Sort by nonocc			Sort by all			Sort by disc						Average Percent Bad Pixels
Algorithm	Avg. Rank	Tsukuba ground truth nonocc	Tsukuba ground truth all	Tsukuba ground truth disc	Venus ground truth nonocc	Venus ground truth all	Venus ground truth disc	Teddy ground truth nonocc	Teddy ground truth all	Teddy ground truth disc	Cones ground truth nonocc	Cones ground truth all	Cones ground truth disc	
AdaptingBP [17]	3.8	1.11 8	1.37 5	5.79 9	**0.10** 1	0.21 3	1.44 2	4.22 3	7.06 3	11.8 4	**2.48** 1	7.92 5	7.32 2	4.23
CoopRegion [41]	4.0	**0.87** 1	**1.16** 1	**4.61** 1	0.11 2	0.21 2	1.54 4	5.16 10	8.31 5	13.0 7	2.79 5	7.18 2	8.01 8	4.41
DoubleBP [35]	5.4	0.88 3	1.29 2	4.76 3	0.13 5	0.45 11	1.87 8	3.53 2	8.30 4	**9.63** 1	2.90 6	8.78 15	7.79 5	4.19
OutlierConf [42]	6.1	0.88 2	1.43 7	4.74 2	0.18 10	0.26 6	2.40 12	5.01 6	9.12 7	12.8 6	2.78 4	8.57 10	**6.99** 1	4.60
SubPixDoubleBP [30]	8.3	1.24 15	1.76 16	5.98 10	0.12 4	0.46 12	1.74 6	**3.45** 1	8.38 6	10.0 2	2.93 8	8.73 13	7.91 7	4.39
WarpMat [55]	9.9	1.16 9	1.35 4	6.04 11	0.18 11	0.24 5	2.44 13	5.02 7	9.30 8	13.0 9	3.49 15	8.47 9	9.01 18	4.98
Undr+OvrSeg [48]	12.8	1.89 31	2.22 29	7.22 26	0.11 3	0.22 4	**1.34** 1	6.51 15	9.98 9	16.4 16	2.92 7	8.00 6	7.90 6	5.39
GC+SegmBorder [57]	13.7	1.47 25	1.82 18	7.86 29	0.19 12	0.31 7	2.44 13	4.25 4	**5.55** 1	10.9 3	4.99 38	**5.78** 1	8.66 13	4.52
AdaptOvrSegBP [33]	14.5	1.69 28	2.04 26	5.64 8	0.14 6	**0.20** 1	1.47 3	7.04 22	11.1 12	16.4 18	3.60 18	8.96 17	8.84 15	5.59
SymBP+occ [7]	16.3	0.97 6	1.75 15	5.09 6	0.16 8	0.33 8	2.19 10	6.47 14	10.7 10	17.0 22	4.79 35	10.7 31	10.9 30	5.92
PlaneFitBP [32]	16.3	0.97 7	1.83 19	5.26 7	0.17 9	0.51 14	1.71 5	6.65 17	12.1 19	14.7 10	4.17 30	10.7 30	10.6 28	5.78
AdaptDispCalib [36]	17.7	1.19 12	1.42 6	6.15 13	0.23 14	0.34 9	2.50 16	7.80 28	13.6 28	17.3 27	3.62 19	9.33 18	9.72 22	6.10
YOUR METHOD	18.2	2.07 36	2.48 30	9.76 39	0.14 7	0.35 10	1.80 7	6.60 16	11.0 11	16.3 15	3.45 14	8.76 14	9.10 19	5.99
C-SemiGlob [19]	18.3	2.61 41	3.29 36	9.89 40	0.25 17	0.57 16	3.24 21	5.14 9	11.8 13	13.0 7	2.77 3	8.35 8	8.20 9	5.76

Table 7.1: Middlebury stereo benchmark ranking
(with the bad pixel threshold set as 1 pixel).

Coarse level results



Fine level results

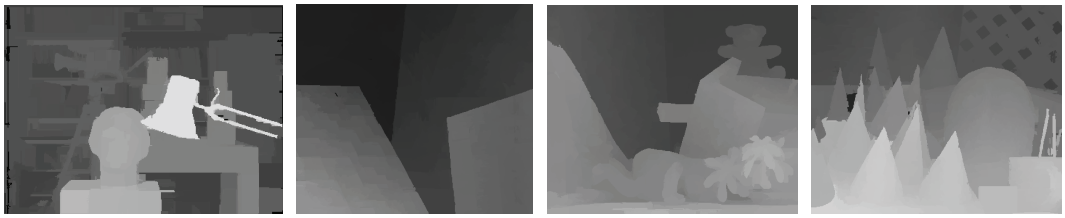


Figure 7.3: Coarse and fine segmentation level stereo result comparison.

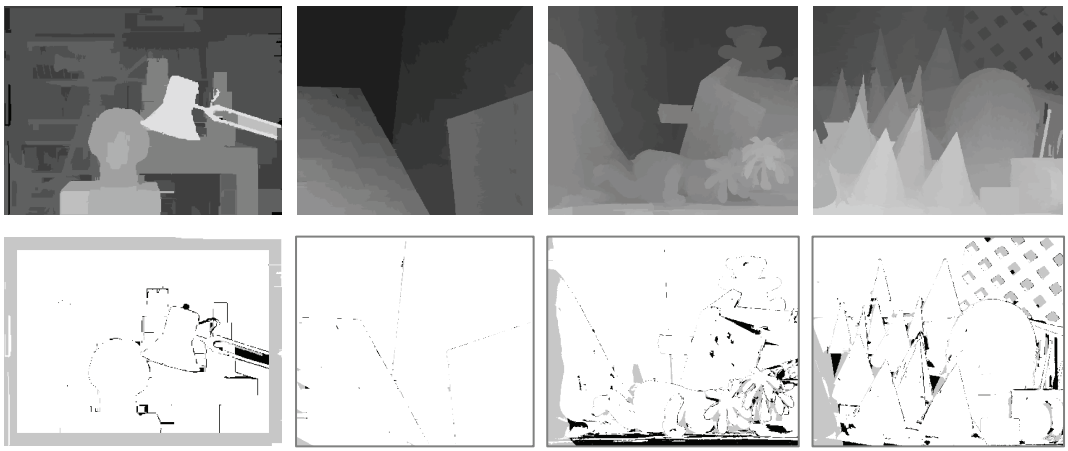
7.6.2 Performance Comparison of Using Different Optimization Methods

Our region-tree based labeling framework can also use other optimization methods such as belief propagation or graph cuts, which are capable of handling non-grid-based image representations. To investigate the performance difference of using different optimization methods, we compare DP with graph cuts. That is, instead of DP, we use graph cuts to optimize the stereo matching energy function

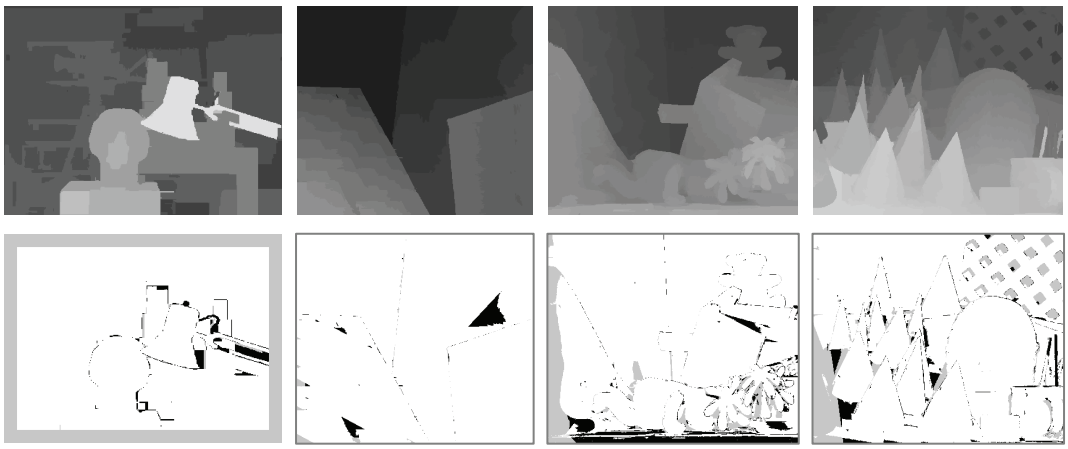
defined on the region-tree and compare the difference of the resulting disparity map with that of using DP. Please note that the same data and smoothness functions are used in the comparisons and the only difference is the optimization method used. We use the expansion based graph cuts implementation from [110], which provides a generalized coding interface for defining neighborhood influence system of labeling primitives. Furthermore, unlike DP, since graph cuts can also handle graph-based influence system, we also investigate the performance of using graph cuts on region adjacency graph, which is obtained in the image segmentation step as explained in Section 6.4, for comparison purposes. For simplicity and better performance, only the single coarse level segmentation is used in the graph cuts based experiments.

Specifically, the top two rows of Figure 7.4 are disparity results using graph cuts on region-tree. For the Venus, Teddy and Cones datasets, the results are very close to the coarse level results using DP as shown in Figure 7.3, with the largest quality measure improvement for the Teddy dataset by only 0.02%. The only exception is the Tsukuba dataset, for which the non-occlusion error rate increases from 2.07% for DP to 2.19% for graph cuts. Therefore, we can see that for a tree structure, DP has a very similar performance to that of graph cuts in minimizing the energy functions. However, DP optimization is significantly faster than graph cuts optimization. For the Teddy or Cones datasets, the speedup can be more than 10 times.

On the other hand, the lower two rows of Figure 7.4 show the results of using graph cuts on region adjacency graph instead of region tree. As can be seen from the "bad pixel" maps shown, for all four dataset, the results are worse than the ones using region tree, with the results of the Venus dataset being the worst. One possible reason could be the fact that the region-tree minimizes the number of edge crossing disparity discontinuities compared to a region graph and hence, avoids unnecessary and error-incurring smoothness enforcement on them. Therefore, using a region-tree does have a unique advantage than using a region graph.



Disparity results and corresponding "bad pixel" maps using graph cuts and region-tree



Disparity results and corresponding "bad pixel" maps using graph cuts and region-graph

Figure 7.4: Comparison of using graph cuts on region-tree and on region-graph.

7.6.3 Performance Sensitivity to Segmentation Granularity

We further investigate the performance sensitivity of our stereo algorithm to segmentation granularity. In Figure 7.5, profiling curves of the disparity error rate with respect to the minimum region size limit, which varies from 5 pixels to 80 pixels, in non-occluded, all and discontinuity areas for the above 4 Middlebury datasets are shown. Two examples of the tested segmentation granularities for the Teddy dataset are shown in the bottom row of the same figure. From the curves, we can see that the general performance is robust to segmentation granularity, though for some datasets with dominant slanted surfaces such the Venus dataset, a larger region size appears to incur a slightly higher error rate. The main reason is because our current formulation assumes that all the pixels in a region have the

same label (disparity). Therefore using larger regions will increase errors when disparity changes occur within a region, in which case using smaller regions is expected to produce better results.

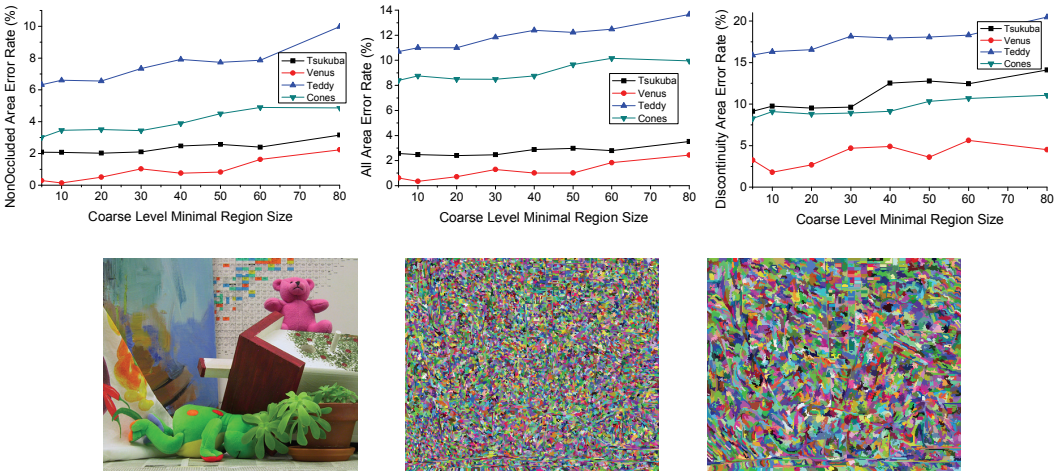


Figure 7.5: Profiling curves of performance to segmentation granularities on four Middlebury stereo datasets. Bottom row shows two example segmentation granularities for the Teddy dataset.

7.6.4 More Experiments

In addition to using the Middlebury binocular stereo matching benchmark datasets, the proposed algorithm is also extensively tested with the other challenging datasets with better and similar performance obtained.

Figure 7.6: Result of Flower & Lady dataset from [144]. The left two are the input images. The right is the disparity result.

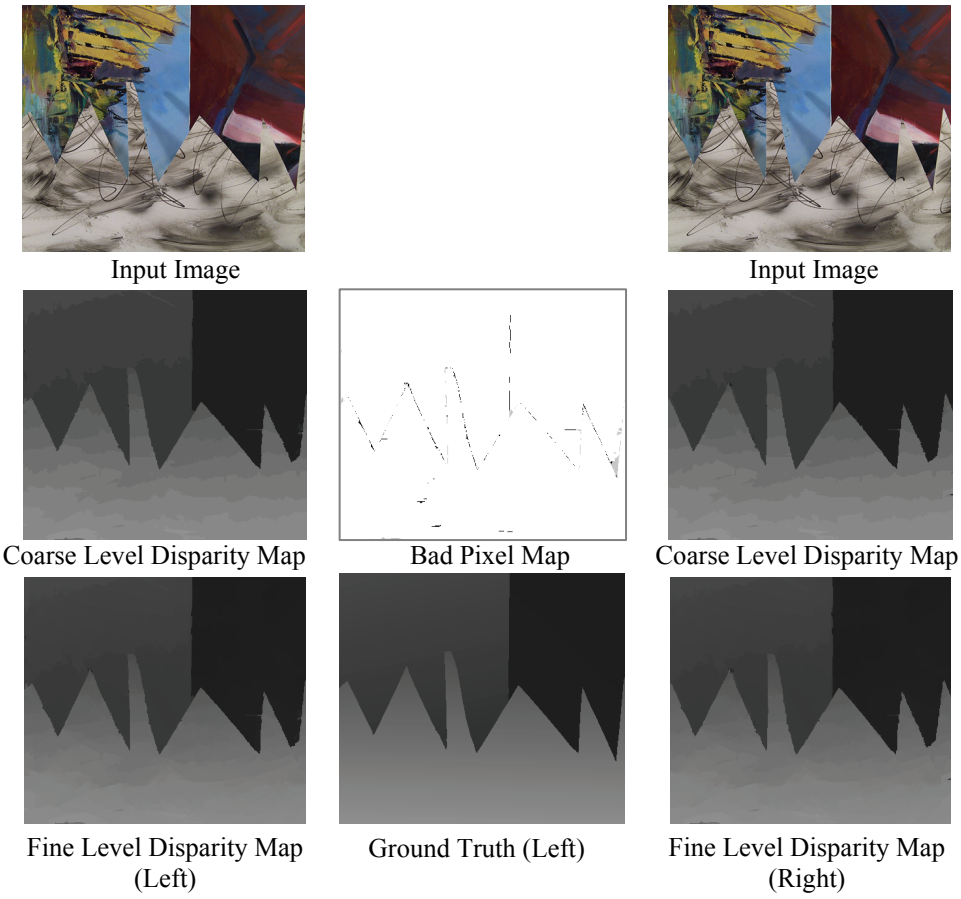


Figure 7.7: Results of the Middlebury Sawtooth dataset.

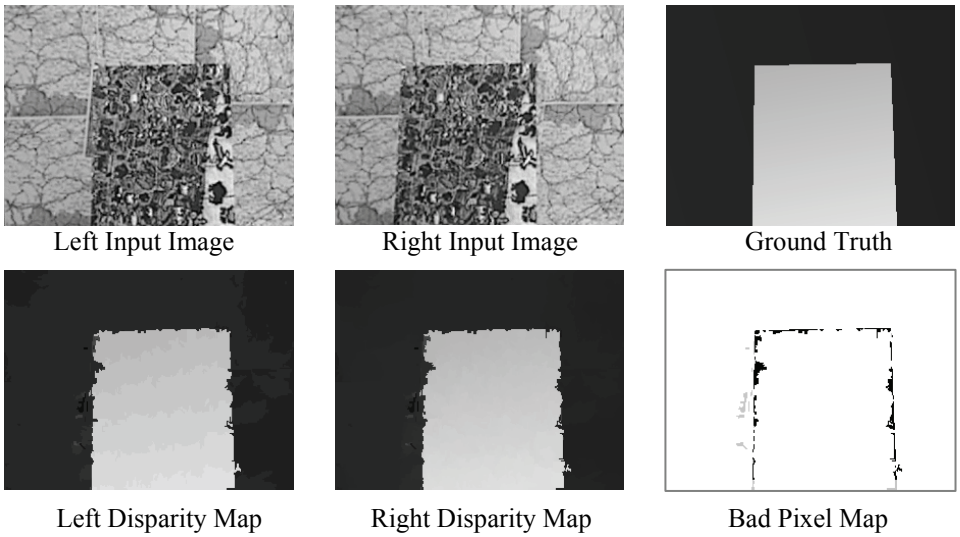


Figure 7.8: Result of the Middlebury Map dataset. Similar to the observations reported in many other region-based stereo papers, our method also gives relatively less satisfactory result for the Map dataset due to the violation of the disparity discontinuity assumption.

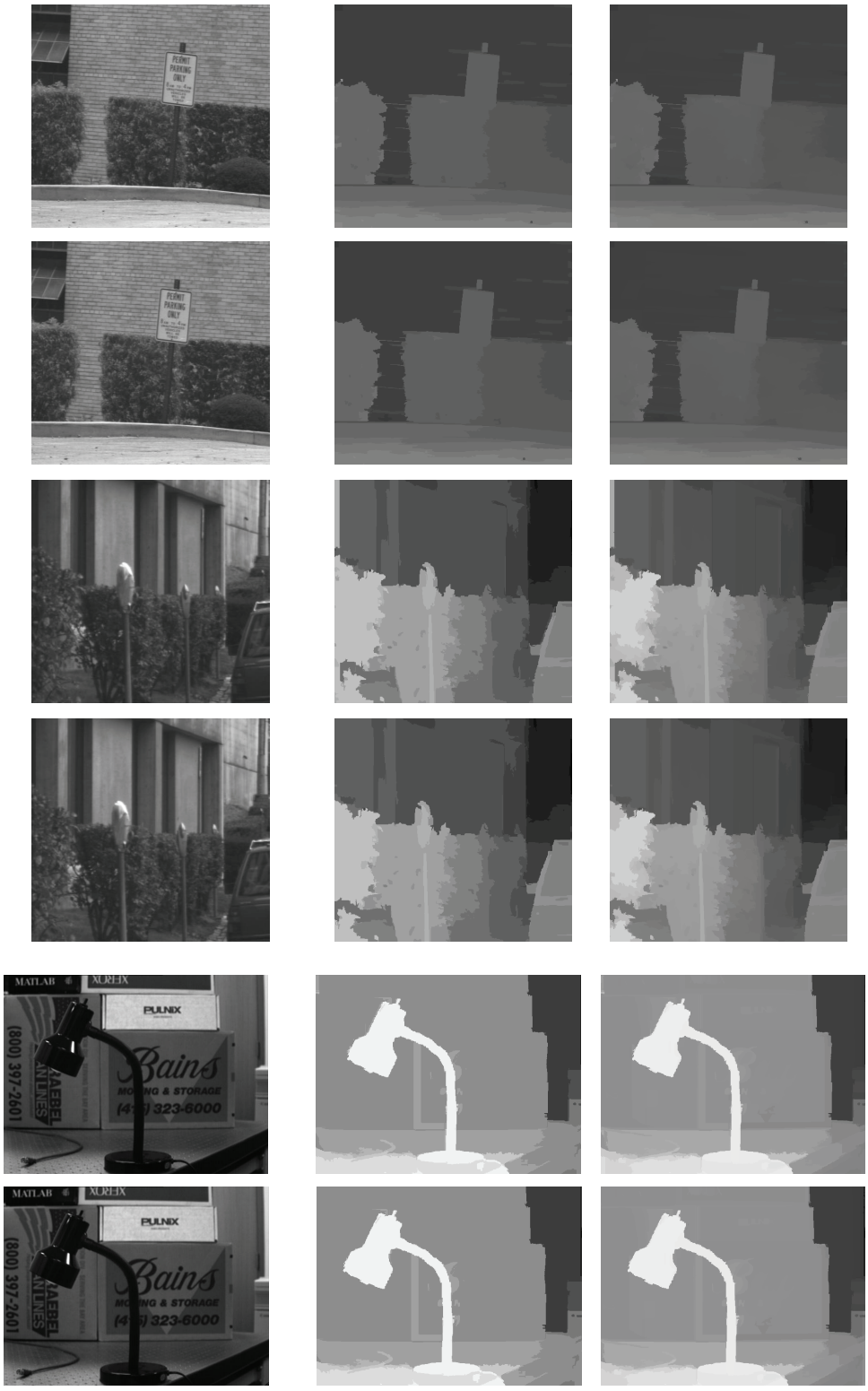
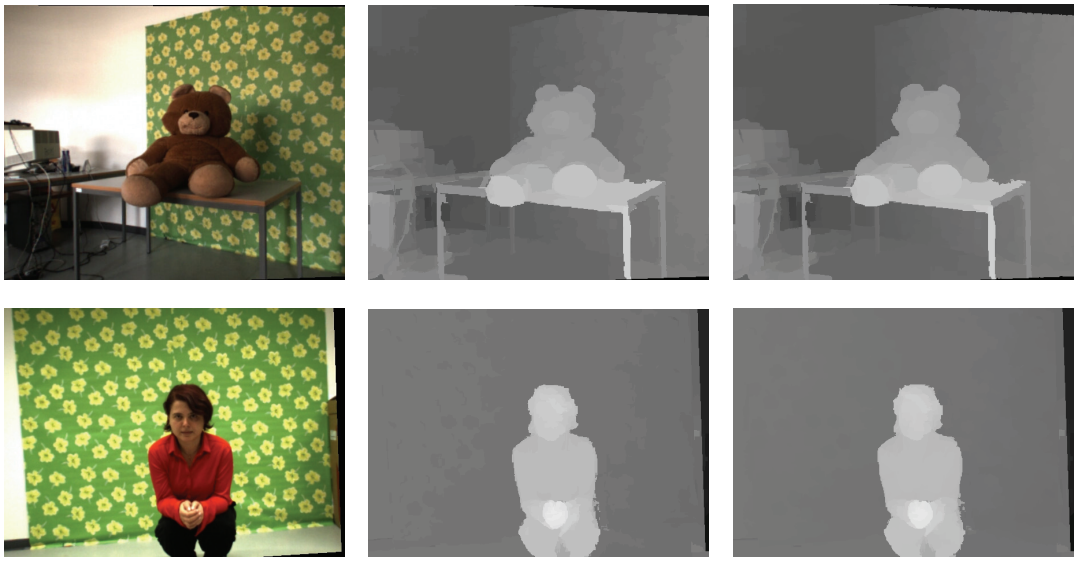


Figure 7.9: Results of the Stanford dataset from [145].



Input Images | Coarse Level Disparity Maps | Fine Level Disparity Maps

Figure 7.10: Results of datasets from [128, 146].

