Image based 3D Sensing and Modeling Technology for High-end Digital Cameras

A Dissertation Presented

by

Xue Tu

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

August 2009

Stony Brook University

The Graduate School

<u>Xue Tu</u>

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Dr. Murali Subbarao, Dissertation Advisor
Professor, Department of Electrical & Computer Engineering

Dr. Petar M. Djurić, Chairperson of Defense
Professor, Department of Electrical & Computer Engineering

Dr. Gene R. Gindi,
Associate Professor, Department of Electrical & Computer Engineering and
Department of Radiology

Dr. Xianfeng David Gu,
Assistant Professor, Department of Computer Science

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Image based 3D Sensing and Modeling Technology for High-end Digital Cameras

by

Xue Tu

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2009

Image-based 3D sensing techniques have become increasingly important due to advances in digital camera technology. These techniques can be used in numerous applications like auto-focusing, 3D modeling, and photorealistic rendering. In this dissertation, we first present Depth from Defocus (DFD) technology used in camera auto-focusing. This technique models image formation process as a convolution operation and recovers the depth of an object in a scene by comparing two different blurred images. We extend the technique for cameras operating in macro-mode by adopting a new lens setting, a new magnification normalization algorithm, and a three-image DFD scheme in a certain error sensitive area. We then present a novel Shape from Shift-variant Blurring technique to relax the shift-invariance assumption in the DFD technique. This technique localizes the global shift-variant transform in a small neighborhood which elegantly models the actual imaging process. The process can be inverted to recover not only depth of the object, but also its slope and curvature. Both simulation and experimental results are presented to illustrate its high potential. We also present a novel multi-stereo camera system for 3D modeling. This multi-camera system consists of three recently designed stereo cameras. Each camera is equipped with a stereo adaptor which splits the incoming light and projects random patterns onto the object in a scene. The camera is programmed to capture two images consecutively in a short time. The first image is with the random dots projected and the second is the normal texture image. Through the Stereopsis technique, each camera is able to reconstruct a partial shape of the target object. We propose a new calibration algorithm specially tailored for its unique stereo camera architecture that significantly improves the accuracy. Then a fast Gaussian-Pyramid based stereo matching technique is presented for partial shape recovery. Three stereo cameras are placed about 1.2 m away from the object and at 45 degrees apart. After stereo matching, a fast registration algorithm is carried out to bring three partial shapes into a common camera coordinate system. Then we adopt a volumetric based shape integration method to extract an iso-surface from layered point clouds and construct a complete triangle mesh using Marching-Cube algorithm. Finally, we combine three texture images to render