Summary

In this chapter, our new region-tree based labeling framework is applied to binocular stereo matching. The promising results in the de facto benchmarking tests and various real-world datasets show its effectiveness and uniqueness. This work forms the foundation of the new video depth based FVV scene reconstruction to be elaborated in Chapter 9.

Chapter 8
Region-Tree Based Optical Flow Estimation

As another important application of the new region tree labeling framework, this chapter illustrates how to use it in optical flow estimation.

Similar to binocular stereo vision, optical flow estimation or motion analysis is also an active topic in computer vision research with intensive research activities over the last two decades. Formally speaking, its main task is to automatically reconstruct a dense field of displacement vectors encoding scene object motion or camera motion between corresponding pixels in consecutive images which capture a dynamic scene using a single camera. Such 2D displacement vectors are commonly known as the ***optical flow***. Accurate and efficient optical flow estimation is very important to applications such as motion detection [147] and video compression [148]. Specific to this thesis, we are interested in it because it is indispensible to recovering temporally consistent dense depth information for FVV scene reconstruction, which is very desirable for high quality FVV rendering.

Although optical flow estimation is to establish the dense correspondences between two images, which is similar to the dense binocular stereo matching, solving it is much more difficult. The reason is that the probability of mismatch increases due to the change from the 1D search range in binocular stereo matching to a much larger 2D search area in optical flow estimation. Hence, the correspondence searching is much less constrained and more computational effort is needed.

In the following, a very brief review of relevant prior work on optical flow estimation is given for completeness, with more details on the work most relevant to our proposed method. For a more extensive review, the reader can refer to

some previous surveys such as [149, 150].

8.1 Prior Work

Similar to stereo vision, optical flow estimation is also based on the intensity constancy assumption between matching pixels. Existing methods can be roughly classified into two categories, the differential ones and the parametric ones.

8.1.1 Differential Algorithms

The differential algorithms assume that an image sequence is a differentiable intensity function $\boldsymbol{I}(x, y, t)$ of parameters of pixel coordinates (x, y) and time t. By approximating the intensity constancy equation between two matched points

$$\boldsymbol{I}(x, y, t) = \boldsymbol{I}(x + u, y + v, t + 1) \tag{8.1}$$

using the Taylor series expansion and then performing differencing, the well-known ***optical flow constraint equation***

$$\boldsymbol{I}_x(x, y) \cdot u + \boldsymbol{I}_y(x, y) \cdot v + \boldsymbol{I}_t(x, y) = 0 \tag{8.2}$$

is obtained, where $\boldsymbol{I}_x$, $\boldsymbol{I}_y$ and $\boldsymbol{I}_t$ denote the corresponding partial derivatives of $\boldsymbol{I}(x, y, t)$, with respect to *x*, *y*, and *t*, respectively. This constraint equation is the foundation of all differential motion analysis algorithms, but it is under-constrained and needs additional constraints on the unknowns u and v. Therefore, the prior knowledge of displacement smoothness between neighboring pixels has to be used.

Based on the specific strategy used for employing such smoothness constraint, the differential algorithms can be further classified into the local and global ones.

The local algorithms such as [151] assume that the displacement vectors within a window are all the same and solve the resultant equation system locally for each pixel's displacement (u, v). They are usually very fast, but have poor performance for pixels in un-textured image areas.

In contrast, the global algorithms such as [152-157] enforce the smoothness constraint globally by formulating the optical flow estimation as an energy

minimization problem similar to what's done in stereo vision. Among all the approaches proposed to optimizing such an energy functional so far, the variational-calculus based computation framework has been the top-performing one and gained much popularity in the research community [154-157]. However, such optimization schemes based on continuous mathematics often suffer the problems of over-smoothing due to their restricted convex flow smoothness regularizations. Also due to their gradient-descent based minimization, the optimization could be trapped in local minima, which result in poor performance for sharp motion discontinuities and for large motion displacements [158].

8.1.2 Parametric Algorithms

Different from solving optical flow on a pixel-wise basis as in the differential algorithms, the parametric algorithms try to segment the images into a set of regions/layers in which the pixels undergo consistent motions which can be described by the parametric models. When appropriate motion models are used and the motion segmentation is accurate, this class of algorithms can usually give more accurate optical flow estimation since they can robustly constrain the motions in large image areas by very few parameters.

Representative works in this category include the work of Black and Jepson and Wang and Adelson [159,160]. In particular, Black and Jepson [159] first segment the reference image into regions of homogenous color, for each of which a variable order parametric motion model is estimated using an initial dense optical flow field. The number of parameters used for each region is determined by selecting the model resulting in minimal color registration error when warping the region into the second image using the model induced motion. The model parameters themselves are refined using area based regression techniques. Wang and Adelson [160] use an affine model to fit motions of a set of seed patches in the reference image from an optical flow field computed using a local method. After using K-means clustering, similar motions are identified as layer motions and blocks are assigned with the corresponding layer motion model. Iteratively, the algorithm updates the layer partition and layer motion model parameters until

convergence to get the final optical flow field.

However, the main drawbacks of parametric algorithms are in their strong reliance on the accuracy of image segmentation and motion layer extraction, which are themselves difficult problems in many situations.

8.2 Motivations

Similar to binocular stereo matching, most state-of-the-art optical flow algorithms formulate the optical flow estimation as an energy minimization problem based on the seminal work of Horn and Schunck [152]. However, although discrete optimization schemes such as graph-cuts, belief propagation and dynamic programming have gained dominant role in binocular stereo matching, their use is surprisingly rare in optical flow estimation, in which the variational-calculus based optimization scheme dominates. Then it is natural to ask the question if such discrete optimization schemes are also applicable to optical flow estimation which shares many commonalities with the stereo matching problem.

Considerable efforts [158-162] have been devoted recently in enabling discrete optimization schemes for optical flow estimation and promising results have been reported using the Middlebury quantitative evaluation and benchmarking datasets [163]. In general, most of them map the original optical flow problem into a labeling problem through the so-called ***displacement discretization*** and then adapt a well-known discrete optimization scheme such as graph-cuts or belief-propagation to find the best label assignment for all the labeling primitives such as pixels, regions or layers, from which the final optical flow field is induced by mapping the labels back to displacement vectors.

Based on whether or not the displacement discretization is directly done in the optical flow solution space, we can roughly classify such methods as direct [161, 164] and indirect [158, 162] discretization based methods. In direct discretization based methods, the labels are a direct discrete sampling of the final 2D displacement search space. That is, each label corresponds to a sampled 2D displacement vector. While in indirect discretization based methods, no discrete displacement sampling is done.

Specifically, in the fusion-based method [158], the pixel-wise label sets are locally created from a set of candidate solutions obtained by running different continuous flow algorithms or the same algorithm using different parameter settings. Then graph-cuts optimization is used to find the best label assignment for fusing candidate solutions. This procedure is equivalent to fusing the best partial results from different sources together to get a better complete one.

A similar fusion idea has also been investigated in [162]. The original minimization problem is formulated as a series of binary sub-problems, each of which is solved iteratively via the extended discrete graph-cuts with the alpha-expansion method that facilitates large energy minimization moves. Similar to [158], the set of candidate displacement vectors to be fused have to be provided by a standard continuous optical flow algorithm. Thus the success of both methods is largely dependent on the quality of the initial solution.

Another piece of related work is presented in [164], in which a framework based on a dynamic, discrete MRF is proposed for morphing images using a grid of control points. Discrete MRF optimization is used to iteratively and accumulatively optimize the displacement vectors at the control points from which the dense optical flow field is derived based on the influence functions.

The promising performance as demonstrated using the Middlebury benchmark database of all of the above mentioned recent attempts suggests that discrete optimization has a great potential in optical flow estimation.

In addition to adapting discrete optimization, efforts have also been made to use different image representation other than the traditional pixel grid for optical flow estimation. For example, some researchers [159, 165-167] try to take advantage of region based representations to address the problematic issues of texture-less regions and occlusions in optical flow estimation.

In particular, Zitnick et al. [165] propose a method to jointly segment consecutive frames into small regions of consistent size and to compute the optical flow based on statistical modeling of an image pair using constraints based on appearance and motion. Bidirectional motion is estimated using spatial

coherence and color similarity between segmented regions using a translational motion model.

Bleyer et al. [166] incorporate image segmentation and graph-cuts optimization to tackle the optical flow problem using a layer-based model. Each region is first assigned with an affine motion model from sparse correspondences. Motion layers are extracted by grouping regions with similar rigid motions and by identifying the dominant ones. Then as an indirect discretization based method, an energy function measuring the quality of label assignments of regions and pixels to layers is minimized using graph-cuts. Although very promising results have been obtained, the assumption of the existence of dominant rigid motion layers limits its applications.

Different from the above mentioned work [165, 166], Xu et al. [167] use the segmented color regions as soft constraints or regularization in the affine motion model in the classic variational optical flow framework instead of as matching primitives. To avoid over-regularization for non-rigid motions, a confidence map encoding the fitness of the affine region motion model is used.

Motivated by the promising performance of the above recent attempts in using discrete optimization or region based representation and also specifically needed for temporal consistent FVV scene reconstruction, in this chapter, we further apply our region-tree based framework for performing optical flow estimation as a labeling problem.

8.3 Implementation Details

8.3.1 Optical Flow Estimation Formulation as a Labeling Problem

As mentioned above, using discrete optimization to recover essentially continuous optical flow field $\boldsymbol{\mathcal{O}}_{t\to t+1}$ from image $\boldsymbol{I}_t$ to image $\boldsymbol{I}_{t+1}$ requires discretization, by which the continuous 2D displacement solution space is quantized and mapped to a discrete set of labels. In particular, we uniformly sample the corresponding displacement search ranges in both the horizontal and vertical directions each with a sampling interval. Then the labels are defined as the index into such a

displacement sample array. However, the number of labels required for sampling such a 2D search area with fine enough precision could be too large for efficient optimization.

We address this problem using the C2F region-tree representation presented in Chapter 6. In particular, multiple-level coarse-to-fine over-segmentations $\{S_l | l = 0, \ldots, L\}$ are applied to the reference image $\boldsymbol{I}_t$. For each segmentation level l, the over-segmented regions $\boldsymbol{\mathcal{R}}_l$ form the corresponding labeling primitive set $\boldsymbol{\mathcal{P}}_l$ on which the spanning region-tree $\boldsymbol{\mathcal{T}}_l$ is built as explained in Section 6.2.

At each segmentation level l, the label-to-displacement LUT for each region $r_l \in \boldsymbol{\mathcal{R}}_l$ is built by uniformly sample the corresponding region-dependent displacement search ranges $\left[u_{min}^{r_l}, u_{max}^{r_l}\right]$ and $\left[v_{min}^{r_l}, v_{max}^{r_l}\right]$ in both the horizontal and vertical directions with a sampling interval δ_l. At the coarsest level, the displacement search ranges of all regions are initialized globally with

$$u_{max}^{r_l} = v_{max}^{r_l} = -u_{min}^{r_l} = -v_{min}^{r_l} = \alpha \cdot \max(w, h) \tag{8.3}$$

wherein w and h are the width and height of the input images, respectively, and α is a positive constant. That is, we assume that the horizontal and vertical displacements each have upper bounds related to the image dimension.

Then with the result of the last coarser l level known, the search range of a region $r_k \in \boldsymbol{\mathcal{R}}_k$ at the finer level $k > l$ is set up based on its container region $r_l \in \boldsymbol{\mathcal{R}}_l$ at the last coarser segmentation level l. That is, suppose that the recovered displacement vector for region r_l is (u^{r_l}, v^{r_l}), the displacement search ranges for region $r_k \in \boldsymbol{\mathcal{R}}_k$ will be defined in a neighborhood around $\left(u^{r_l}, v^{r_l}\right)$ as $\left[u^{r_l} - \Delta_k, u^{r_l} + \Delta_k\right]$ and $\left[v^{r_l} - \Delta_k, v^{r_l} + \Delta_k\right]$, each of which is sampled with δ_k to set up the corresponding label-to-displacement LUT for region $r_k \in \boldsymbol{\mathcal{R}}_k$.

In addition to decreasing Δ_k and δ_k level by level, the incremental displacement refinement can also be achieved by performing multiple iterations at the finest level and using the last iteration results to define the displacement search ranges for the next iteration.

Therefore, by using region-dependent displacement search ranges and incrementally refining each region's displacement search ranges level by level or

iteration by iteration, just by using a small number of labels can achieve similar quality of sampling to continuous methods, resulting in better efficiency and accuracy. In this way, the original optical flow estimation can be recovered by solving multiple labeling problems level by level using our region tree framework.

8.3.2 Initial Search Range Probation

As mentioned in the last section, we use an upper bound based on the image dimension using Equation (8.3) to initialize the displacement search range at the coarsest segmentation level. Since the optical flow directions and magnitude can be arbitrary, we have to use a reasonably large α value to safely capture the full range. However, due to the limitation of memory and efficiency considerations, the number of labels used has to be limited. Therefore for large images, the sampling resolution may not be fine enough for accurate displacement estimation. Then the corresponding errors will be propagated to the next level and cannot be removed.

To address this issue, we further take advantage of the image pyramid based on multiple resolution strategy [81], which is often used in many continuous optical flow methods. In particular, for large images, we first apply our proposed method to their half-size down-sampled images and recover the displacement ranges $[u'_{min}, u'_{max}]$ and $[v'_{min}, v'_{max}]$. Since for smaller search ranges, using the same number of labels enables using a smaller sampling interval, the result usually contains less errors. Then we apply our proposed method again w.r.t. the original images using $[2u'_{min}, 2u'_{max}]$ and $[2v'_{min}, 2v'_{max}]$ as the initial search ranges. If necessary, more pyramid levels can be used.

8.3.3 Region-tree based Displacement Computation

Suppose at segmentation level *l*, the region-tree $\boldsymbol{\mathcal{T}}_l$ in question is defined on region node set $\boldsymbol{\mathcal{R}}_l$ with edge set $\boldsymbol{\mathcal{E}}_l$. Each edge $e_{(i,j)} \in \boldsymbol{\mathcal{E}}_l$ corresponds to a link between two adjacent regions $r_i \in \boldsymbol{\mathcal{R}}_l$ and $r_j \in \boldsymbol{\mathcal{R}}_l$. Each region r_i has $\boldsymbol{N}_{r_i}$ pixels $(x, y) \in r_i$ and is assigned with a label $\boldsymbol{L}(r_i) \in \mathcal{L}$ after optimization, which

corresponds to the 2D displacement vector $\left(u_{\boldsymbol{L}(r_i)}, v_{\boldsymbol{L}(r_i)}\right)$. Then as explained above, our algorithm estimates the optical flow by repeatedly optimizing the corresponding energy function (6.1) on multiple-level region trees level by level starting from the coarsest one.

Based on the label to displacement mapping explained above, for a pixel (x, y) in image $\boldsymbol{I}_t$, its corresponding matching point to a hypothesized label ℓ in image $\boldsymbol{I}_{t+1}$ is defined as $(x + u_\ell, y + v_\ell)$. To evaluate the label space image $c(x, y, \ell)$ (or the so-called ***displacement space image*** specific to optical flow estimation) which measures the pixel intensity similarity between pixels $\boldsymbol{I}_t(x, y)$ and $\boldsymbol{I}_{t+1}(x + u_\ell, y + v_\ell)$ as is done in stereo matching. In the current implementation, we also use the ZNCC measure as defined in (7.5).

As for data cost aggregation, the same scheme as the one described in Section 7.3 is used.

Furthermore, to evaluate the smoothness cost of assigning two linked regions r_i and r_j with displacement vectors $\left(u_{\boldsymbol{L}(r_i)}, v_{\boldsymbol{L}(r_i)}\right)$ and $\left(u_{\boldsymbol{L}(r_j)}, v_{\boldsymbol{L}(r_j)}\right)$, respectively, the following smoothness cost function is used

$$\boldsymbol{S}\left(\boldsymbol{L}(r_i), \boldsymbol{L}(r_j)\right) = \left|u_{\boldsymbol{L}(r_i)} - u_{\boldsymbol{L}(r_j)}\right| + \left|v_{\boldsymbol{L}(r_i)} - v_{\boldsymbol{L}(r_j)}\right| \tag{8.4}$$

It is different from the one used in stereo matching, which is more adaptive by including the edge weight between two involved regions. However, in our experiments, we have found that the above smoothness cost function performs slightly better for optical flow estimation. A possible reason might be due to the different smoothness characteristics of depth maps and optical flow field.

Then optimizing the corresponding energy function using DP for recovering the optical flow field for each segmentation level is done in a similar manner as explained in Section 6.5.

8.3.4 Optical Flow Filed Refinement

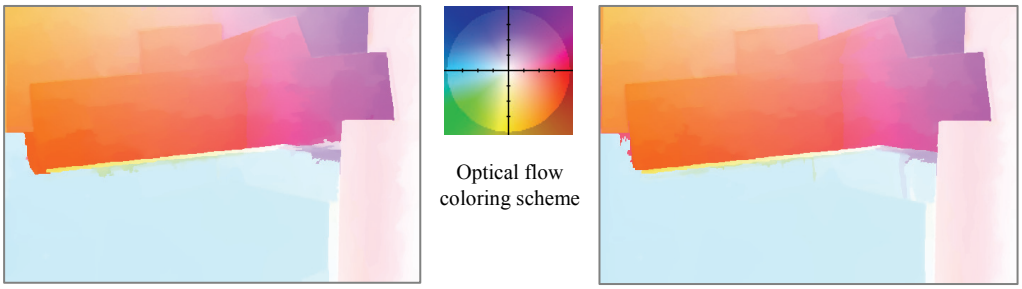
(a) Cross Checking based Occlusion Handling

A similar symmetric cross-checking technique introduced in Section 7.5.1 is also used for correcting optical flow errors due to occlusions. In particular, two

optical flows are estimated for images $\boldsymbol{I}_t$ and $\boldsymbol{I}_{t+1}$ respectively. Then occlusion reasoning is done by symmetrically cross-checking for consistency violations at the pixel level between these two optical flow fields.

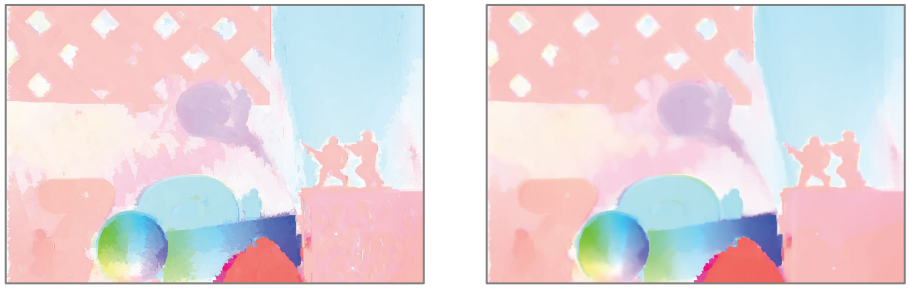
Specifically for a pixel $(x, y) \in \boldsymbol{I}_t$ with recovered optical flow vector (u, v), if its correspondence $(x + u, y + v) \in \boldsymbol{I}_{t+1}$ has optical flow vector of (u', v') and the consistency measure $|u - u'| + |v - v'|$ is greater than a preset threshold β, then pixel (x, y) will be flagged as an inconsistent pixel. For each region in the finest segmentation level, if over half of its pixels are flagged as inconsistent, the region will be flagged as occluded.

After all the occluded regions are flagged, a new DP optimization pass is done without applying the data and smoothness cost penalties to links involving an occluded region so that a larger motion change is made possible at such links. During the bottom-up DP evaluation traversal as explained in Section 6.5, an occluded region node will behave as a "pass-through" node, while during the top-down DP decision making traversal, an occluded region node will be assigned with its parent node label. This is similar to using neighbor information as is done in the traditional hole-filling approaches in stereo vision. But the difference is that the chosen neighbor is not necessarily found in the spatial domain, but in the region-tree domain in which the parent-child link is assumed to connect regions with similar attributes based on the region-tree construction procedure. Of course, we have to point out that since our region-tree spans over the whole image, at some points some edges must cross discontinuities, violating such an assumption. Despite its simplicity, this simple approach has shown to give very good performance in all of our experiments. Shown in Figure 8.1 is an example showing the difference of results with and without performing crosschecking. Please note that in this thesis, we always use the same coloring scheme as in the work of Baker et al. [163], which is also shown in Figure 8.1.



(a) Without cross-checking (b) With cross-checking

Figure 8.1: Cross-checking based inconsistency detection helps to correct errors due occlusions for the "Wooden" dataset, resulting in sharp motion discontinuities.



(a) Without continuous optimization (b) With continuous optimization

Figure 8.2: Initialized with optical flow result from discrete optimization, local continuous optimization at the pixel level can smooth out the "grains" due to the use of constant region motion model.

(b) Continuous Optimization based Smoothing

Our method can recover very smooth optical flow results by using single-pixel regions and performing multiple iterations at the finest segmentation level. For better quality, a final local continuous optimization similar to the fusion approach [158] is further performed. Since the results from discrete optimization are usually very close to the true displacements, such local continuous optimization converges very fast and mainly provides smoothing effects. Illustrated in Figure 8.2 is a comparison of results without and with continuous smoothing.

8.4 Algorithm Workflow

For better clarity, our proposed algorithm is summarized as follows.

Step 1: Build image pyramids and use downsized images to determine the initial displacement search ranges (Section 8.3.2)

Step 2: At each image pyramid level, over-segment image $\boldsymbol{I}_t$ using L different granularity constraints and build the corresponding L region-trees

Step 3: Loop from segmentation level $l = 0$ to L

Step 4: At each segmentation level:

(a) Set up the label-to-displacement LUT for each region r_l in the current region-tree $\boldsymbol{\mathcal{T}}_l$ (Section 8.3.1)

(b) Evaluate "*label space images*" for all the hypothesized labels $\ell \in \mathcal{L}$. GPU is used for better efficiency in fast image interpolation (Section 8.3.3)

(c) Run DP to optimize the corresponding energy function and induce the optical flow field $(\boldsymbol{U},\boldsymbol{V})_l$ from the resulting optimal labeling $\boldsymbol{L}(\mathcal{T}_l)$

Step 5: Go to Step 3 if $l < L$, otherwise perform K more iterations at the finest level to further refine the optical flow $(\boldsymbol{U},\boldsymbol{V})$ from $\boldsymbol{I}_t$ to $\boldsymbol{I}_{t+1}$ at the current pyramid level (Section 8.3.1)

Step 6: Recover the optical flow $(\boldsymbol{U}',\boldsymbol{V}')$ from $\boldsymbol{I}_{t+1}$ to $\boldsymbol{I}_t$ and perform cross-checking based correction (Section 8.3.4-a)

Step 7: Perform continuous optimization for smoothing (Section 8.3.4-b)

Step 8: Go to Step 2 if there is a finer scale image pyramid level

---

List 8.1: Workflow of our C2F region-tree based optical flow estimation algorithm

8.5 Experimental Results and Evaluation

8.5.1 Middlebury Optical Flow Benchmarking

We use the 2-frame color version of the Middlebury optical flow benchmarking datasets [163] to quantitatively evaluate the proposed method. In particular, 12 image sequences from hidden fluorescent texture, realistic synthetic, stereo and real video categories are tested. In all of the experiments, we used the same set of parameters, which were not explicitly optimized for performance tuning. We use

$L = 2$ level image over-segmentations as it is observed to be enough to provide good performance. The constant $\alpha = 0.04$ is used to initialize the global search range and $\Delta = 0.5$ is used for the local search range refinement. At the fine segmentation level, $K = 2$ refinement iterations are performed with each $\Delta = 0.25$ and $\Delta = 0.125$ are used.

In discrete optimization, 25×25 labels are used by default and the sampling interval δ is correspondingly determined based on the ranges being sampled. When the displacement range is too wide or too narrow, the number of labels used in the corresponding direction is adjusted adaptively for better sampling resolution or efficiency.

As for over-segmentation, the granularity in the coarse level is determined by limiting the region number to be around 3000, while for the fine level, the minimal region size is fixed at 5 pixels, instead of using single pixel regions as done in stereo matching. This is because we have found that although using single pixel regions in the finest segmentation level can improve the results, such improvement is usually minor due to the continuous optimization based smoothing, while being slower than using larger finest-level regions.

The threshold β is set as 1.0 and the ZNCC window size is 3×3 for the coarse segmentation level and 5×5 for the fine level. The smoothness weight scale ρ is set as 0.85 . All of these parameters are empirically determined and could be further optimized.

In Table 8.1, we show the average angular error (AAE) and the average end-point error (AEPE) of the top five algorithms at the time of writing this thesis. Our results on 8 datasets for quantitative evaluation are shown in Figures 8.1 to 8.5. It can be seen that the overall ranking of our method is pretty high (both 4th for AAE and AEPE). Furthermore, it is ranked first or second on 4 out of 16 available performance measures including angular SD, endpoint SD, R2.0 and A95 measures. In particular, the lowest AAE is obtained for the "Teddy" and "Shcefflera" datasets and the lowest AEPE is obtained for "Shcefflera" and the "Teddy" and "Grove" datasets. One possible reason for the relatively inferior AAE performance on these datasets may be due to our current method of

discretization. That is, uniformly sampling in the horizontal and vertical displacement search range results in non-uniformity in angular sampling.

Moreover, the average performance of our method on the Yosemite sequence also negatively impact the overall ranking. This is mainly due to its small image dimension which makes the finest granularity regions still not fine enough to capture subtle motion details. We have found that using different parameters specific to this dataset could improve its performance to some extent. On the other hand, our method obtained superior evaluations around motion discontinuities in most datasets, showing our region-based representation has the advantage in preserving motion boundaries. Furthermore, using continuous optimization gives slightly better statistics than the one without using it and boosts the overall ranking by approximately one position.

Average angle error	avg. rank	Army (Hidden texture) GT im0 im1			Mequon (Hidden texture) GT im0 im1			Schefflera (Hidden texture) GT im0 im1			Wooden (Hidden texture) GT im0 im1			Grove (Synthetic) GT im0 im1			Urban (Synthetic) GT im0 im1			Yosemite (Synthetic) GT im0 im1			Teddy (Stereo) GT im0 im1		
		all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext
Complementary OF [27]	4.9	4.44 10	11.2 7	4.04 11	2.51 2	9.77 3	1.74 1	3.93 4	10.6 4	2.04 1	3.87 8	18.8 5	2.19 6	3.17 1	4.00 1	2.92 4	4.64 7	13.8 3	3.64 6	2.17 5	3.36 3	2.51 13	3.08 2	7.04 2	3.65 9
Adaptive [26]	5.3	3.29 1	9.43 1	2.28 1	3.10 7	11.4 8	2.46 7	6.58 11	15.7 11	2.52 6	3.14 1	15.6 2	1.56 1	3.67 9	4.46 6	3.48 8	3.32 1	13.0 2	2.38 1	2.76 12	4.39 12	1.93 8	3.58 4	8.18 4	2.88 3
Aniso. Huber-L1 [28]	6.7	3.71 3	10.1 3	3.08 3	4.36 14	13.0 9	3.77 13	6.92 13	15.3 9	3.60 14	3.54 3	15.9 3	2.04 5	3.38 3	4.45 5	2.47 2	3.88 3	12.9 1	2.74 2	3.37 16	4.36 11	2.85 15	3.16 3	7.52 3	2.90 4
DPOF [21]	7.2	5.12 14	12.9 14	3.49 8	3.07 6	10.3 4	2.44 6	3.09 1	7.47 2	2.43 5	3.42 2	12.9 1	2.41 10	3.55 6	4.56 9	3.35 6	4.69 8	14.2 4	5.14 8	3.59 18	4.67 16	3.83 21	2.00 1	4.93 1	1.65 1
Spatially variant [22]	7.3	3.73 4	10.2 6	3.33 4	3.02 5	11.0 7	2.67 8	5.36 7	13.8 8	2.35 2	3.67 4	19.3 7	1.84 3	3.81 12	4.81 16	3.69 11	4.48 6	16.0 9	3.90 7	2.11 3	3.26 2	2.12 10	4.66 12	9.41 10	4.35 12
TV-L1-improved [20]	8.2	3.36 2	9.63 2	2.62 2	2.82 4	10.7 6	2.23 2	6.50 10	15.8 12	2.73 7	3.80 7	21.3 13	1.76 2	3.34 2	4.38 4	2.39 1	5.97 10	18.1 15	5.67 14	3.57 17	4.92 20	3.43 19	4.01 8	9.84 11	3.44 7
Occlusion bounds [29]	8.8	4.42 8	12.4 10	3.90 9	3.86 11	13.2 10	3.32 11	5.00 6	13.0 5	3.30 10	4.45 11	20.7 11	2.37 8	3.84 13	4.67 13	4.39 20	3.75 2	15.9 8	3.30 4	2.19 6	4.00 10	1.17 2	4.33 10	9.20 7	3.19 5
Brox et al. [8]	9.4	4.44 10	12.4 10	4.22 15	3.72 10	13.5 11	3.06 9	4.97 5	13.3 6	3.11 8	4.58 13	22.0 15	2.37 8	3.79 11	4.60 10	4.33 19	3.91 4	17.0 12	3.45 5	2.22 7	3.79 6	1.19 3	4.62 11	10.0 12	3.38 6
Multicue MRF [24]	9.5	4.50 12	10.1 3	4.18 14	2.52 3	7.07 1	2.36 5	3.09 1	7.41 1	2.36 3	4.46 12	20.8 12	2.73 13	3.51 5	4.11 2	4.06 15	6.08 12	15.6 7	5.40 12	5.25 26	5.36 22	9.02 25	3.63 5	8.39 5	4.15 11

Average end-point error	avg. rank	Army (Hidden texture) GT im0 im1			Mequon (Hidden texture) GT im0 im1			Schefflera (Hidden texture) GT im0 im1			Wooden (Hidden texture) GT im0 im1			Grove (Synthetic) GT im0 im1			Urban (Synthetic) GT im0 im1			Yosemite (Synthetic) GT im0 im1			Teddy (Stereo) GT im0 im1		
		all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext	all	disc	untext
Adaptive [26]	4.9	0.09 1	0.26 1	0.06 1	0.23 7	0.78 6	0.18 6	0.54 11	1.19 13	0.21 5	0.18 1	0.91 3	0.10 1	0.88 4	1.25 4	0.73 5	0.50 2	1.28 2	0.31 2	0.14 11	0.16 12	0.22 10	0.65 3	1.37 3	0.79 4
Complementary OF [27]	6.1	0.11 6	0.28 5	0.10 9	0.18 1	0.63 2	0.12 1	0.31 4	0.75 4	0.18 1	0.19 2	0.97 5	0.12 4	0.97 10	1.31 6	1.00 11	1.78 20	1.73 8	0.87 15	0.11 5	0.12 3	0.22 10	0.68 4	1.48 4	0.95 7
Aniso. Huber-L1 [28]	6.8	0.10 3	0.28 5	0.08 3	0.31 14	0.88 9	0.28 14	0.56 13	1.13 10	0.29 14	0.20 5	0.92 4	0.13 7	0.84 3	1.20 3	0.70 2	0.39 1	1.23 1	0.28 1	0.17 16	0.15 10	0.27 17	0.64 2	1.36 2	0.79 4
DPOF [21]	7.0	0.13 14	0.35 14	0.09 5	0.25 8	0.79 7	0.19 7	0.24 1	0.49 1	0.21 5	0.19 2	0.62 1	0.15 12	0.74 1	1.09 1	0.49 1	0.66 7	1.80 12	0.63 9	0.19 19	0.17 15	0.35 22	0.50 1	1.08 1	0.55 1
Spatially variant [22]	7.2	0.10 3	0.27 4	0.08 3	0.22 5	0.75 5	0.19 7	0.43 7	1.00 8	0.18 1	0.19 2	1.05 7	0.10 1	1.05 11	1.41 14	1.16 13	0.59 5	1.61 6	0.43 4	0.13 7	0.11 1	0.28 18	0.96 13	1.72 12	1.28 15
TV-L1-improved [20]	8.0	0.09 1	0.26 1	0.07 2	0.20 4	0.71 4	0.16 2	0.53 10	1.18 12	0.22 7	0.21 8	1.24 14	0.11 3	0.90 5	1.31 6	0.72 3	1.51 14	1.93 13	0.84 11	0.18 18	0.17 15	0.31 19	0.73 7	1.62 8	0.87 6
Multicue MRF [24]	8.5	0.11 6	0.26 1	0.11 13	0.19 2	0.53 1	0.17 5	0.24 1	0.49 1	0.19 3	0.24 11	1.13 10	0.15 12	0.79 2	1.10 2	0.72 3	1.47 13	1.60 5	0.85 12	0.28 26	0.19 23	0.71 26	0.78 9	1.53 6	1.09 11
Occlusion bounds [29]	9.0	0.11 6	0.32 9	0.10 9	0.29 11	0.94 13	0.24 12	0.39 5	0.93 5	0.26 11	0.22 10	1.10 9	0.12 4	1.10 14	1.37 10	1.50 20	0.93 9	1.77 9	0.57 8	0.10 2	0.14 8	0.11 1	0.87 11	1.71 11	1.06 8
Brox et al. [8]	9.5	0.11 6	0.32 9	0.11 13	0.27 10	0.93 11	0.22 10	0.39 5	0.94 6	0.24 9	0.24 11	1.25 15	0.13 7	1.10 14	1.39 12	1.43 18	0.89 8	1.77 9	0.55 7	0.10 2	0.13 5	0.11 1	0.91 12	1.83 14	1.13 13
Second-order prior [11]	9.8	0.11 6	0.31 8	0.09 5	0.26 9	0.93 11	0.20 9	0.57 15	1.25 17	0.26 11	0.20 5	1.04 6	0.12 4	0.94 7	1.34 8	0.83 9	0.61 6	1.93 13	0.47 6	0.20 20	0.16 12	0.34 21	0.77 8	1.64 9	1.07 10

Table 8.1: Middlebury benchmark ranking (average angle & endpoint errors). Red color highlights rows with results using our method.

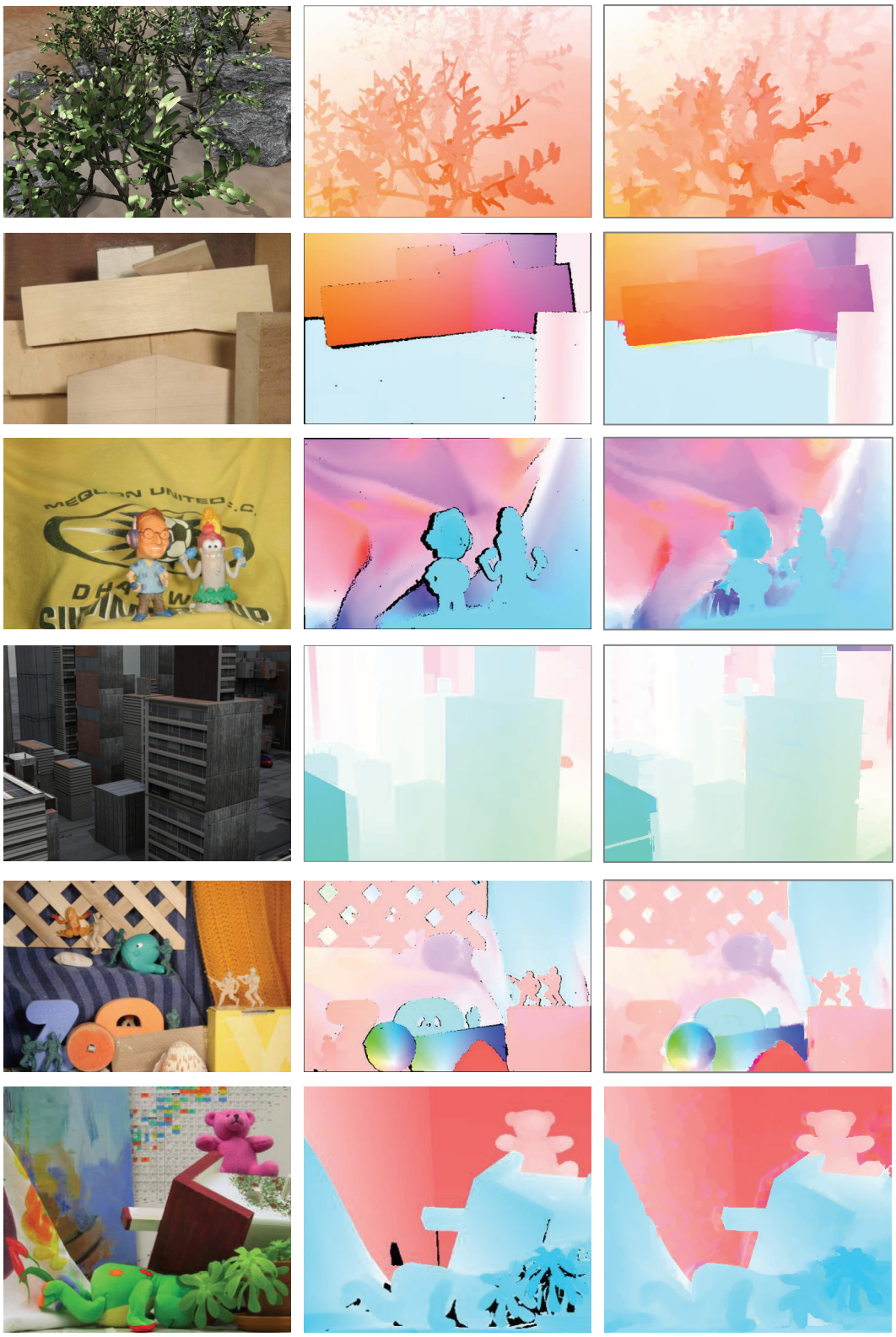


Figure 8.3: Results (right column) of the Middlebury Grove, Wooden, Mequon, Urban, Army and Teddy datasets along with the ground truths (middle column), in which black color means no ground truth is known.

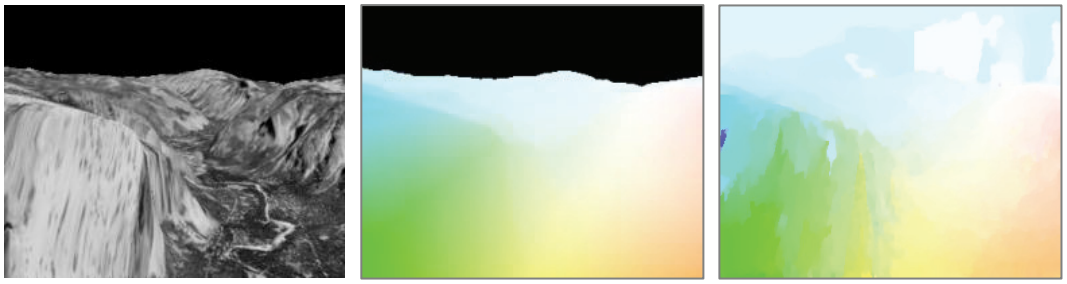


Figure 8.4: Result of the Middlebury "Yosemite" dataset.

As for the computation efficiency, take the Urban dataset [163] (image size 640x480) as an example, our un-optimized implementation takes a total running time of about 261 seconds on a PC with a dual-core AMD 2.2GHz CPU. The fairly high time cost is mainly due to its much larger displacement range (over 40 pixel in both the horizontal and vertical directions). For the other datasets with smaller displacement ranges, the time needed is much shorter.

8.5.2 Performance Comparison of using C2F or Single-level Region-tree

As shown in Figure 8.5, we compare the performance difference between using multiple level C2F region trees and the single-level one. Specifically, the optical flow result is obtained as shown in the second column by using only the finer level regions. However, C2F displacement refinement is still performed. As we can see, more errors are incurred along motion boundaries compared to the result using two-level region-trees as shown in the third column. Therefore, using C2F region tree does help to improve the accuracy. Furthermore, for reference, the result from the state-of-art continuous optimization based method [159] is also included. By comparison, we can see that using over-segmented regions instead of using pixels does have the unique advantage in handling sharp motion discontinuities.

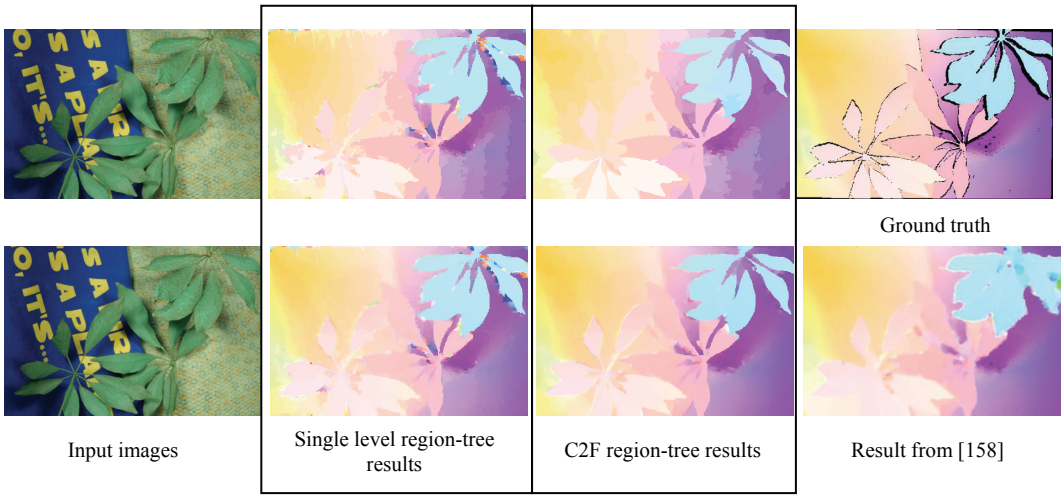


Figure 8.5: Comparison of using single-level region-tree and C2F region-tree for the "Schefflera" dataset.

8.5.3 Performance Profiling w.r.t Different Parameters

To quantitatively investigate the performance sensitivity of our algorithm to parameter settings, we further perform experiments on the other eight Middlebury datasets. Different from the benchmarking datasets shown previously, the flow field ground truth of these datasets are made public by Middlebury for algorithm designers to fine tune parameters. Some example results are shown in Figure 8.6.

Due to time and space limit, exhaustively profiling the performance of all the possible parameter combinations is beyond the scope of this thesis. Based on the experimental experience, here only three relatively more important parameters are investigated, i.e., the coarse level ZNCC window size, the coarse level and fine level segmentation granularities. For performance profiling, we vary the parameter of interest and keep the other ones fixed as reported in Section 8.5.1. In particular, odd number ZNCC window sizes from 3×3 to 11×11 are tested. For the coarse level segmentation granularity, the minimal region size of 30, 60, 90, 120, 150, 180 and 210 are used and the fine level segmentation granularity, the minimal region size of 1, 3, 5, 10, 15, 20 and 25 are profiled.

Shown in Figure 8.7 are the profiling curves of AAE and AEPE with respect to the corresponding parameters. From the curves, we can see that the general

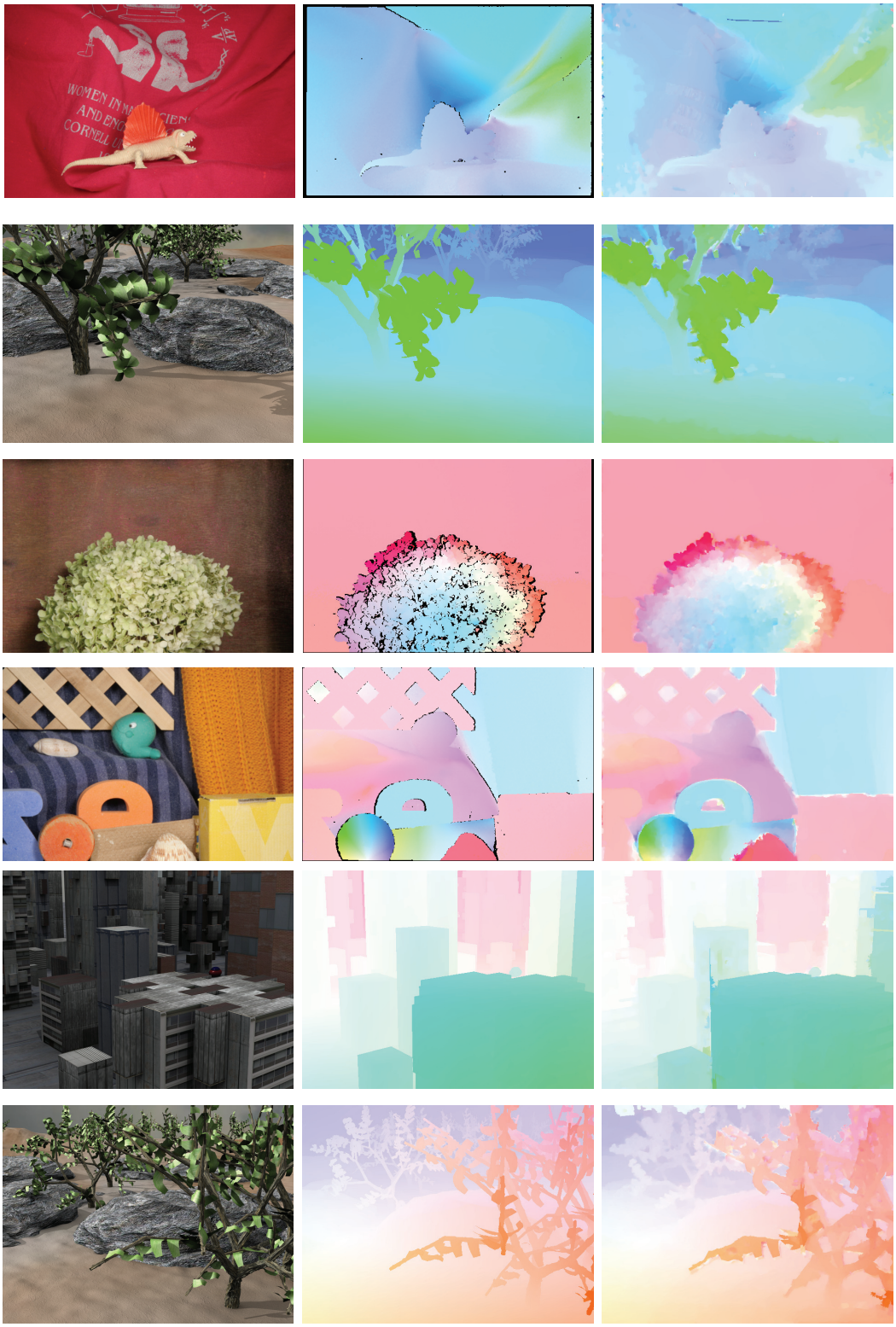


Figure 8.6: Example results (right column) of the Middlebury Dimetrodon, Grove2, Hydrangea, RubberWhale, Urban2 and Grove3 datasets along with the ground truths (middle column).

performance is pretty robust to the varied parameters for most datasets. However, some datasets do show higher sensitivity to specific parameters than others. For example, when the ZNCC window size is larger than 5, the datasets of Grove2 and Urban2 show larger AAE and AEPE. Furthermore, when the fine level minimal region size is equal to 20, the AEPE of Urban2 shows larger increases.

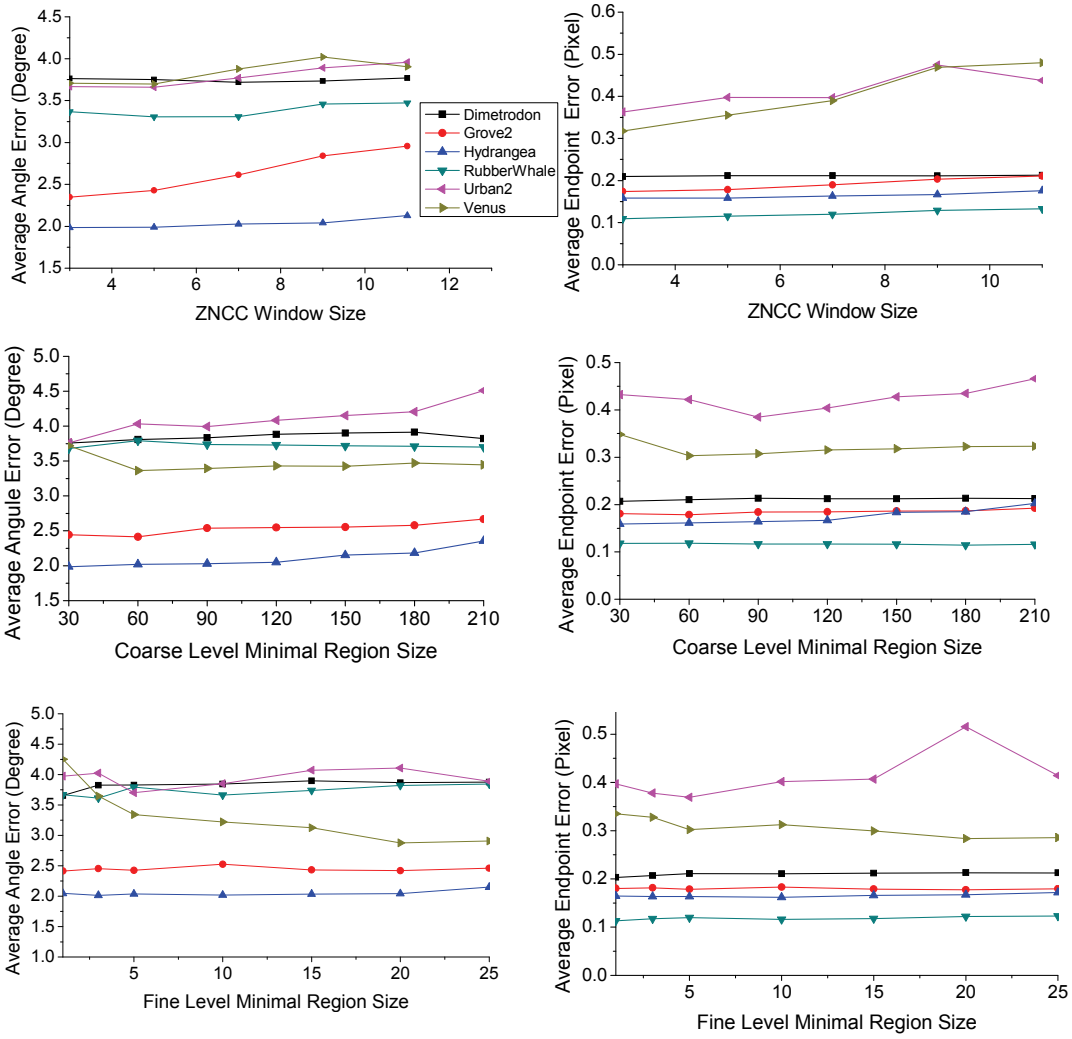


Figure 8.7: Profiling curves of performance to ZNCC window size, coarse and fine level segmentation granularities on 6 Middlebury datasets. For simplicity and clarity, the same legend is only added in the first graph.

Summary

In this chapter, we have presented another important application of our new region-tree based labeling framework to optical flow estimation. By using the C2F region-tree based image representation and incremental displacement search range refinement, good efficiency and performance are achieved. The presented method can produce sharp motion discontinuities through coarser segmentation while it is also capable of recovering subtle details through finer segmentation. As one of its many possible applications, in the next chapter, the new region-tree based optical flow estimation method is used in dense video depth recovery for FVV applications.

Chapter 9
Region-tree based Spatial-temporal Consistent Video Depth Recovery for FVV Rendering

In this chapter, the FVV scene reconstruction problem of recovering multiple view-dependent depth videos from the video streams captured from multiple viewpoints is addressed. This is commonly called the general position temporal stereo and its main challenge is to enforce spatial consistency across multiple synchronized views and temporal consistency across different time instants. That is, spatially, the global depth values assigned to pixels corresponding to the same 3D point in different synchronized views should be spatially close. And temporally, the depth values assigned to pixels corresponding to the same 3D point in successive frames of the same view should be consistent w.r.t. the continuity of its 3D motion. Such space-time consistency is especially important in FVV rendering, with which the rendering quality can be improved by suppressing the objectionable flickering artifacts during view transitioning or video playback.

We generalize the binocular stereo matching algorithm presented in Chapter 7 to general position multi-ocular stereo matching. Motion information obtained using the algorithm presented in Chapter 8 is also incorporated using a new temporal variant of region-tree representation for enforcing temporal consistency. To enforce spatial consistency, a novel inconsistency map based progressive optimization approach is used.

9.1 Prior Work

To enforce temporal consistency in video depth recovery, most related work relies on incorporating motion information. For example, Gong [168] proposes an integrated representation called disparity flow to encode the temporal mapping

between pixels and disparities and their corresponding pixels and disparities in two consecutive frames. The temporal consistency is enforced by using the globally optimized disparity flow field to predict disparity map of the next frame and by biasing the stereo matching cost volume in favor of the predicted disparity values. Zhang and Kambhamettu [169] compute the structure and motion simultaneously as an energy minimization problem by iteratively fitting an affine motion model to the partitioned image patches with global spatial smoothness regularization. Discontinuities are further preserved using color image segmentation information. Temporal consistency between depth maps of successive frames is enforced whenever possible by enhancing the traditional belief propagation (BP) algorithm to operate on a 3D graph that includes both spatial and temporal neighbors induced from the optical flow information [170]. By enabling BP to automatically disconnect neighbors whose beliefs are incompatible, it can avoid performance degradation due to errors in the optical flow. Similar region-based idea is also used in Tao *et al.*'s method [171], in which the color segmentations of input video frames are used to induce planar surface patch based 3D scene representation. By optimizing a cost function incorporating the spatial color consistency constraint and a smooth scene motion model, temporal consistent depth maps can be recovered.

Our method incorporates motion information in depth recovery by using a new temporal region-tree based framework with which the consistency of depth estimation between the current and the last frame can be enforced. Additionally, our method further enforces spatial consistency through an inconsistency map based progressive optimization. In particular, using multi-view geometric constraints, inconsistencies between depth maps for each view are obtained in each iteration of optimizations across different views and are captured in an inconsistency map, which is then used in the next iteration to progressively propagate consistency information and to correct inconsistencies through visibility reasoning and depth hypotheses pruning. Ours is different from the outlier confidence map concept proposed in [172], which dynamically measures how likely pixels are occluded and is mainly used as a soft constraint to regularize

matching cost evaluation. Instead, the role of our iteratively updated inconsistency map is to capture all spatial inconsistencies between depth maps, which are used to help in region-wise visibility reasoning and depth hypotheses pruning for improving spatial consistency in subsequent optimization iterations. Our work is most related to the work of Zhang et al. [173], in which a bundle adjustment optimization model is used to enforce temporal consistency (or equivalently spatial consistency for a static scene) by explicitly incorporating geometric consistency constraint across multiple frames of a video sequence along with the corresponding depth map video. Their method appears to produce impressive results. However, its reliance on redundancy information of a static scene video and a relatively small baseline may degrade its performance in our application with sparsely distributed multi-view images.

Background information has also been used to enforce space-time consistency before. For example, Goldlucke and Magnor [174] perform stereo matching and background separation simultaneously using graph-cut optimization under the assumption that clean background color and depth images are known. As a result, each pixel is assigned with a binary background or foreground label in addition to a depth label. Lee and Ho [175] enforce temporal consistency for each frame using temporal filtering using five consecutive frames and a background mask generated by thresholding. All five background masks are merged to form the temporal median filter mask. Larson et al. [170] apply clustering to the depth estimates together with the corresponding colors for all the pixels in all the frames of a sequence in a 4D HSV plus depth space. The median color and depth of the furthest significant cluster are selected as the background color and depth of each pixel. The background models are then used in belief propagation to define a pixel-wise background-depth bias based on color similarity and depth likelihood distribution. Our method also takes advantage of the background model. Its difference from [170] or [175] is in its new way for background estimation and its use in a new background-biased optimization.

9.2 Problem Formulation

Given a set of M video sequences $\widehat{\boldsymbol{S}} = \{\boldsymbol{S}_m | m = 0, \dots, M-1\}$, in which each video sequence $\boldsymbol{S}_m$ is captured by a stationary camera C_m for view V_m and has T video frames $\widehat{\boldsymbol{I}}_m = \{\boldsymbol{I}_m^t | t = 0, \dots, T-1\}$. At each time instant t, the synchronized frame set $\widehat{\boldsymbol{I}}^t = \{\boldsymbol{I}_m^t | m = 0, \dots, M-1\}$ contains snapshots of a dynamic scene from different viewpoints. Then the objective of our space-time consistent dynamic depth recovery problem is to estimate M depth map sequences $\widehat{\boldsymbol{D}}_m = \{\boldsymbol{D}_m | m = 0, \dots, M-1\}$ by maximizing the temporal consistency between any two consecutive depth maps D_m^t and D_m^{t+1} for view V_m, and the spatial consistency between the synchronized depth maps $\widehat{\boldsymbol{D}}^t = \{\boldsymbol{D}_m^t | m = 0, \dots, M-1\}$ for time instant t.

To use our region-tree based framework for matching a synchronized frame set $\widehat{\boldsymbol{I}}^t$, the general position multi-ocular stereo matching has to be formulated as a labeling problem.

We follow the well-known plane-sweeping scheme, by which the depth search space is globally discretized into a set of depth planes $\boldsymbol{\pi}_z = (0,0,d_z)^T$ $(z = 0, \dots, Z-1)$ each of which is perpendicular to the optical axis of camera C_0, with corresponding depths d_z $(z = 0, \dots, Z-1)$ tessellating a known depth range $[d_{near}, d_{far}]$ linearly or non-linearly. Each depth plane $\boldsymbol{\pi}_z$ is tagged with a label $\ell = z$, which together form the label set $\boldsymbol{\mathcal{L}}$. In this chapter, we use the same non-linear depth tessellation scheme as the one used in [15], that is,

$$d_z = \frac{1}{z/(Z-1)\left(\frac{1}{d_{near}} - \frac{1}{d_{far}}\right) + \frac{1}{d_{far}}} \quad with\ (z = 0, \dots, Z-1) \tag{9.1}$$

In this way, recovering the depth map D_m^t is mapped to a region-tree labeling problem.

Since there are $M - 1 > 1$ images to be matched, a generalized label space image function $c(\boldsymbol{I}_m^t, \boldsymbol{I}_k^t, x, y, d_\ell)$ is used to evaluate the matching cost of pixel (x, y) on $\boldsymbol{I}_m^t$ to its corresponding pixel in another image $\boldsymbol{I}_k^t$ according to depth d_ℓ and $M - 1$ label space images will be generated. As explained in Chapter 2, such pixel correspondence between views is found through depth based pixel forward

warping from view $\boldsymbol{I}_m^t$ to $\boldsymbol{I}_k^t$

$$\boldsymbol{\mathcal{F}}_{m\to k}(x,y,d) \sim \boldsymbol{K}_k \boldsymbol{R}_k^T \boldsymbol{R}_m \boldsymbol{K}_m^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} + d\boldsymbol{K}_k \boldsymbol{R}_k^T (\boldsymbol{T}_m - \boldsymbol{T}_k) \quad (9.2)$$

Similar to binocular stereo, different photo consistency measure such as SD, AD or ZNCC can be used to compute $c(\boldsymbol{I}_m^t, \boldsymbol{I}_k^t, x, y, d_\ell)$.

Then in the cost aggregation step, the data cost for a region r_i is calculated as

$$\mathcal{D}(r_i, \ell_{r_i}) = \frac{\sum_{k\in V_{r_i}} \sum_{(x,y)\in r_i} c\left(\boldsymbol{I}_m^t, \boldsymbol{I}_k^t, x, y, d_{\ell_{r_i}}\right)}{M_{r_i} \cdot N_{r_i}} \quad (9.3)$$

with V_{r_i} being the set of M_{r_i} views selected for evaluating the matching cost of region r_i . That is, for a specific region, the corresponding evaluations in the $M-1$ available label space images may have errors due to occlusions. Then some visibility reasoning based view selection mechanism is used to select only the views without occlusion errors for cost aggregation. Please note that such view selection is region dependent.

The smoothness cost function is defined as

$$\mathcal{S}\left(r_i, \ell_{r_i}, r_j, \ell_{r_j}\right) = \frac{min\left(max\left(\left|d_{\ell_{r_i}} - d_{\ell_{r_j}}\right| - \delta, 0.0\right), 15\delta\right)}{\left(w_{e_{(i,j)}} + \epsilon\right)} \quad (9.4)$$

where ϵ is a small constant value for avoiding divided-by-zero error and $\delta = \mu\left|d_{far} - d_{near}\right|$, the "penalty exempt" threshold is to allow for small gradual depth changes between regions. As detailed later, during iterative optimizations, functions $\mathcal{D}(r_i, \ell_{r_i})$ and $\mathcal{S}\left(r_i, \ell_{r_i}, r_j, \ell_{r_j}\right)$ will be changed to incorporate the last iteration results and to allow for larger depth deviations for error correction.

9.3 Algorithm Overview

Our proposed method has two passes. In each pass, for each input video frame $\boldsymbol{I}_m^t$ $(t > 0)$, a region tree $\boldsymbol{\mathcal{T}}(\boldsymbol{I}_m^t)$ is built from a set of over-segmented regions $\boldsymbol{\mathcal{R}}_m^t$ as described in Chapter 7. Then the region-tree based optical flow estimation is done as described in Chapter 8 to estimate an optical flow field $\boldsymbol{\mathcal{O}}_{t\to t-1}$ from $\boldsymbol{I}_m^t$

to $\boldsymbol{I}_m^{t-1}$. With $\boldsymbol{\mathcal{O}}_{t\to t-1}$ known, for each region $r_i \in \boldsymbol{\mathcal{R}}_m^t$, its "temporally adjacent" region $\mathcal{H}_{r_i}$ is found in $\boldsymbol{I}_m^{t-1}$. Then new inter-frame edges connecting $r_i \in \boldsymbol{\mathcal{R}}_m^t$ and its corresponding $\mathcal{H}_{r_i}$ are augmented to $\boldsymbol{\mathcal{T}}(I_m^t)$, resulting in a temporal region tree $\overline{\boldsymbol{\mathcal{T}}}(I_m^t)$. For the first frame $\boldsymbol{I}_m^t$ $(t = 0)$, $\overline{\boldsymbol{\mathcal{T}}}(\boldsymbol{I}_m^t)$ is just $\boldsymbol{\mathcal{T}}(\boldsymbol{I}_m^t)$.

After $\overline{\boldsymbol{\mathcal{T}}}(\boldsymbol{I}_m^t)$ is constructed, a progressive optimization scheme is used to iteratively enforce depth spatial consistency among different views by correcting inconsistencies captured in the corresponding inconsistency map $\boldsymbol{Q}_m^t$.

Specifically, K iterations of optimizations are performed with the first iteration initializing depth map $\boldsymbol{D}_m^t$ using the normal DP on region-tree $\overline{\boldsymbol{\mathcal{T}}}(\boldsymbol{I}_m^t)$ optimization approach. Then in each of the subsequent iterations, the last iteration depth maps $\widehat{\boldsymbol{D}}^t$ for all views are used to generate an inconsistency map $\boldsymbol{Q}_m^t$ for each view by identifying inconsistent regions whose depth values violate the multi-view geometric constraint.

The depth values of consistent regions will be "*frozen*" and will not change in subsequent iterations. The inconsistency maps together with the last iteration depth maps are used to prune incorrect depth hypotheses through a Z-buffer like mechanism and to select appropriate view based on visibility reasoning. In this way, inconsistencies can be corrected progressively and the final depth map $\boldsymbol{D}_m^t$ is then used for recovering $\boldsymbol{D}_m^{t+1}$ if $t < T - 1$.

After all T frames of $\widehat{\boldsymbol{I}}_m$ have been processed in the first pass, the background information is estimated for each view V_m and used in the second pass for background biased optimization to further improve the space-time consistency between depth maps.

Shown in Figure 9.1 is an illustration of the typical steps in our proposed algorithm. The second row shows the inconsistency map based iterative optimizations of the depth map in the first pass. After the first pass, the background information is estimated. Then with the background information available, which is shown in the third row, in the second pass, the depth map is iteratively refined. As we can see, in the first iteration of pass one, the depth map exhibits many errors. By iteratively enforcing spatial consistency, many of these errors are corrected and accordingly the number of inconsistent pixels (white

pixels in the inconsistency map) decreases after each iteration. Furthermore, the use of background information in the second pass further improves the depth accuracy.

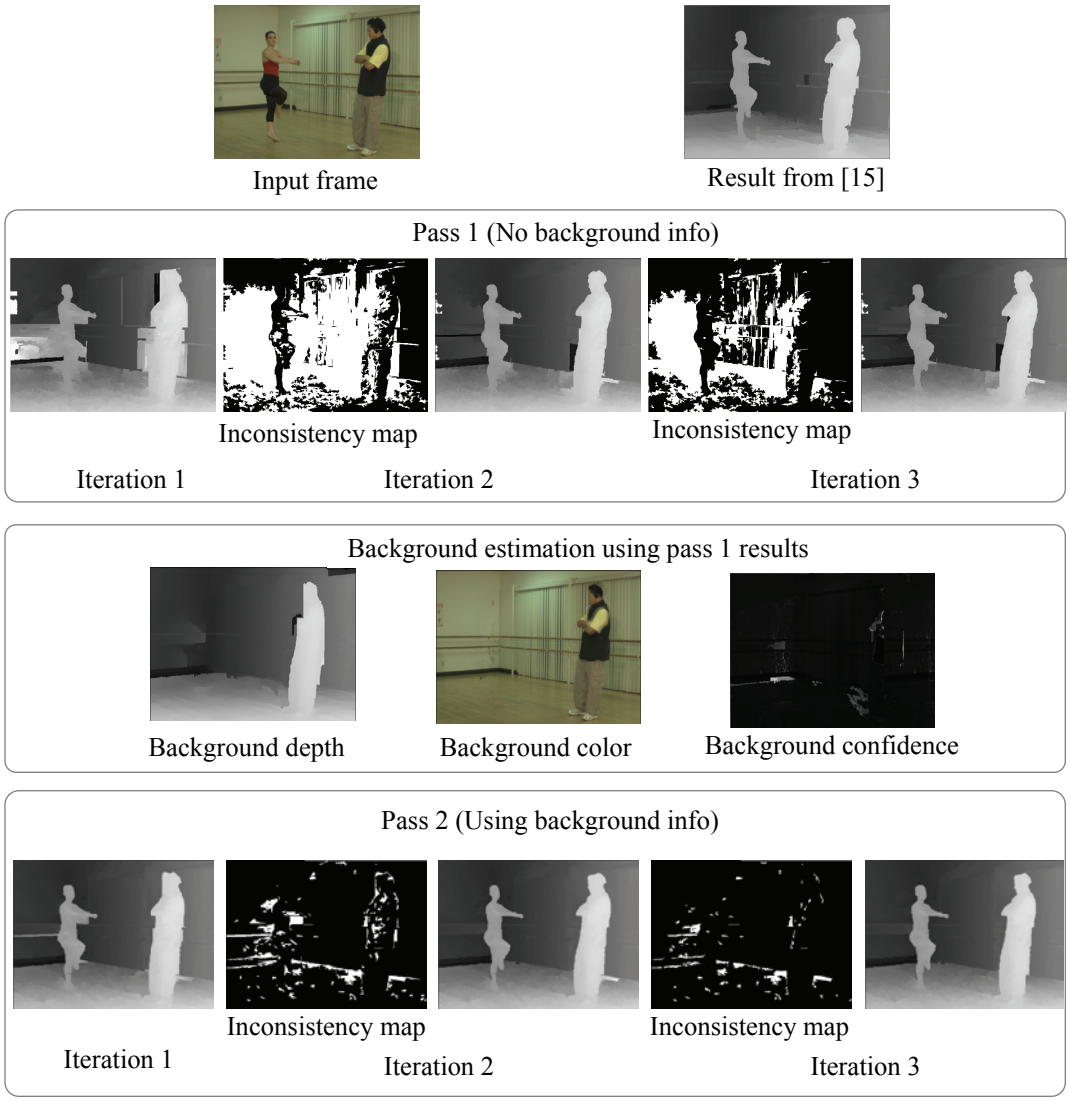


Figure 9.1: Illustration of the typical steps in our proposed algorithm with frame 57 view 7 of the Ballet dataset.

For better clarity, the workflow of our proposed algorithm is summarized in List 9.1.

Starting with pass $p = 0$ and time $t = 0$

Step 1: If $p > 0$, estimate background information for each view using results from pass $p - 1$

Step 2: Over-segment frame $\boldsymbol{I}_m^t$

Step 3: If $t > 0$, estimate the optical flow field $\boldsymbol{\mathcal{O}}_{t \to t-1}$ from $\boldsymbol{I}_m^t$ to $\boldsymbol{I}_m^{t-1}$ and build the temporal region-tree $\overline{\boldsymbol{\mathcal{T}}}(\boldsymbol{I}_m^t)$, otherwise build $\overline{\boldsymbol{\mathcal{T}}}(\boldsymbol{I}_m^t)$ as the normal one $\boldsymbol{\mathcal{T}}(\boldsymbol{I}_m^t)$

Step 4: Iterate K times $k = 0, \ldots, K - 1$

Step 5: In each iteration k

(a) If $k = 0$, set all regions $\in \overline{\boldsymbol{\mathcal{T}}}(\boldsymbol{I}_m^t)$ as inconsistent;
else use the depth map results $\widehat{\boldsymbol{D}}^t$ from iteration $k - 1$ to generate/update inconsistency map $\boldsymbol{Q}_m^t$ and to identify consistent and inconsistent regions

(b) Run DP on $\overline{\boldsymbol{\mathcal{T}}}(\boldsymbol{I}_m^t)$ minimizing energy function (6.1) without changing the depth values of consistent regions and obtain new depth maps $\boldsymbol{D}_m^t$. Background information is used if $p > 0$

(c) $k = k + 1$, If $k < K - 1$, goto Step 4, otherwise obtain the final $\boldsymbol{D}_m^t$

Step 6: $t = t + 1$, if $t < T - 1$ goto Step 2

Step 7: $p = p + 1$, if p < 2, $t = 0$ and goto Step 1

List 9.1

9.4 Implementation Details

9.4.1 Temporal Region-tree based Temporal Consistency Enforcement

We extend the traditional 2D region-tree $\boldsymbol{\mathcal{T}}(\boldsymbol{I}_m^t)$ to a temporal one $\overline{\boldsymbol{\mathcal{T}}}(\boldsymbol{I}_m^t) = \overline{\boldsymbol{\mathcal{T}}}(\overline{\boldsymbol{\mathcal{R}}_m^t}, \bar{\mathcal{E}})$ for enforcing temporal consistency between depth maps D_m^t and D_m^{t-1} when $t > 0$. For this, the optical flow estimation from $\boldsymbol{I}_m^t$ and $\boldsymbol{I}_m^{t-1}$ is performed using our region-tree based method elaborated in the last chapter. However, we only use single-level region-tree instead of a C2F one here. The main reason is that most of our videos are of low frame-rate which makes accurate optical flow estimation too challenging for scenes with fast movements. Therefore, as to be explained below, in the current implementation, we only trust optical flow

information of static background regions, for which using single-level region-tree is sufficient and efficient.

With the optical flow field $\boldsymbol{\mathcal{O}}_{t\to t-1}$ from frame $\boldsymbol{I}_m^t$ to $\boldsymbol{I}_m^{t-1}$ obtained, each region $r_i \in \boldsymbol{\mathcal{T}}(I_m^t)$ is assigned with a 2D displacement vector $\boldsymbol{\mathcal{O}}(r_i)$ by which "temporal neighbor" $\mathcal{H}_{r_i}$ of region r_i in $\boldsymbol{I}_m^{t-1}$ can be found.

At this stage the depth map $\boldsymbol{D}_m^{t-1}$ of the previous frame $\boldsymbol{I}_m^{t-1}$ has been estimated with spatial and/or temporal consistency enforced. Since the result is pretty accurate, it is reasonable to further enforce the depth smoothness between each region r_i and its temporal neighbor $\mathcal{H}_{r_i}$, in addition to the smoothness between r_i and its spatial neighbors. To this end, for each region r_i, we add a new node to $\boldsymbol{\mathcal{R}}_m^t$for $\mathcal{H}_{r_i}$ and a "temporal edge" $e_{r_i\to\mathcal{H}_{r_i}}$connecting r_i and $\mathcal{H}_{r_i}$, extending $\boldsymbol{\mathcal{T}}(\boldsymbol{I}_m^t)$ to a temporal region-tree $\bar{\boldsymbol{\mathcal{T}}}(\boldsymbol{I}_m^t)$. Then performing energy minimization on $\bar{\boldsymbol{\mathcal{T}}}(\boldsymbol{I}_m^t)$ to estimate $\boldsymbol{D}_m^t$ implicitly enforces its consistency to D_m^{t-1}.

However, to limit propagation of possible errors in $\boldsymbol{D}_m^{t-1}$ and over smoothness enforcement due to errors in the optical flow field $\boldsymbol{\mathcal{O}}_{t\to t-1}$, we trust the optical flow estimation accuracy of static background regions more than dynamic ones and only add temporal edges to regions whose $|\mathcal{O}(\mathrm{r_i})| = 0$. Furthermore, we perform consistency checking on the optical flow fields $\boldsymbol{\mathcal{O}}_{t\to t-1}$ for all views based on depth maps $\boldsymbol{D}_m^{t-1}$ to filter out erroneously identified static background regions. The weight of temporal edge $e_{r_i\to\mathcal{H}_{r_i}}$is defined as

$$w_{e_{r_i\to\mathcal{H}_{r_i}}} = \frac{\sum_{(x,y)\in r_i}\left(2.0-\eta\left(I_m^t(x,y),I_m^{t-1}\left(x+u_{\mathbb{L}(r_i)},y+v_{\mathbb{L}(r_i)}\right)\right)\right)}{\boldsymbol{N}_{r_i}} \tag{9.5}$$

That is, the similarity between two regions r_i and $\mathcal{H}_{r_i}$is evaluated based on the pixel ZNCC measure η and is used to define the weight. The more similar they are, the smaller the weight, the more influence on r_i from $\mathcal{H}_{r_i}$.

Shown in the first row of Figure 9.2 is an example of the optical flow estimation result for two consecutive frames of a lab scene video, which is color

coded using the same scheme as that specified in the Middlebury Stereo Vision site [162]. As we can see, the optical flow result is quite accurate. For low FPS sequences with fast movements, even though more errors could be incurred, most static regions (colored as white) are still accurate. Conservatively using only static regions to build temporal region tree can lower the impact of inaccuracies in the optical flow estimation while still can improve in enforcing temporal consistency.

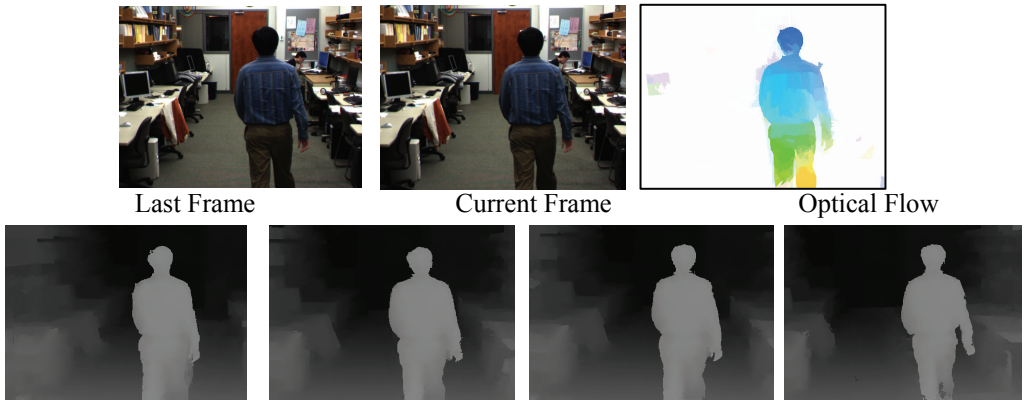


Figure 9.2: Results of an example lab test dataset.

9.4.2 Progressive Spatial-consistency Enforcement

The depth spatial consistency across different views is progressively enforced by using inconsistency maps.

(1) Inconsistency map generation

To generate inconsistency map $\boldsymbol{Q}_m^t$, we use the geometric consistency constraint proposed by Zhang et al. [173] as the measure for identifying inconsistent pixels as follows. For a pixel $\boldsymbol{x} = (x, y)$ with depth $\boldsymbol{D}_m^t(x, y)$ in depth map of view V_m, its corresponding pixel $\boldsymbol{x}' = (x', y')$ in another view $V_i (i \neq m)$ is found using Equation (9.4) as

$$(x', y') = \boldsymbol{\mathcal{F}}_{m \to i}\left(x, y, D_m^t(x, y)\right)$$

Then the conjugate pixel (x', y') is warped back to pixel $\boldsymbol{x}" = (x", y")$ in view V_m similarly. Ideally, pixels $\boldsymbol{x}$ and $\boldsymbol{x}"$ should coincide with each other. So the distance $\|\boldsymbol{x} - \boldsymbol{x}"\|$ between pixels $\boldsymbol{x}$ and $\boldsymbol{x}"$ provides a reasonable measure of the consistency between two depth values $\boldsymbol{D}_m^t(x, y)$ and $\boldsymbol{D}_i^t(x', y')$. Therefore after

checking all of the view pairs $(V_m, V_i)(i = 0, \ldots, M-1, i \neq m)$, if the total number of views for which $\|\boldsymbol{x} - \boldsymbol{x}"\| > 2$ is greater than $M/2$, pixel $\boldsymbol{x}$ is marked as inconsistent by setting $\boldsymbol{Q}_m^t(\boldsymbol{x}) = 1$ to indicate that its depth value $\boldsymbol{D}_m^t(x, y)$ needs to be re-estimated, otherwise $\boldsymbol{x}$ is marked as consistent by setting $\boldsymbol{Q}_m^t(\boldsymbol{x}) = 0$. For reference, example inconsistency maps are shown in Figure 9.1.

After inconsistency map $\boldsymbol{Q}_m^t$ is generated, all of the corresponding regions in $\overline{\boldsymbol{T}}(\boldsymbol{I}_m^t)$ which have over one quarter of their pixels marked as inconsistent will be flagged as inconsistent otherwise consistent. The depth value of a consistent region is regarded as determined and will not be changed, while the depth value of an inconsistent region is regarded as undetermined and can be changed in subsequent iterations.

(2) Progressive spatial consistency enforcement

Our iterative optimization based spatial consistency enforcement scheme is a progressive one similar to that proposed by Wei and Quan [176]. In any non-first iteration, previously identified consistent regions provide useful information for constraining current matching optimization and resolving matching ambiguities. In particular, inconsistency map $\boldsymbol{Q}_m^t$ together with the last iteration depth map $\boldsymbol{D}_m^t$ are used in the matching cost evaluation step to prune wrong depth hypotheses of regions and to use visibility reasoning to select views to better handle occlusions.

(a) Depth hypotheses pruning

In any iteration after the first one, the label space image function used is a variant of the original $c(\boldsymbol{I}_m^t, \boldsymbol{I}_k^t, x, y, d)$ for incorporating the inconsistency and depth information of the last iteration. Specifically, in iteration>1, when evaluating the matching cost of pixel $\boldsymbol{p} \in I_m^t$ for a candidate depth d, there are three possible cases for its corresponding pixel $\boldsymbol{p}'$ in another view V_i. That is, $\boldsymbol{p}'$ has been flagged as inconsistent since $\boldsymbol{Q}_i^t(\boldsymbol{p}') = 1$, or $\boldsymbol{p}'$ is consistent with $\boldsymbol{Q}_i^t(\boldsymbol{p}') = 0$ and d is larger or smaller than its determined depth $D_i^t(\boldsymbol{p}')$. Accordingly, we use a new pixel-based photo consistency function as $\tilde{c}(\boldsymbol{I}_m^t, \boldsymbol{I}_k^t, x, y, d)$

$$= \begin{cases} c(I_m^t, I_k^t, x, y, d) & Q_i^t(\boldsymbol{p}') = 1 \\ \vartheta & Q_i^t(\boldsymbol{p}') = 0 \;\&\; d > D_i^t(\boldsymbol{p}') \\ \infty & Q_i^t(\boldsymbol{p}') = 0 \;\&\; d \leq D_i^t(\boldsymbol{p}') \end{cases} \tag{9.6}$$

where ϑ is the occlusion cost. In particular, for the case when the depth of $\boldsymbol{p}'$ is inconsistent, i.e. undetermined, the original cost evaluation is used. For the case when $\boldsymbol{p}'$ has been determined before, if $d > \boldsymbol{D}_i^t(\boldsymbol{p}')$, i.e. the 3D point corresponding to $(\boldsymbol{p}, d)$ is occluded by other known closer 3D point $(\boldsymbol{p}', \boldsymbol{D}_i^t(\boldsymbol{p}'))$, an occlusion penalty is incurred. Conversely, if $d \leq \boldsymbol{D}_i^t(\boldsymbol{p}')$, which means that 3D point $(\boldsymbol{p}, d)$ occludes 3D point $(\boldsymbol{p}', \boldsymbol{D}_i^t(\boldsymbol{p}'))$, which is not possible because otherwise depth $\boldsymbol{D}_i^t(\boldsymbol{p}')$ cannot be flagged as consistent with $\boldsymbol{Q}_i^t(\boldsymbol{p}') = 0$, a larger penalty is given to candidate depth d so that it will less likely be chosen in the optimization result. Through this Z-buffer like mechanism [176], invalidate depth hypotheses can be implicitly pruned.

Furthermore, to correct errors caused by possible over-smoothness in the first iteration, a new "penalty exempt" depth deviation threshold $\delta' = 5\delta$ is used in function $\mathcal{S}$ for any edge linking a consistent region to an inconsistent region to allow for a larger depth discontinuity.

(b) Visibility reasoning and view selection

As observed in various experiments, appropriate region-wise view selection in the stereo matching cost evaluation step is very important for better result. In particular, the "best half of candidate views" selection strategy proposed in [177] is used for region-wise view selection. That is, views V_{r_i} used in Equation (9.3) for region r_i will correspond to half of the candidate views that give lower matching costs for depth hypothesis d. In the first iteration, all available M views are used as candidate views. Then during subsequent iterative optimizations, we further take advantage of the last iteration results from other views to improve view selection. Specifically, we first warp all the other views' depth maps $\boldsymbol{D}_i^t\ (i \neq m)$ to view V_m. For each inconsistent region in view V_m, all the other views witnessing larger depth values at the same location will be collected as candidate views. If the number of collected candidate views is less than 2, then as

above all M views are used for that region. This simple visibility reasoning is based on the observation that an occluded region is more likely to be assigned with the depth value of a foreground object.

9.4.3 Background Biased Optimization

(1) Background estimation

After the first pass is done, we apply a novel segmentation-based background estimation algorithm to recover background color $\boldsymbol{B}_m^{color}$, depth $\boldsymbol{B}_m^{depth}$ and confidence $\boldsymbol{B}_m^{conf}$ for each view V_m.

Specifically, given the depth map results $\widehat{\boldsymbol{D}}_m$ from the first pass, B_m^{depth} is estimated as

$$\boldsymbol{B}_m^{depth}(x,y) = \boldsymbol{D}_m^{f_m(x,y)}(x,y)$$

where $f_m(x,y)$ corresponds to an index $t \in [0, T-1]$ that selects a pixel (x,y) in an image in $\widehat{\boldsymbol{D}}_m$ out of T candidates as the background depth at (x,y). $f_m(x,y)$ is solved scanline by scanline as a labeling problem by minimizing a cost function using DP [178].

To further improve smoothness, after $f_m(x,y)$ is determined, all the pixels at (x,y) in the entire depth map sequence $\widehat{\boldsymbol{D}}_m$ with their absolute intensity difference to the candidate $\boldsymbol{D}_m^{f_m(x,y)}(x,y)$ is less than 5 are collected for linear blending. By applying the same selection results and blending to the corresponding color image $\widehat{\boldsymbol{I}}_m$, the background color image $\boldsymbol{B}_m^{color}$ is obtained simultaneously.

In addition, the variances of each pixel in $\boldsymbol{B}_m^{depth}$ and $\boldsymbol{B}_m^{color}$ are computed and used to define the background confidence map $\boldsymbol{B}_m^{conf}$. If the variance is high, then the confidence that the pixel is a static background pixel is low. We further apply inconsistency checking as done in Section 9.4.2 to all background depth maps $\boldsymbol{B}_m^{depth}(m = 0, \dots, M-1)$. All the identified inconsistent pixels will be given a zero confidence to avoid error propagation to subsequent background biased optimization. As can be seen in Figure 9.3, by doing estimation on depth maps instead of color images directly, much better background color and depth

quality can be obtained compared to traditional median filtering or clustering based approaches [170]. This is because depth maps are invariant to illumination changes as well as to ambiguities in the foreground and background colors.

Background depth | Background color | Background confidence

Figure 9.3: An example of background estimation for view 5 of the "Breakdancing" dataset from [15]. (Brighter color in background confidence map means lower confidence.)

(2) Background biased optimization

Then the background information obtained above is used as a high-level prior by biasing the depth hypothesis cost evaluations of the background-like pixels. In particular, for each pixel (x, y) in $\boldsymbol{I}_m^t$, a background bias weight is defined based on its likelihood to be a static background pixel for depth hypothesis d as

$$w_{bias}(x, y, d) = \beta_1 / \eta\left(\boldsymbol{I}_m^t(x, y), \boldsymbol{B}_m^{color}(x, y)\right) \tag{9.7}$$

if $\boldsymbol{B}_m^{conf}(x, y) > \beta_2$, $\eta\left(I_m^t(x, y), \boldsymbol{B}_m^{color}(x, y)\right) > \beta_3$ and $d = \boldsymbol{B}_m^{depth}(x, y)$, otherwise $w_{bias}(x, y, d) = 1.0$, where η is the ZNCC measure. Then the label space image function becomes a biased version as

$$\tilde{c}(I_m^t, I_k^t, x, y, d) = w_{bias} \cdot c(I_m^t, I_k^t, x, y, d) \tag{9.8}$$

In this way, the optimization and inconsistency based error correction has a higher probability to assign background pixels with the correct background depths, resulting in better sequence wise temporal consistency.

Background information is also used in inconsistency checking for identifying inconsistencies due to occluded background pixels. Specifically, for a pixel $\boldsymbol{I}_m^t(x, y)$, if $\boldsymbol{B}_m^{conf}(x, y) > \beta_2$, $\eta\left(\boldsymbol{I}_m^t(x, y), \boldsymbol{B}_m^{color}(x, y)\right) > \beta_4$ and $\boldsymbol{D}_m^t(x, y) = d = \boldsymbol{B}_m^{depth}(x, y)$, then it will not flag as an inconsistent pixel even if it fails the consistency checking. We use a conservatively higher β_4 value to avoid pre-mature errors.

9.5 Experimental Results and Evaluations

The algorithm proposed in this paper is extensively tested using a variety of multi-view stereo video sequences, including the Microsoft Research Breakdancing and Ballet datasets [15] and sequences captured by our own FVV system. All video sequences were captured with Point Grey Research Flea2 1024×768 cameras at a frame rate of 15 FPS. Our cluster-based implementation processes each video in parallel and uses GPU's for match cost evaluation. In all the experiments, the same set of empirically tuned parameters are used: $\mu = 0.008$, smoothness weight scale $\rho = 4/3$, $\vartheta = 0.1$, $N = 256$, $\beta_1 = 0.35$, $\beta_2 = 0.95$, $\beta_3 = 0.65$, $\beta_4 = 0.85$ and the ZNCC window size is 5×5. $K = 3$ iterations are found enough for acceptable accuracy. The frames are downsized to 512×384 for speed. Furthermore, we use inter-view depth smoothing schemes as those used in [15].

9.5.1 Quantitative Temporal Consistency Evaluation

To quantitatively evaluate the effectiveness of our temporal consistency enforcement, we compare the depth stability of background regions throughout the whole video sequence. As was done in [170], for each pixel on a chosen static background scanline, we compute the median of 100 estimated depth values and the standard deviation (STD) from the median. The resultant curve and the 100 frame depth scanline stack for row 20 (equivalent to row 40 for full-size frames as that used in [170]) of view 3 of the Breakdancing dataset is shown in Figure 4. From the comparison to the corresponding curve and stack for [15], which process each frame independently, our curve is more stable with much smaller deviation and the depth stack is smoother. Large deviations mainly appear around depth discontinuities between the two walls[2]. So our depth profile shown as the black

[2] Please note the claim in [169] that the curve should be close to a flat line is incorrect since we can see easily in the snapshot in Figure 9.4 that the upper right brown wall is not on the same

curve is closer to the scene geometry qualitatively. Specifically, the average/min SD[3] over all pixels is 2.80/0.33 for [15], while 0.29/0.00 for our method, which means that our depth estimations for static pixels across all frames are more stable and hence, more consistent.

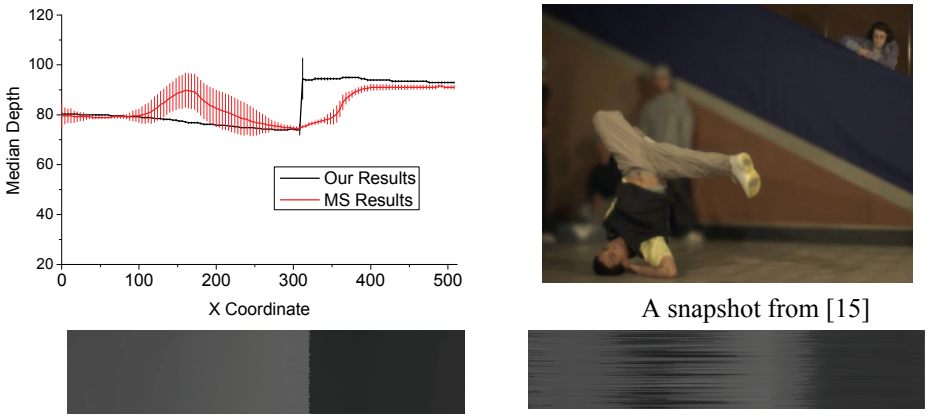


Figure 9.4: Median-Deviation curves and depth scaline stacks.

9.5.2 Experiments on MS Datasets

The publically available datasets and depth maps from [15] have been invaluable to the FVV research community as inputs and baseline results. The Breakdancing and Ballet datasets are particularly challenging due to large indoor textureless surfaces, surface reflections, and fast moving people.

Our depth map results for frames 78 and 79 of the Breakdancing dataset are illustrated in Figure 9.5 and 9.6 as examples. By comparison to the results of [15], we can see that even though both methods can achieve similar overall spatial accuracy, ours performs better in handling occlusions. Furthermore, in our results, the depth values of static and textureless regions of the scene such as the wall and

---

plane as the blue wall so that they should have different depths as shown in our results but not in theirs.

[3] Unlike in [169], which appears to use a different definition, the SD is defined as the square root of the variance.

floor are more temporally consistent than those shown in [15], which can be better observed when viewing the depth videos.

Similar performance is also observed for the Ballet dataset. As shown in Figure 9.7, compared to the results of [15], the quality of our depth map results is comparable if not better. Furthermore, the results are very consistent across different views and temporally.

9.5.3 Experiments on Datasets Captured by Our FVV System

In addition to the widely referred MS datasets, our proposed algorithm has also been integrated into our FVV system as a built-in service and extensively tested. In all the experiments, the camera array total calibration is done using the method introduced in Chapter 5.

Illustrated in Figure 9.8 is one example indoor scene with a person playing soccer in a lounge area. Similar to the MS datasets, this scene is also very challenging since most surfaces are textureless. Since only natural lighting is used, there are washed-out issues for bright or reflective surfaces due to the long exposure time, which also causes slight motion blur issue to the moving soccer ball in some frames. From the shown video frames, it can be seen that the simple photometric calibration is pretty accurate. The colors in all the views are very consistent. From the shown depth maps, though they are also accurate enough for FVV rendering, compared to the results we obtained for MS datasets, there are more errors. A possible reason could be the much larger depth range and viewpoint change due to the widely spaced camera arrangement compared to that used in the Breakdancing or Ballet datasets.

Illustrated in Figure 9.9 is another indoor scene example with a person playing basketball in a lab. Due to the limited lab space, only 8 cameras are used. As there is no natural lighting, to avoid the washed-out problem in some image regions due to the uneven fluorescent lamp lighting, a short exposure time (10ms) and high gain (5.0) are used, making the videos a little dark and noisy. Therefore bilateral smoothing is used before depth recovery. As can be seen from the corresponding depth maps, our algorithm is robust to video noise and performs well.

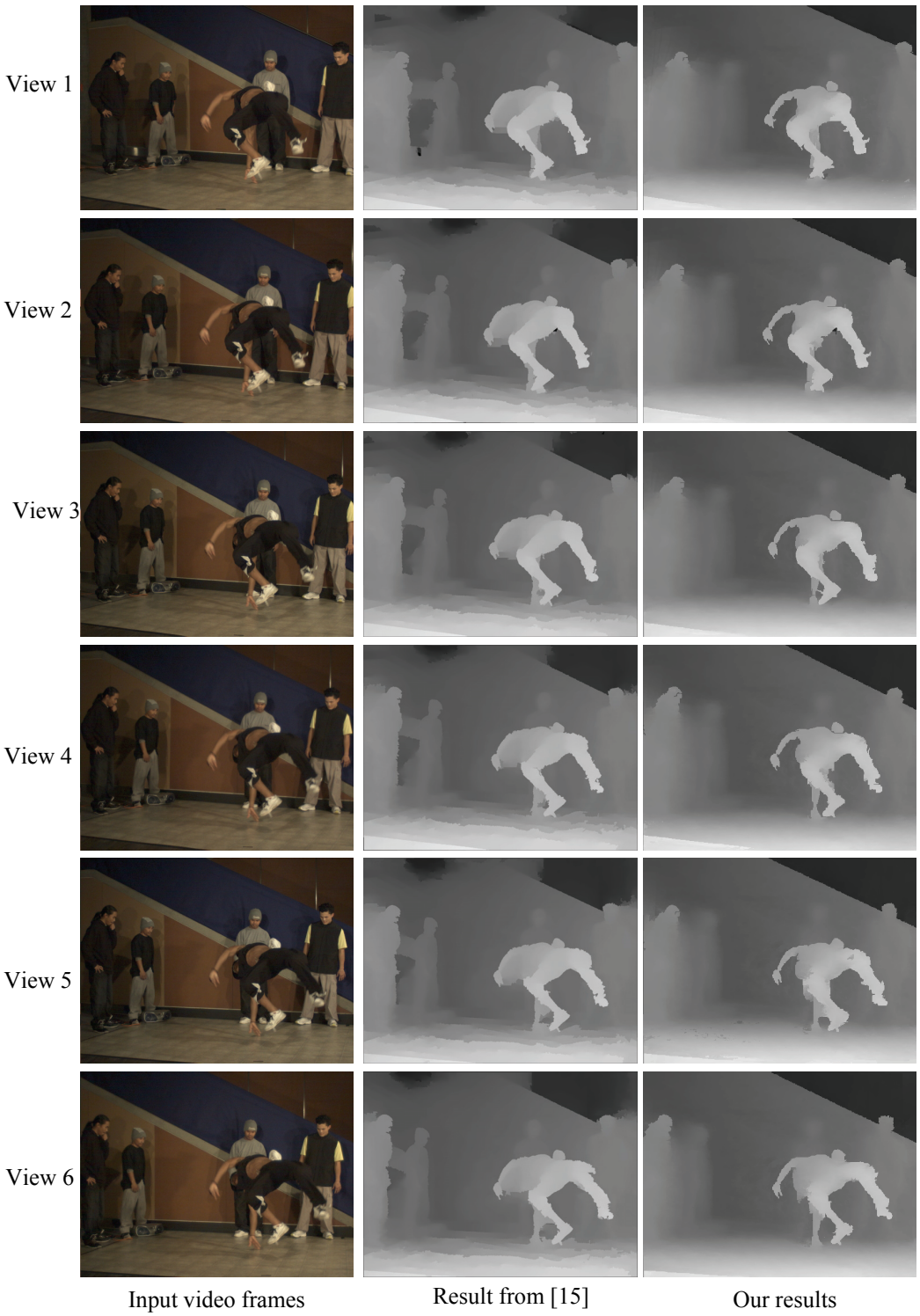


Figure 9.5: Qualitative comparison of our results to results from [15] on the Breakdancing dataset (frame 78).

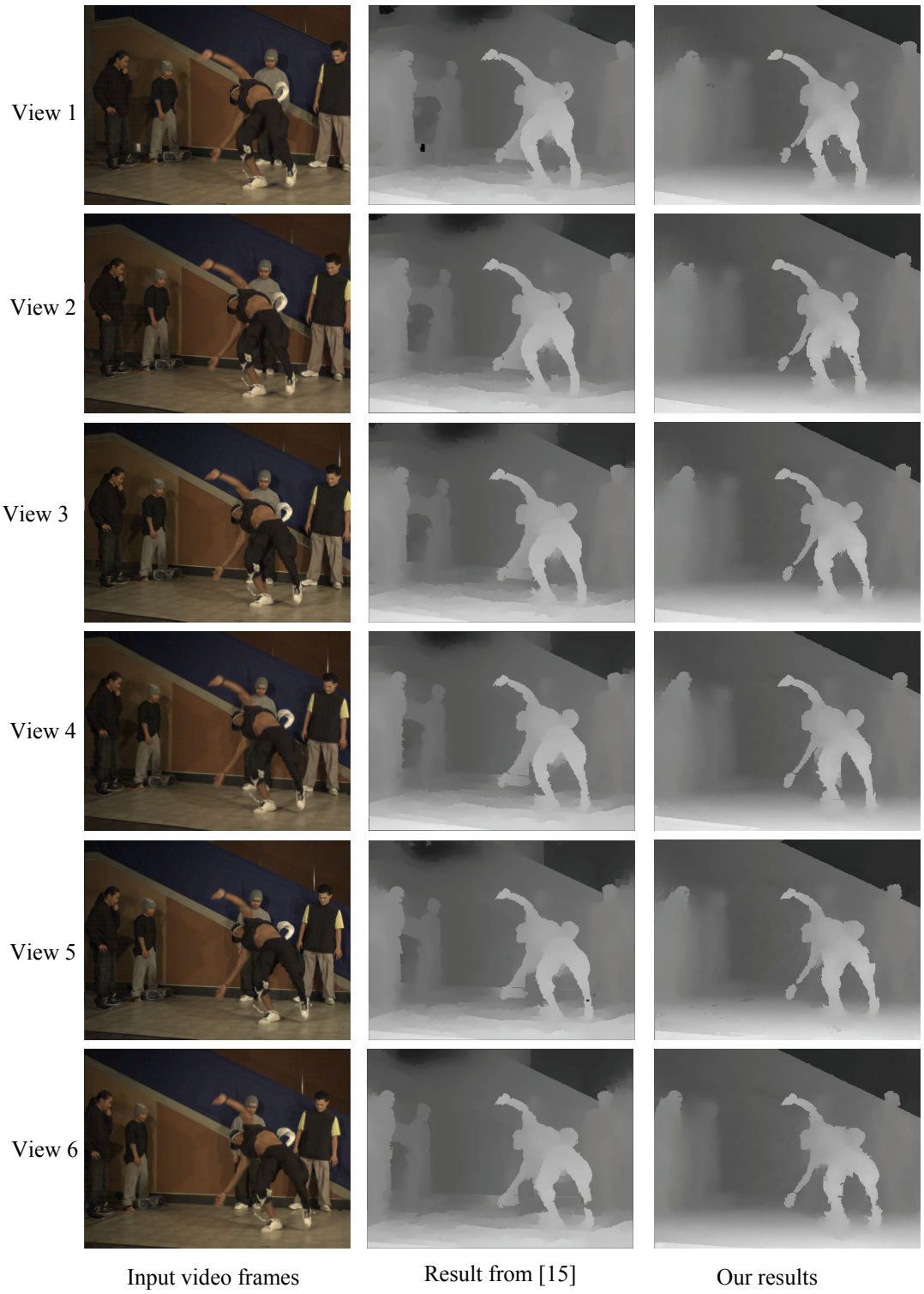


Figure 9.6: Qualitative comparison of our results to results from [15] on the Breakdancing dataset (frame 79).

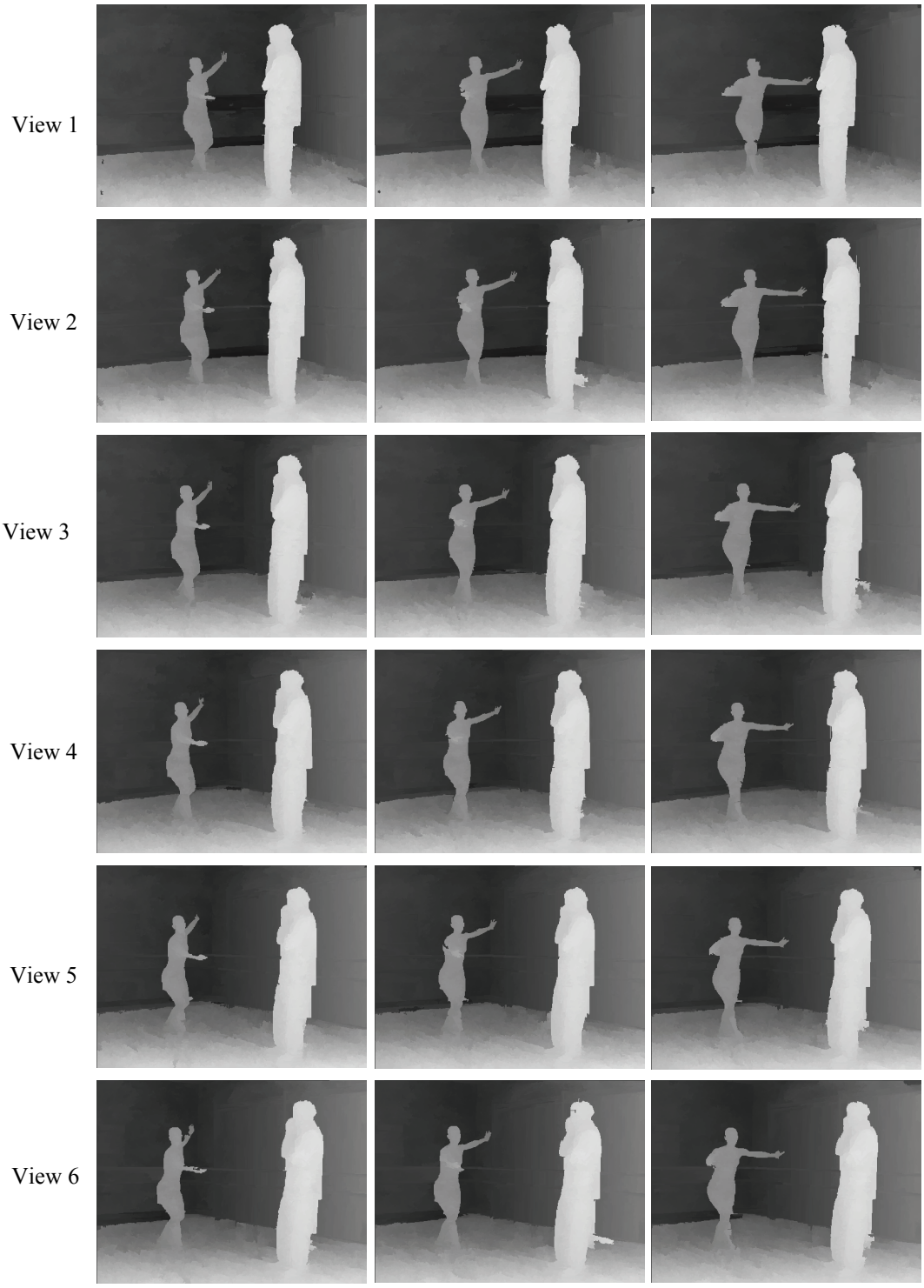


Figure 9.7: Our example results of the Ballet dataset (frames 2, 3, and 4).

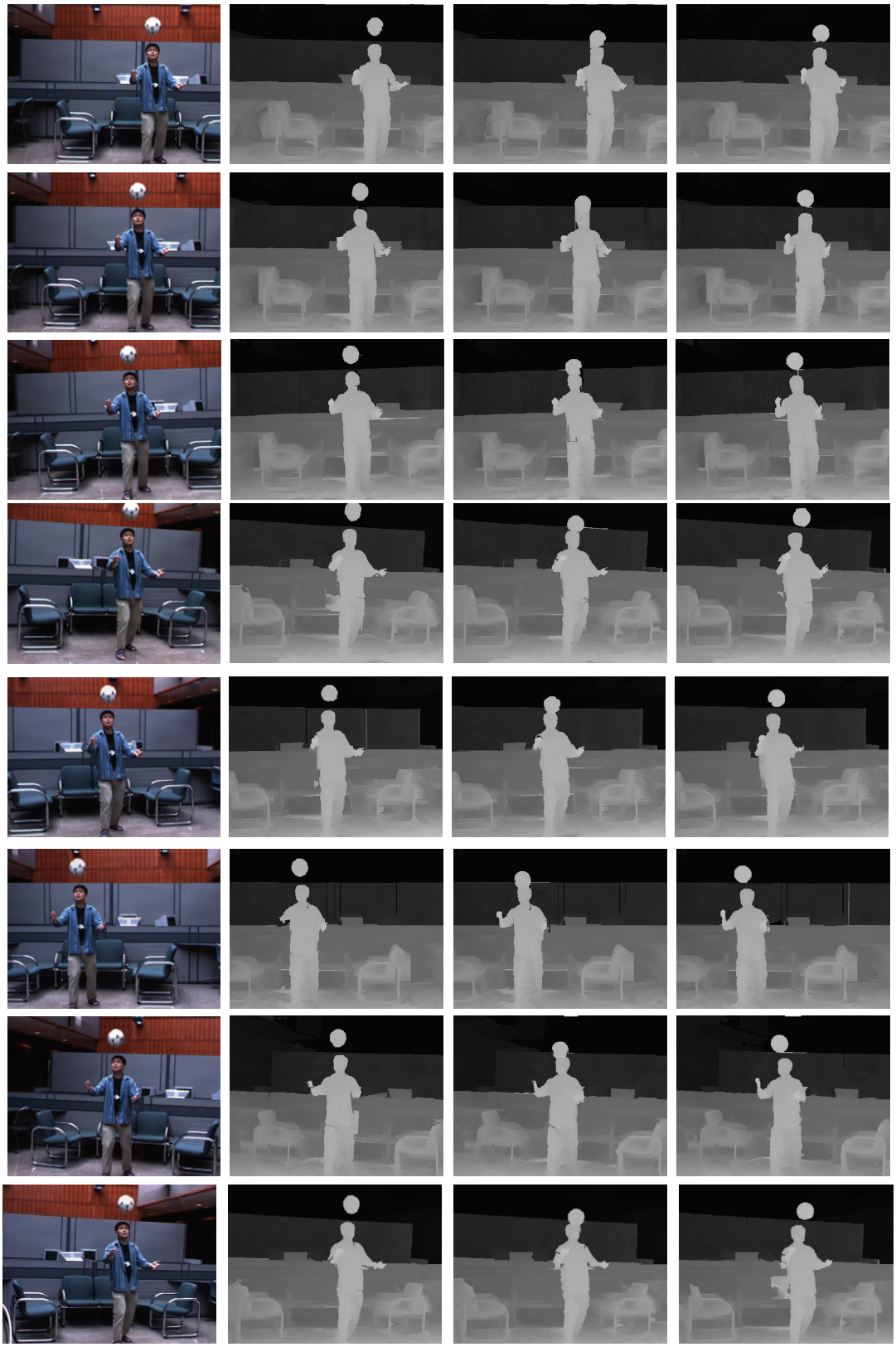


Figure 9.8: Results of a soccer scene dataset captured by our FVV system
(8 out of 16 views).

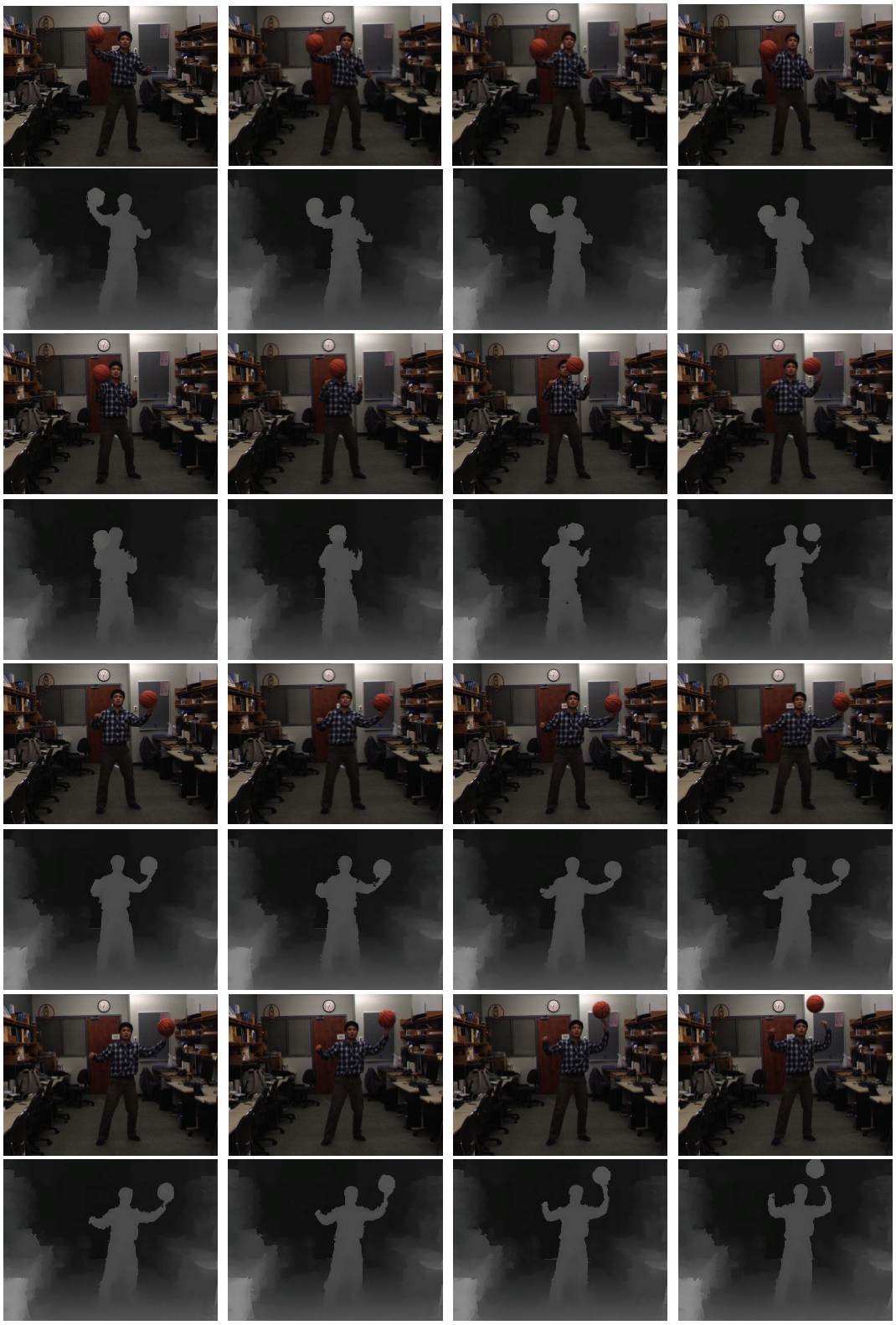


Figure 9.9: Consecutive depth map results (center view) of a Basketball dataset captured by our FVV system.

9.5.4 Rendering based Performance Evaluation and Profiling

(a) Real-time FVV rendering using depth based forward warping

Similar to [15], our FVV system also takes advantage of depth based forward 3D image warping to synthesize virtual views for FVV rendering. As shown in Figure 9.10, its workflow includes two main steps of warping and blending.

Specifically, given a novel view to synthesize, the two nearest source cameras are selected. For each of them, the corresponding depth map is used to warp the color video frame to the novel view. Due to occlusions, there must be some uncolored pixels in the warped images, which are shown as red pixels in Figure 9.10. To generate more complete image, these two warped images are blended together using the novel viewpoint to source camera distance information to define the blending ratio. For the two pixels to be blended, if one of them is uncolored, then the blended color is just the color of the other pixel if it is colored. If both pixels are uncolored, then the blended pixel remains uncolored. All the uncolored pixels form holes which are then filled using the color of neighbor's pixel.

The depth based 3D image warping is done by converting the depth map and the corresponding color image into a colored/textured 3D geometry first and then projecting them into the novel view. For this, two approaches have been implemented.

In the first point based one, all the pixels are independently back-projected into the reference 3D space using per-pixel depth information, forming a point cloud with each 3D point associated with the corresponding color attribute. This approach is fast and does not need special geometry processing. However, without further using more advanced point based rendering techniques such as splatting, the rendering quality is not good enough.

In contrast, the second approach converts the color image into a textured 3D mesh through depth based pixel back-projection. That is, each back-projected 3D point is not independent any more, but being a vertex connected to other vertices in a mesh. The advantage of this approach is that the projected geometry is more complete and all the gaps between projected pixels are automatically interpolated.

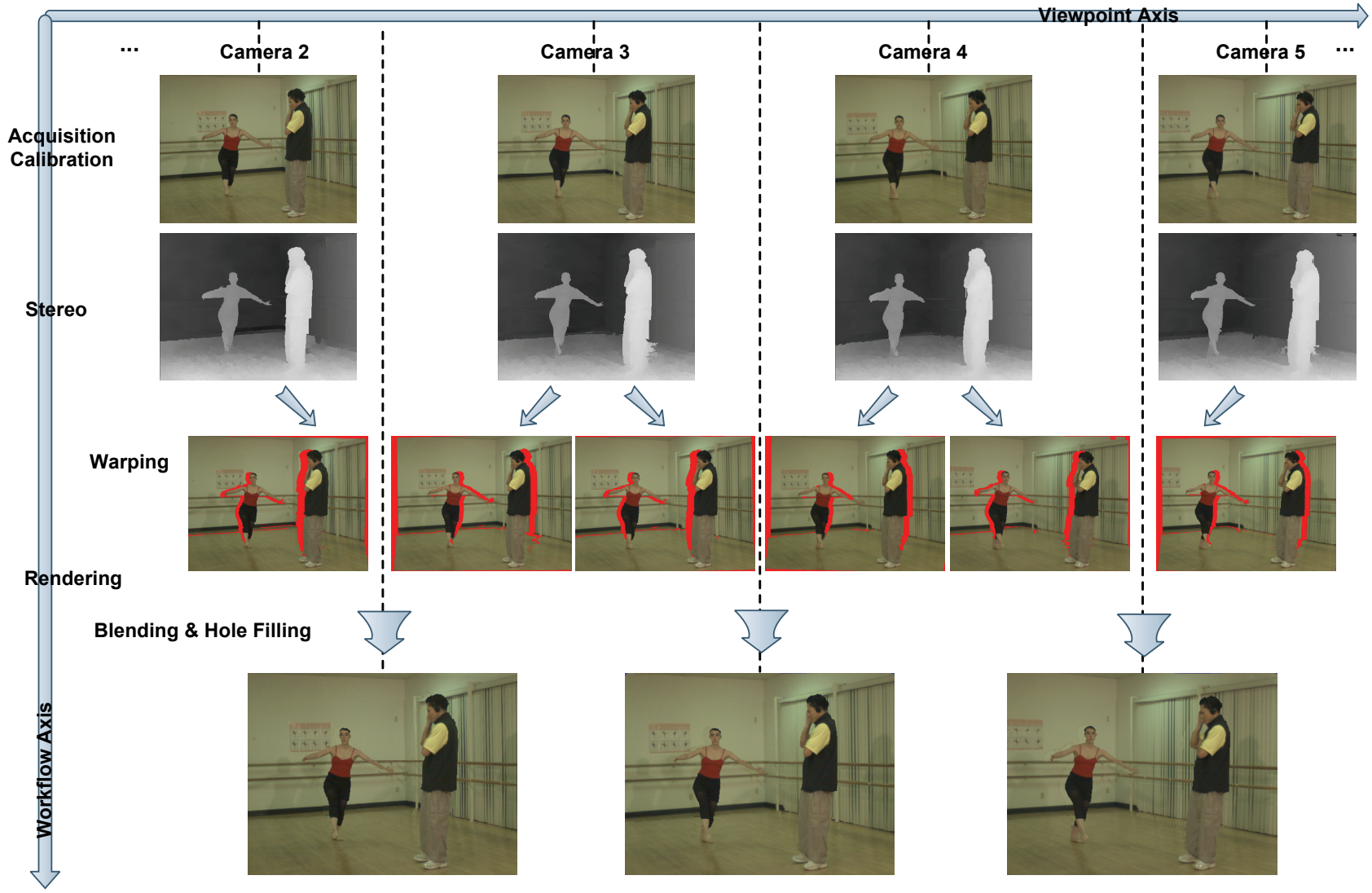


Figure 9.10: Illustration of forward warping based FVV rendering workflow.

Its main disadvantage is that mesh connectivity has to be updated for different views.

As shown in Figure 9.11, it can be seen that the mesh based approach gives better rendering quality than the point based one. Therefore in this thesis, we choose the mesh based approach. All the FVV rendering is implemented on the GPU using GLSL. In our current implementation, for better performance, we always use a fixed mesh formed from the pixel grid and update the mesh by discarding the triangles across big depth discontinuities using a geometry shader to avoid the so-called "rubber-sheet" rendering artifacts.

Mesh based FVV Rendering

Point based FVV Rendering

Figure 9.11: Quality comparison of FVV rendering using point based and mesh based approaches. It can be seen that mesh based rendering is sharper and has fewer artifacts.

(b) Rendering based performance evaluation

To further quantitatively evaluate the performance of our new FVV-oriented video depth recovery framework, FVV rendering based depth accuracy analysis is done.

In FVV rendering, the most important issue is to make the synthesized views visually plausible. Therefore, although there is no ground truth depth maps available, the difference between images synthesized using the corresponding depth maps and the actually observed ones provides a good measure for evaluation. However, please note that this measure is only suitable for FVV rendering applications since the synthesized views can still appear to be correct even though the recovered depth has errors.

Specifically, given a set of synchronized video frames $\widehat{\boldsymbol{I}}^t = \{\boldsymbol{I}_m^t | m = 0, \dots, M-1\}$ and their depth maps $\widehat{\boldsymbol{D}}^t = \{\boldsymbol{D}_m^t | m = 0, \dots, M-1\}$, each view can be synthesized as $\overline{\boldsymbol{I}_m^t}$ using the nearest two views generated using the above rendering process. The average of the sum and standard deviation (STD) of the absolute color difference $\left|\overline{\boldsymbol{I}_m^t} - \boldsymbol{I}_m^t\right|$ for all pixels can be used as the color plausibility measure for that view. For video sequences, the measure is further averaged for all the time instants. In addition to color difference, depth plausibility can be evaluated similarly through differencing the synthesized depth map $\overline{\boldsymbol{D}_m^t}$ and $\boldsymbol{D}_m^t$. The depth difference is usually normalized w.r.t. the depth range. However, it should be noted that using estimated $\boldsymbol{D}_m^t$ as the depth ground truth may bias the measure unless it is very accurate.

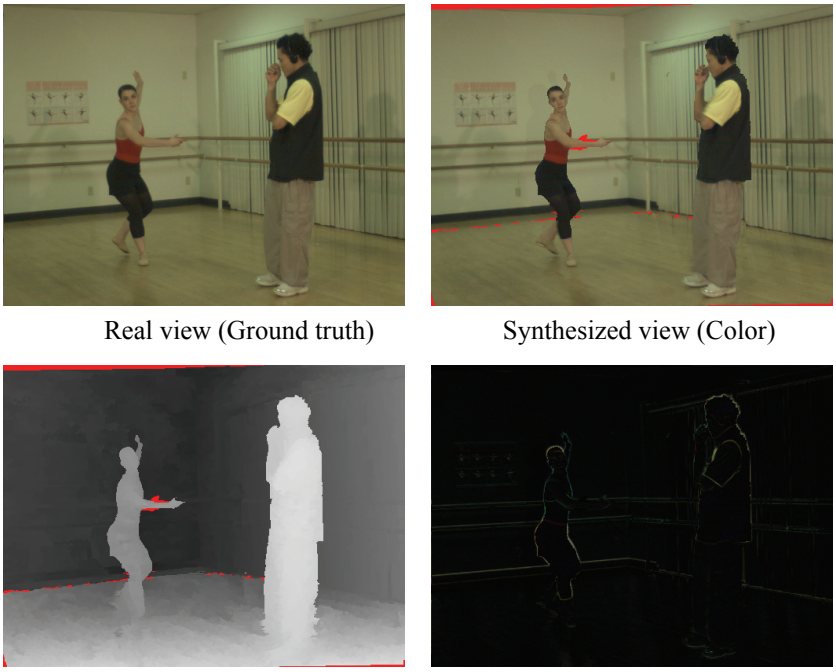
We profile the above defined color and depth plausibility for the Breakdancing, Ballet and Basketball datasets, and for each of them the spatial-temporal consistent depth recovery is done multiple times using different parameters. Only sub-sequences of 25 frames are used for faster profiling. The profiled parameters include:

(a) the number of depth levels Z

(b) Smoothness penalty exempt threshold scale μ

(c) Smoothness weight scale ρ

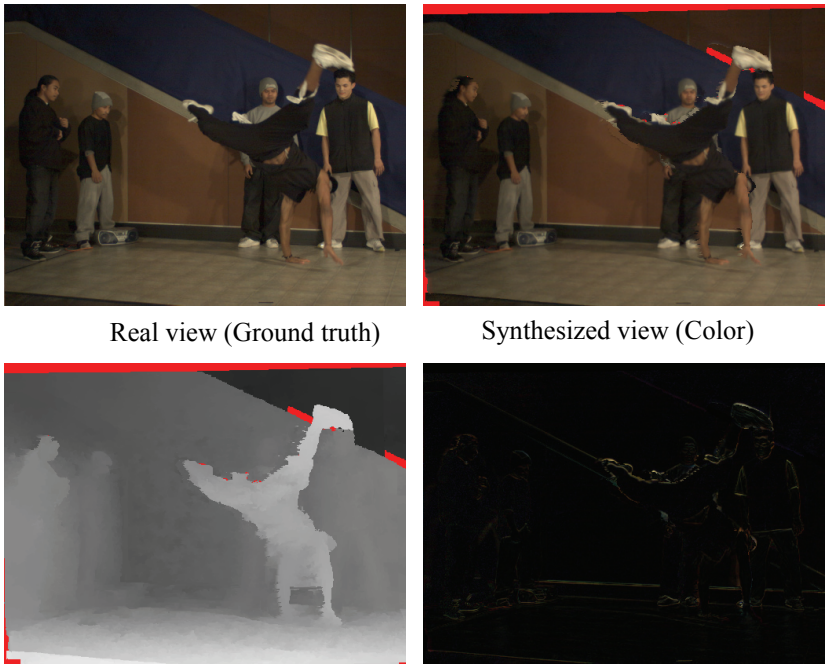
Example synthesized color/depth images and the corresponding color difference images are shown in Figures 9.12-9.14. Please note in the following shown synthesized color or depth images, the red color denotes pixels with no color information, which is mainly due to occlusions or depth errors. These pixels will not be counted in the color difference statistics. In the color difference images, the brighter the color, the larger the difference. As can be easily seen, most color errors appear on the depth discontinuities.



Real view (Ground truth) Synthesized view (Color)

Synthesized view (Depth) Color difference

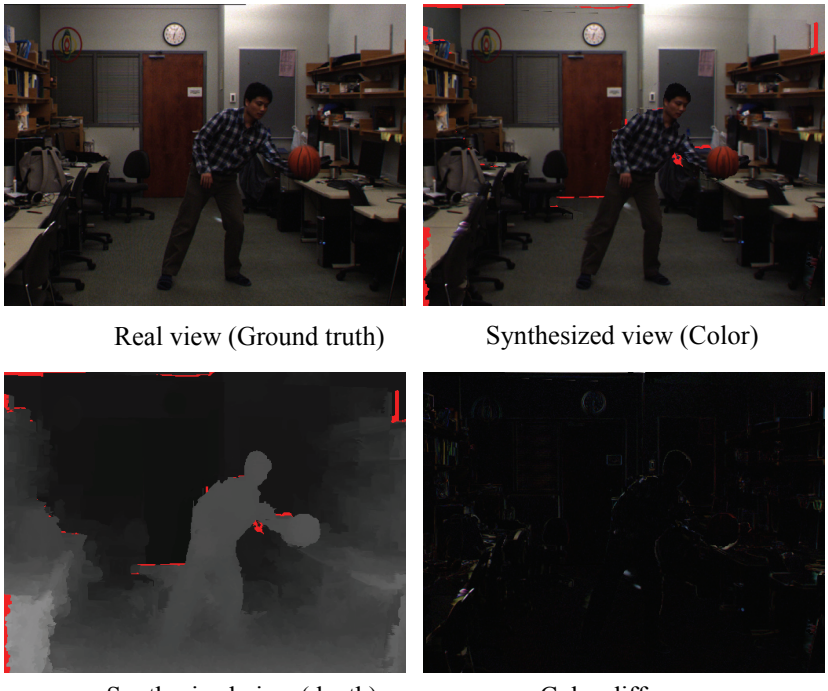
Figure 9.12: Example of rendering evaluation of the Ballet dataset.



Real view (Ground truth) Synthesized view (Color)

Synthesized view (depth) Color difference

Figure 9.13: Example of rendering evaluation of the Breakdancing dataset.



Real view (Ground truth) Synthesized view (Color)

Synthesized view (depth) Color difference

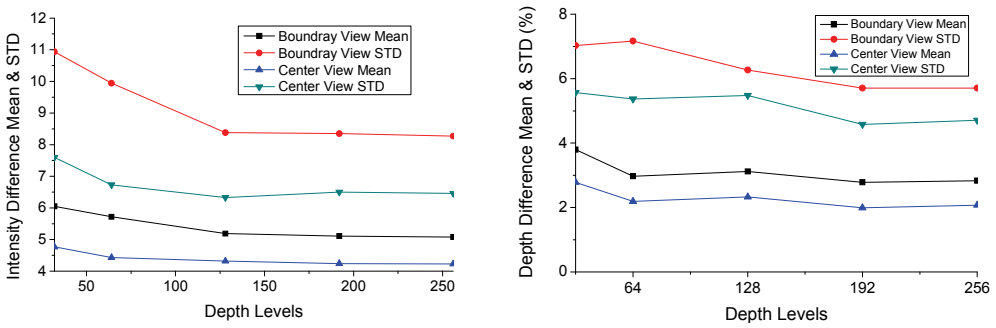
Figure 9.14: Example of rendering evaluation of the Basketball dataset.

Shown in Figure 9.15 are the corresponding parameter profiling curves. Please note that there is a set of curves for each individual view so that it is not possible to show all of them here. However, most of them share very similar shapes and hence only the representative ones are presented.

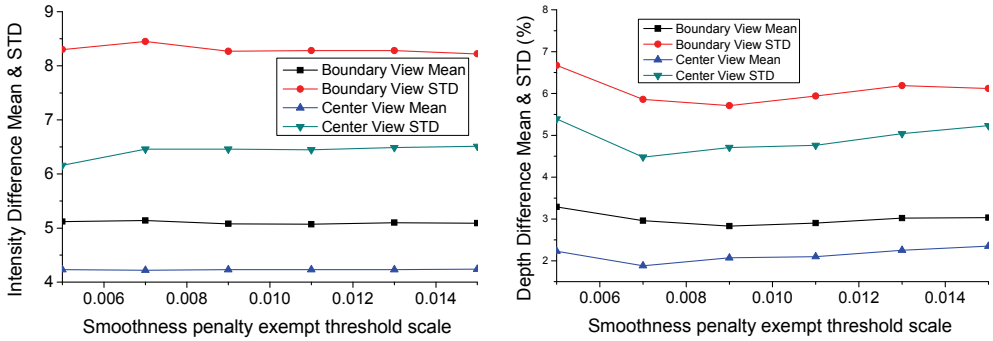
To investigate the relationship between viewpoint location and depth accuracy and rendering quality, in each sub-figure of Figure 9.15, two views are compared. One is a boundary view where the remaining views are on the same side. The other one is the center view or the reference view. From the curves it can be seen that the center view usually can have higher depth accuracy and rendering quality compare to the boundary one. Moreover, both views behave similarly to parameters changes. For example, as shown in the top row, as the depth level number increases, the finer depth resolution help to improve the accuracy. However, the improvement becomes very small up to some point. Although the color difference statistics information changes very little w.r.t. different smoothness penalty exempt threshold scales, from the depth difference statistics sub-figure, it can be seen that setting it as 0.008 results in better performance. Finally as for the smoothness weight scale parameter, the center view is more sensitive to this parameter than the boundary view is. Relatively large scale value gives slightly better results.

Summary

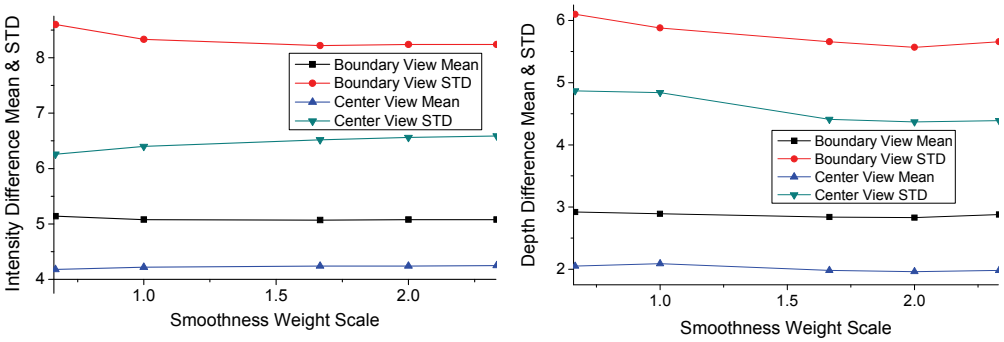
In this chapter, the region-tree based binocular stereo matching and optical flow estimation presented in the last two chapters are combined together into a unified framework for spatial-temporal consistent video depth-based FVV scene reconstruction. By using the new temporal region-tree representation and inconsistency map based progressive optimization, high quality video depth can be generated efficiently and robustly, which is indispensible to our FVV system. Extensive video depth recovery and FVV rendering experiments have shown its effectiveness.



Depth level number profiling



Penalty exempt threshold scale profiling



Smoothness weight scale profiling

Figure 9.15: Representative parameter profiling curves using rendering based evaluation.

Chapter 10
Conclusions and Future Work

The driving motivation of this thesis is to provide a convenient and efficient hardware and software solution for free viewpoint video content creation and rendering using multiple cameras. Since there are too many scientific challenges to achieve this goal, we particularly focused more on the camera array based FVV scene acquisition and depth based FVV scene reconstruction. In the following, the main technical contributions of this thesis in these two areas are summarized first, followed by their respective limitations and possible future works.

10.1 Contributions

The main contributions of this thesis include: efficient camera array calibrations, a new region-tree based image labeling framework and a new cluster based FVV system (platform).

(1) Efficient and accurate camera array total calibrations

By extending the classic plane based single camera geometric calibration method [28], our automatic camera array total calibration algorithm enables efficient and accurate geometric, photometric and temporal calibrations. The traditional static scene oriented bundle-adjustment optimization is enhanced to further handle temporal parameters, through which unsynchronized cameras can be calibrated more conveniently. By enabling the use of unsynchronized cameras and performing the geometric and photometric calibrations simultaneously, camera array based multiple view video acquisition is made easier and much more efficient.

(2) Region tree based labeling framework

As one of the main contributions of this thesis work, our new proposed region-tree based image representation provides a good combination of traditional pixel based and layer based ones. By taking advantage of image over-segmentation, the bests of both worlds are nicely combined. Through the new representation, the desired trade-off between enabling image primitives to contain enough information with a large support area and reducing the risk of violating the layer parameterization assumption with a small support area is achieved. The spanning tree structure further provides many advantageous properties for fast optimization and image manipulation. The region-tree image representation is also extensible. Different variants such as the coarse-to-fine and temporal ones are also been presented for different application requirements.

Based on the region-tree representation, a general discrete image labeling framework is presented. Its successful applications in solving two classic computer vision problems – binocular stereo matching and optical flow estimation have shown its unique potentials and versatility.

In the spatial-temporal consistent video depth recovery application which is of crucial importance to depth based FVV rendering, our region-tree framework also shows very promising performance. The temporal consistency between temporally consecutive depth maps is enforced through the temporal region-tree variant which is used for integrating optical flow estimation with general position multi-view stereo matching. While for the spatial consistency, it is enforced using the new inconsistency map based progressive optimization.

(3) Cluster based FVV system design and implementation

As another main contribution, a cluster based FVV system has been developed, with its hardware and software architectures designed towards more general camera array applications. With the developed system, several highly desired features such as centralized workflow management, distributed data storage and transparent data access are enabled. Powerful computation

performance is also provided by taking advantage of the parallel computing and hardware acceleration using the GPU's.

The system is highly modularized and use service provider/consumer model. New functionalities can be easily extended or incorporated. Hence, the system can be used as a platform or test bed for prototyping new FVV algorithms. By integrating our camera array total calibration technique and parallelizing our region-based FVV depth scene reconstruction on cluster, the whole FVV process from the multi-view video acquisition to the final rendering can be done efficiently and in a highly automatic way. Extensive practical experiments have shown its robustness and effectiveness.

10.2 Limitations and Future Work

One limitation of our camera total calibration method is the use of relatively simple linear color transform model for color normalization. More advanced models can be investigated as one possible future work. Although in most experiments the current color normalization appears good enough for stereo matching and optical flow estimation, better normalization accuracy is expected to further improve their performance.

As for the region-tree framework, its main limitation is in its relatively inferior performance for large images due to the use of mean-shift segmentation. Also for the case of requiring large-size label set such as in optical flow estimation, the optimization can become slow due to the cubic computational complexity of DP (though it can still be faster than other global optimization methods such as graph cuts). In the future, different ways to further utilize the image pyramid approach for faster speed up can be investigated. For example, segmentation result from down-sized image could be used to speedup full-size image segmentation. Also labeling result of down-sized levels could be further used to limit the label set size.

On the other hand, the current region-tree framework implementation uses the constant region label model, i.e., all the pixels in a region share the same label. In the future, different region label distribution model such as the affine model can

be investigated. Moreover, currently for video depth recovery, each video frames are segmented independently which is not time economical and also cannot maintain segmentation consistency between consecutive frames. In the future, a possible approach for incremental region-tree generation can be investigated to take advantage of the region-tree information of the last frame.

As for possible future work of our FVV system, a real-time FVV scene reconstruction module can be developed and integrated. More advanced FVV rendering techniques such as the alpha matting based method [15] could be implemented. Also more camera array applications can be developed by taking advantage of current system modules or services.

References

[1] "Introduction to the special issue on immersive telecommunications," *IEEE Trans. on Circuits and Systems for Video Technology*, 14(3), 2004.

[2] O. Schreer, P. Kauff, T. Sikora, "3D video communication: algorithms, concepts and real-time systems in human centred communication," *Wiley Press*, 2005.

[3] S. L. Hill, "Scalable multi-view stereo camera array for real world real-time image capture and three-dimensional displays", M.Sc. Thesis, MIT, 2004.

[4] ISO/IEC JTC1/SC29/WG11, "Report of 3DAV exploration," *Doc. N5878*, Trondheim, Norway, July 2003.

[5] ISO/IEC MPEG & ITU-T VCEG, "Joint Draft 1.0 on multiview video coding," JVT-U209, Nov. 2006.

[6] S.B. Gokturk, H. Yalcin, and C. Bamji, "A time-of-flight depth sensor, system description, issues and solutions," in *Proc. of IEEE Conf. Computer Vision and Pattern Recognition,* Washington, DC, 2004, pp. 35.

[7] C. Lei, J. Selzer, and Y.H. Yang, "Region-tree based stereo using dynamic programming optimization," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, New York, NY, USA, June 17-22, 2006, pp. 2378-2385.

[8] C. Lei and Y.H. Yang, "Optical flow estimation on coarse-to-fine region-trees using discrete optimization," in *Proc. of International Conf. on Computer Vision*, Kyoto, Japan, Sept. 27-Oct. 2, 2009.

[9] C. Lei, X. Chen, and Y.H. Yang, "A new multiview spacetime-consistent depth recovery framework for free viewpoint video rendering," in *Proc. of International Conf. on Computer Vision*, Kyoto, Japan, Sept. 27-Oct. 2, 2009.

[10] C. Lei and Y.H. Yang, "Efficient geometric, photometric, and temporal calibration of an array of unsynchronized video cameras," in *Proc. of the 6th Canadian Conference on Computer and Robot Vision*, Kelowna, BC, May 25-27, 2009.

[11] C. Lei and Y.H. Yang, "Design and implementation of a cluster-based smart camera array application framework," in *Proc. Second ACM/IEEE International Conference on Distributed Smart Cameras*, Stanford, California, September 7-11, 2008.

[12] G. Cheung, T. Kanade, J. Bouguet, and M. Holler, "A real time system for robust 3-D voxel reconstruction of human motions," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2000, pp. 714-720.

[13] W. Matusik, C. Buheler, R. Raskar, S. Gortler, and L. McMillan, "Image-based visual hulls," in *Proc. of ACM SIGGRAPH 2000*, pp. 369-374.

[14] M. Gross, S. Wuermlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, K. Strehlke, A. Vande Moere, and O. Staadt, "Blue-c: A spatially immersive display and 3D video portal for telepresence," in *Proc. of ACM SIGGRAPH 2003*, July 2003, pp. 819-827.

[15] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," in *Proc. of ACM SIGGRAPH 2004*, Los Angeles, CA, USA, August 2004.

[16] B. Wilburn, N. Joshi, V. Vaish, E. V. Talvala, E. Antunez, A. Barth, A. Adams, M. Levoy, and M. Horowitz, "High performance imaging using large camera arrays," in *Proc. of ACM SIGGRAPH 2005*, pp. 765-776.

[17] P. Kauff and O. Schreer, "An immersive 3-D video-conferencing system using shared virtual team user environments," in *Proc. of ACM Collaborative Virtual Environments*, 2002, pp. 105-112.

[18] H. Baker, D. Tanguay, I. Sobel, D. Gelb, M. Gross, W. Culbertson, and T. Malzbender, "The Coliseum immersive teleconferencing system," in *Proc. of Int. Workshop Immersive Telepresence*, Juan-les-Pins, France, 2002.

[19] M. Tanimoto, "Overview of free viewpoint television," *Signal Processing: Image Communication*, vol. 21, no. 6, pp. 454-461, July 2006.

[20] J. G. Lou, H. Cai, and J. Li, "A real-time interactive multi-view video system," in *Proc. of ACM Multimedia 2005*, pp.161-170.

[21] J. Carranza, C. Theobalt, M. Magnor, and H.-P. Seidel, "Free-viewpoint video of human actors," in *Proc. of ACM SIGGRAPH 2003*, vol. 22, no. 3, July 2003, pp. 569-577.

[22] Point Grey Research Sync Unit, http://www.ptgrey.com/products/sync/

[23] G. Litos, X. Zabulis, and G. Triantafyllidis, "Synchronous image acquisition based on network synchronization," in *IEEE Workshop on Three-Dimensional Cinematography*, 2006.

[24] Multi-Sync, http://www.ptgrey.com/products/multisync/index.asp

[25] T. Svoboda, H. Hug, and L. van Gool, "ViRoom - Low cost synchronized multicamera system and its self-calibration," in *Pattern Recognition, DAGM Symposium*, 2002, pp. 515-522.

[26] W. Faig, "Calibration of close-range photogrammetry systems: Mathematical formulation," *Photogrammetric Engineering and Remote Sensing*, 41(12):1479-1486, 1975.

[27] 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): 323-344, Aug. 1987.

[28] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330-1334, 2000.

[29] S. J. Maybank and O. D. Faugeras, "A theory of self-calibration of a moving camera," *The International Journal of ComputerVision*, 8(2):123-152, Aug. 1992.

[30] Q. T. Luong and O. Faugeras, "Self-calibration of a moving camera from point correspondences and fundamental matrices," *International Journal of Computer Vision*, 22(3):261-289, 1997.

[31] Intel OpenCV Library, http://www.intel.com/technology/computing/opencv/

[32] Camera Calibration Toolbox for Matlab, http://ww.vision.caltech.edu/bouguetj/

[33] L. Lee, R. Romano, and G. Stein, "Monitoring activities from multiple video streams: establishing a common coordinate frame," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):758-767, August 2000.

[34] V. Vaish, "Light Field Camera Calibration," http://graphics.stanford.edu/projects/array/geomcalib/

[35] P. T. Baker and Y. Aloimonos, "Complete calibration of a multi-camera network," *in Proc. of the IEEE workshop on Omnidirectional Vision*, Hiton Head Island, SC, 2000, pp.134-141.

[36] P. T. Baker and Y. Aloimonos, "Calibration of a multicamera network," *in Proc. of the IEEE workshop on Omnidirectional Vision*, 2003.

[37] S. Prince, A. D. Cheok, F. Farbiz, T.Williamson, N. Johnson, M. Billinghurst, and H. Kato, "3D live: Real time captured content for mixed reality," in *International Symposium on Mixed and Augmented Reality (ISMAR'02)*, 2002, pp. 7-13.

[38] T. Svoboda, D. Martinec, and Tomas Pajdla. "A convenient multi-camera self-calibration for virtual environments," *PRESENCE: Teleoperators and Virtual Environments*, 14(4): 407-422, MIT Press, August 2005.

[39] X. Chen, J.E. Davis, and P. Slusallek, "Wide area camera calibration using virtual calibration objects," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2000, Vol. II , pp.520-527.

[40] V. Vaish, B. Wilburn, N. Joshi, and M. Levoy, "Using plane + parallax for calibrating dense camera arrays," *in Proc. of IEEE Conference on Computer Vision and Pattern Recognition 2004*, Vol. I, pp.2-9, 2004.

[41] N. Joshi, "Color calibration for arrays of inexpensive image sensors," *Stanford University, Technical Report CSTR 2004-02*, 2004.

[42] A. Ilie, G. Welch, "Ensuring color consistency across multiple cameras," *in Proc. of the 10th IEEE Conference on Computer Vision (ICCV)*, Vol. 2:1268-1275, Oct. 17-20, Beijing, China, 2005.

[43] C. Zhang and T. Chen, "A survey on image-based rendering – representation, sampling and compression," *EURASIP Signal Processing: Image Communication*, pp. 1-28, Vol. 19, No. 1, Jan 2004.

[44] L. McMillan, "Plenoptic modeling: An image based rendering system," in *Proc. of ACM SIGGRAPH 1995*, ACM Press, 1995, pp 39-46.

[45] M. Levoy and P. Hanrahan, "Light filed rendering," in *Proc. of ACM SIGGRAPH 1996*, ACM Press, 1996, pp. 31-42.

[46] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen, "The lumigraph," in *Proc. of ACM SIGGRAPH 1996*, ACM Press, 1996, pp. 43-54.

[47] J. C. Yang, M. Everett, C. Buehler and L. McMillan, "A real-time distributed light field camera," in *Proc. of the 13th Eurographics workshop on Rendering*, Eurographics Association, 2002, pp.77-86.

[48] T. Fujii and Tanimoto, "Free-viewpoint television based on the ray-space representation," *SPIE ITCom*, 2002, pp. 175-189.

[49] P. Bangchang, T. Fujii and Tanimoto, "Ray-space data compression using spatial and temporal disparity compensation," in *Proc of IWAIT 2004*, Singapore, 2004, pp. 171-175.

[50] M. Panahpour, P. Bangchang, T. Fujii and Tanimoto, "The optimization of distributed processing for arbitrary view generation in camera sensor networks," *IEICE Transactions on Fundamentals of Electronics, Communication and Computer Sciences*, E87-A, 8, pp. 1863-1870, 2004.

[51] A. Lumsdaine, T. Georgiev, "The focused plenoptic camera," in *Proc. of ICCP*, April 2009.

[52] J. C. Miller, "Computer graphics principles and practice," second edition, *Computers & Graphics* 16(2): 239-240 (1992).

[53] A. Khodakovsky, P. Schroeder, and W. Sweldens, "Progressive geometry compression," in *Proc. of ACM SIGGRAPH*, 2000, pp. 271-278.

[54] D. P. Luebke, "View-dependent simplification of arbitrary polygonal environments," Doctoral Dissertation, University of North Carolina at Chapel Hill, 1998.

[55] D. DeCarlo and J. Gallier, "Topological evolution of surfaces," in *Proc of Conf. of Graphics Interface 1996*, May 1996, pp. 194-203.

[56] A. Lee, D. Dobkin, W. Sweldens and P. Schröder, "Multiresolution mesh morphing," in *Proc. of ACM SIGGRAPH 1999*, August 1999, pp. 343-350.

[57] S. Kircher and M. Garland, "Progressive multi-resolution meshes for deforming surfaces," in *Proc. of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer Animation*, pp.191-200.

[58] M. Pollefeys, "Self-calibration and metric 3d reconstruction from uncalibrated image sequences," Ph.D thesis, 1999.

[59] P. E. Debevec, C. J. Taylor and J. Malik, "Modeling and rendering architecture from photographs," in *Proc. of ACM SIGGRAPH 1996*, August 1996.

[60] P. E. Debevec, Y. Yu, and G. D. Borshukov, "Efficient view dependent image-based rendering with projective texture mapping," in *Proc. Eurographics Rendering Workshop*, June 1998, pp. 105-116.

[61] A. Kaufman. and E. Shimony, "3D scan-conversion algorithms for voxel-based graphics," in *Proc. of ACM Workshop on Interactive 3D Graphics*, Chapel Hill, NC, October 1986, pp. 45-76.

[62] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *International Journal of Computer Vision*, Marr Prize Special Issue, 38(3):199-218, 2000.

[63] S. Vedula, S. Baker, S. Seitz, and T. Kanade, "Shape and motion carving in 6D," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2000.

[64] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 1997, pp. 1067-1073.

[65] S. M. Seitz and K. N. Kutulakos, "Plenoptic image editing," in *Proc. 6th Int. Conf. of Computer Vision*, 1998, pp.17-24.

[66] W. B. Culbertson, T. Malzbender, G. G. Slabaugh, "Generalized voxel coloring," *Workshop on Vision Algorithms 1999*, pp.100-115.

[67] D. Snow, P. Viola, R. Zabih, "Exact voxel occupancy with graph cuts," in *Proc. of IEEE International Conf. of Pattern Recognition*, Vancouver, Canada, 2000.

[68] M. Botsch, A. Wiratanaya, and L. Kobbelt, "Efficient high quality rendering of point sampled geometry," in *Proc of Eurographics Workshop on Rendering*, 2002, pp. 53–64.

[69] R. L. Carceroni, K. N. Kutulakos, "Multi-view scene capture by surfel sampling: From video streams to non-rigid 3D motion, shape and reflectance," *International Journal of Computer Vision* 49(2-3): 175-214 (2002).

[70] S. Würmlin, "Dynamic point samples as primitives for free-viewpoint video," Ph.D. Thesis, No. 15643, Department of Computer Science, ETH Zürich, Switzerland, 2004.

[71] M. Zwicker, H. Pfister, J. Baar, M. H. Gross, "Surface splatting," in *Proc. of ACM SIGGRAPH 2001*, 2001, pp. 371-378.

[72] R. J. Watt, and B.J. Rogers, "Human vision and cognitive science," in *Cognitive Psychology Research Directions in Cognitive Science: European Perspectives*, vol. 1, pp. 10-12, East Sussex: Lawrence Erlbaum Associates, 1989.

[73] M. Gong and Y.H. Yang, "Camera field rendering of static and dynamic scenes," *Graphical Models*, Vol. 67, pp. 73-99, 2005.

[74] C. Fehn, "Depth-image-based rendering (DIBR): compression and transmission for as new approach on 3D-TV," in *Proc. of Conf. on Stereoscopic Displays and Applications*, San Jose, CA, USA, 2004.

[75] Trimble 3D Scanning, http://www.trimble.com/

[76] Z-Cam, http://www.3dvsystems.com/

[77] Q. Li, H. J Feng, Z. H. Xu, H. Q. Huang, "Application of color-encoded structure light in 3D vision technology," in *Proc. SPIE Vol. 4922, Color Science and Imaging Technologies*, 2002, pp. 112-116.

[78] L. McMillan and G. Bishop, "Shape as a perturbation to projective mapping," *UNC Computer Science Technical Report TR95-046*, University of North Carolina, April 1995.

[79] L. McMillan, "An image-based approach to three-dimensional computer graphics," Ph.D. Dissertation, University of North Carolina, April 1997.

[80] W. R. Mark, L. McMillan and G. Bishop, "Post-rendering 3D warping," in *Proc. of Symp. on Interactive 3D Graphics*, pp.7-16, 1997.

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

[82] V. Kolmogorov and R. Zabih, "Multi-camera scene reconstruction via graph cuts," in *Proc. of European Conference on Computer Vision 2002*.

[83] A. M. Waxman and J. H. Duncan, "Binocular image flows: Steps toward stereo-motion fusion," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(6):715-729, Nov. 1986.

[84] Nasrabadi, M. Nasser, Clifford, P. Sandra, Y. Liu, "Integration of stereo vision and optical flow by using an energy-minimization approach," *Journal of the Optical Society of America Series A: Optics, Image Science, and Vision*, 6(6), pp.900-907, 1989.

[85] G. Sudhir, S. Banerjee, R. Bahl, and K. Biswas, "A Cooperative integration of stereopsis and optic flow computation," *Journal of the Optical Society of America Series A: Optics, Image Science, and Vision*, vol. 12, pp. 2564, 1995.

[86] M. Isard, and J. MacCormick, "Dense motion and disparity estimation via loopy belief propagation," in *Proc. of Asian Conference on Computer vision*, 2006, vol.2, pp 32-41.

[87] J. Starck and A. Hilton, "Free-viewpoint video for interactive character animation", *in Proc. of 4th. Symposium on Intelligent Media Integration for Social Information Infrastructure*, Nagoya JAPAN, 2006, pp. 25-30.

[88] M. Naef, O. Staadt, and M. Gross, "Blue-C API: A multimedia and 3D video enhanced toolkit for collaborative VR and telepresence," in *Proc. ACM SIGGRAPH Int'l Conf. Virtual Reality Continuum and Its Applications in Industry (VRCAI '04)*, pp. 11-18, 2004.

[89] http://myrinet.com

[90] http://www.ptgrey.com

[91] http://www-unix.mcs.anl.gov/mpi/

[92] B. Quinn and D. Shute, "Windows sockets network programming," *Addison-Wesley*, MA, ISBN: 0-201-63372-8.

[93] G. P. Stein, "Tracking from multiple view points: Self-calibration of space and time," in *DARPA IU Workshop*, 1998, pp. 521-527.

[94] R.L. Carceroni, F.L.C. Pádua, G.A.M.R. Santos, and K. N. Kutulakos, "Linear sequence-to-sequence alignment," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2004, pp. 746-753.

[95] Y. Caspi, D. Simakov, and M. Irani, "Feature-based sequence-to-sequence matching," in *Proc. of Workshop on Vision and Modeling of Dynamic Scenes,* Copenhagen, Denmark, May 2002.

[96] D. W. Pooley, M. J. Brooks, A. van den Hengel, and W. Chojnacki, "A voting scheme for estimating the synchrony of moving camera videos," *in Proc. of Int. Conf. on Image Pro.*, Barcelona, Sept. 2003, pp. 413-416.

[97] C. Rao, A. Gritai, M. Shah, and T. Syeda-Mahmood, "View-invariant alignment and matching of video sequences," in *Proc. IEEE International Conference on Computer Vision*, Nice, France, Oct. 13-16, 2003, pp. 939-045.

[98] P. Tresadern and I. Reid, "Uncalibrated and unsynchronized human motion capture: a stereo factorization approach," in *Proc. IEEE Conf on Computer Vision and Pattern Recognition*, Washington, D. C., June 27 - July 2, 2004, pp.128-134.

[99] T. Tuytelaars and L. Van Gool, "Synchronizing video sequences," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Washington DC, USA, 2004, Vol. 1, pp. 762-768.

[100] L. Wolf and A. Zomet, "Correspondence-free synchronization and reconstruction in a non-rigid scene," in *Proc. Workshop on Vision and Modeling of Dynamic Scenes, Copenhagen*, Denmark, May 2002.

[101] Y. Caspi and M. Irani, "A step towards sequence-to-sequence alignment," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Hilton Head Island, SC, June 13-15, 2000, pp. 682-689.

[102] Y. Caspi and M. Irani, "Alignment of non-overlapping sequences," in *Proc. of the 8th Int. Conf. on Computer Vision*, Vancouver, B.C., July 2001, pages 76-83.

[103] I. Reid and A. Zisserman, "Goal-directed video metrology," in *Proc. of the 4th European Conference on Computer Vision*, vol. 2 LNCS 1065, Cambridge, April 1996, pp. 647-658.

[104] B. Triggs, "Factorization methods for projective structure and motion," in *Proc. IEEE Conference on Computer Vision and Pattern* Recognition, San Francisco, CA, USA, 1996, pp 845-851.

[105] P. Tresadern and I. Reid, "Synchronizing image sequences of non-rigid objects," in *Proc. British Machine Vision Conference*, Norwich, Sept. 9-11 2003, Vol. 2, pp. 629-638.

[106] J. Shi and C. Tomasi, "Good features to track," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, Washington, June 1994, pp. 593-600.

[107] N. Chiba and T. Kanade, "A Tracker for broken and closely spaced lines," in *Proc. of the Int. Soc. for Photogrammetry and Remote Sensing*, Stuttgart, Germany, 1998, Vol. XXXII, No. 5, pp. 676-683.

[108] The Persistence of Vision Raytracer, http://www.povray.org/

[109] ARToolkit, http://www.hitl.washington.edu/artoolkit/

[110] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. F. Tappen, C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6): 1068-1080 (2008).

[111] C. Chekuri, S. Khanna, J. Naor, and L. Zosin, "A linear programming formulation and approximation algorithms for the metric labeling problem," *SIAM J. Discrete Math*, 18(3): 608-625 (2004).

[112] N. Komodakis, G. Tziritas, "Approximate labeling via graph cuts based on linear programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1436-1453 (2007).

[113] M. Gong and Y.H. Yang, "Near real-time reliable stereo matching using programmable graphics hardware," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition 2005*.

[114] C. Kim, K.M. Lee, B.T. Choi, and S.U. Lee, "A dense stereo matching using two-pass dynamic programming with generalized ground control points," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[115] O. Veksler, "Stereo correspondence by dynamic programming on a tree," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[116] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(2):139–154, 1985.

[117] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in *Proc. of International Conf. on Computer Vision 2001*.

[118] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[119] J. Sun, H. Y. Shum, and N. N. Zheng, "Stereo matching using belief propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003.

[120] Q. Yáng, L. Wang, R. Yang, H. Stewénius, and D. Nistér, "Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition 2006*.

[121] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nistér, "Real-time global stereo matching using hierarchical belief propagation," in *Proc. of BMVC 2006*.

[122] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, 47(1/2/3):7-42, 2002.

[123] A. Blake and A. Zisserman, "Visual reconstruction," *MIT Press*, Cambridge, MA, 1987.

[124] J. Marroquin, S. Mitter, and T. Poggio, "Probabilistic solution of ill-posed problems in computational vision," *Journal of the American Statistical Association*, 82(397):76-89, 1987.

[125] S. D. Cochran and G. Medioni, "3D surface description from binocular stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):981-994, 1992.

[126] A. Agarwala,M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn,B. Curless, D. Salesin, and M. Cohen, "Interactive digital photomontage," in *Proc. of ACM Trans. Graphics*, vol. 23, no. 3, 2004, pp. 294-302.

[127] A. Klaus, M. Sormann and K. Karner, "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," in *Proc. of ICPR 2006*.

[128] M. Bleyer and M. Gelautz, "A layered stereo algorithm using image segmentation and global visibility constraints," in *Proc. of ICIP 2004*.

[129] Y. Boykov, O. Veksler and R. Zabih, "A variable window approach to early vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1283–1294.

[130] Z. Wang and Z. Zheng, "A region based stereo matching algorithm using cooperative optimization," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition 2008*.

[131] J. B. Xiao, M. Shah, "Motion layer extraction in the presence of occlusion using graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10): 1644-1659, 2005.

[132] C. L. Zitnick, S. B. Kang, "Stereo for image-based rendering using image over-segmentation," *International Journal of Computer Vision* 75(1): 49-65, 2007.

[133] Y. Taguchi, B. Wilburn, C. L. Zitnick, "Stereo reconstruction with mixed pixels using adaptive over-segmentation", in *Proc of IEEE Conf. on Computer Vision and Pattern Recognition 2008*.

[134] Y. Deng, X. Y Lin, "A fast line segment based dense stereo algorithm using tree dynamic programming," in *Proc. of European Conference on Computer Vision*, vol.3, 2006, pp. 201-212.

[135] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(5):603-619, May 2002.

[136] C. Christoudias, B. Georgescu, and P. Meer, "Synergism in low-level vision," 16*th International Conference on Pattern Recognition*, Quebec City, Canada, August 2002, vol. IV, pp.150-155.

[137] T. H. Cormen, C. E. Lerserson, Ponald L. Rivest, and C. Stein, "Introduction to algorithms," *The MIT Press*, 2nd Edition, 2001.

[138] C. Harris and M. Stephens, "A combined corner and edge detector," *in Proc. of the 4th Alvey Vision Conference*, 1988, pp. 147-151.

[139] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. of International Conf. of Computer Vision*, 1999, pp.1150-1157.

[140] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406, 1998.

[141] A. Bobick and S. Intille, "Large occlusion stereo," *Int. Journal of Computer Vision*, 33(3):1–20, Sept. 1999.

[142] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window, theory and experiment," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(9):920-932, Sept. 1994.

[143] M. Gong and Y.H. Yang, "Fast stereo matching using reliability-based dynamic programming and consistency constraints," in Proc. of *Int. Conf. on Computer Vision*, Nice, France, 13-16 October 2003.

[144] T. Hai, S. S. Harpreet, K. Rakesh, "A global matching framework for stereo computation," in *Proc. of International Conf. of Computer Vision*, 2001, pp. 532-539.

[145] S. Birchfield, C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," *International Journal of Computer Vision*, 35(3):269-293, 1999.

[146] M. Bleyerand M. Gelautz, "Graph-cut based stereo matching using image segmentation with symmetrical treatment of occlusions," *Signal Processing: Image Communication (Special Issue on Three-dimensional Video and Television)*, 22(2): 127-143, 2007.

[147] F. Dufaux and F. Moscheni, "Motion estimation techniques for digital TV: A review and new contribution," in *Proc. of the IEEE*, vol. 83, pp. 858-876, June 1995.

[148] R. Krishnamurthy, P. Moulin, and J. Woods, "Optical flow techniques applied to video coding," in *Proc. of International Conference on Image Processing*, pp. 570, 1995.

[149] C. Stiller and J. Konrad, "Estimating motion in image sequences: a tutorial on modeling and computation of 2D motion," *IEEE Signal Processing Magazine*, 16(4):70-91, 1999.

[150] J. Weickert, A. Bruhn, T. Brox and N. Papenberg, "A survey on variational optic flow methods for small displacements," *Mathematical Models for Registration and Applications to Medical Imaging (2006)*, pp. 103-113.

[151] J. Bigün, G. H. Granlund, J. Wiklund, "Multidimensional orientation estimation with applications to texture analysis and optical flow," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(8): 775-790, 1991.

[152] B. K. P. Horn, B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, 17(1-3):185-203, 1981.

[153] D. Kalivas and A. Swachuk, "A region matching motion estimation algorithm," *CVGIP: Image understanding*, 54(2):275-288, 1991.

[154] P. Anadan, "A computational framework and an algorithm for the measurement of visual motion," *International Journal of Computer Vision*, 2(3): 283-310, 1989.

[155] M. J. Black and P. Anandan, "The robust estimation of multiple motions: parametric and piecewise smooth flow fields," *Computer Vision and Image Understanding*, 63(1):75–104, January 1996.

[156] I. Cohen, "Nonlinear variational method for optical flow computation," in *Proc. Eighth Scandinavian Conference on Image Analysis*, vol.1, Norway, May 1993, pp. 523-530.

[157] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnorr, "Variational optic flow computation in real-time," *IEEE Transactions on Image Processing*, 14(5):608–615, May 2005.

[158] V. Lempitsky, S. Roth, and C. Rother, "FusionFlow: Discrete-continuous optimization for optical flow estimation," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition 2008*.

[159] M. J. Black, Allan D. Jepson, "Estimating optical flow in segmented images using variable-order parametric models with local deformations," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(10): 972-986, 1996.

[160] Y. A. Wang, E. H. Adelson, "Representing moving images with layers," *IEEE Transactions on Image Processing* 3(5): 625-638, 1994.

[161] M. L. Gong, Y. H. Yang, "Estimate large motions using the reliability-based motion estimation algorithm," *International Journal of Computer Vision* 68(3): 319-330, 2006.

[162] W. Trobin, T. Pock, D. Cremers, and H. Bischof, "Continuous energy minimization via repeated binary fusion," in *Proc. of European Conference on Computer Vision 2008*.

[163] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," in *Proc. of International Conf. on Computer Vision* 2007.

[164] B. Glocker, N. Paragios, N. Komodakis, "Optical flow estimation with uncertainties through dynamic MRFs," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition 2008*.

[165] C. L. Zitnick, N. Jojic, S. B. Kang, "Consistent segmentation for optical flow estimation," in *Proc. of International Conf. on Computer Vision 2005*, pp. 1308-1315.

[166] M. Bleyer, C. Rhemann, M. Gelautz, "Segmentation-based motion with occlusions using graph-cut optimization," *DAGM-Symposium 2006*, pp. 465-474.

[167] L. Xu, J. Chen, and J. Jia, "Segmentation based variational model for accurate optical flow estimation," in *Proc. of European Conference on Computer Vision 2008*.

[168] M. L. Gong, "Enforcing temporal consistency in real-time stereo estimation," in *Proc. of European Conference on Computer Vision 2006*, vol(3), pp. 564-577.

[169] Y. Zhang and C. Kambhamettu, "Integrated 3D scene flow and structure recovery from multiview image sequences," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2000.

[170] E. S. Larsen, P. Mordohai, M. Pollefeys, H. Fuchs, "Temporally consistent reconstruction from multiple video streams using enhanced belief propagation," in *Proc. of International Conf. on Computer Vision 2007*, pp. 1-8.

[171] H. Tao, H. S. Sawhney, R. Kumar, "Dynamic Depth Recovery from Multiple Synchronized Video Streams", in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 118-124, 2001.

[172] L. Xu and J. Jia, "Stereo matching: an outlier confidence approach," in *Proc. of European Conference on Computer Vision*, 2008.

[173] G. Zhang, J. Jia, T. T. Wong and H. Bao, "Recovering consistent video depth maps via bundle optimization," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition 2008*, Anchorage, Alaska, USA, June 2008.

[174] B. Goldlucke and M. A. Magnor, "Joint 3D-reconstruction and background separation in multiple views using graph cuts," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition 2003*, Madison, USA, June 2003, pp. 683-694.

[175] S. B. Lee and Y. S. Ho, "Multi-view depth map estimation enhancing temporal consistency," *ITC-CSCC 2008*.

[176] Y. Wei and L. Quan, "Region-based progressive stereo matching," in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition 2004*, vol (1), pp.106-113.

[177] S. B. Kang, R. Szeliski, J. X. Chai, "Handling occlusions in dense multi-view stereo," in *Proc. of CVPR*, 2001, vol (1), pp. 103-110.

[178] X. Chen and Y. H. Yang, "Segmentation-based background estimation using a set of related images," under submission.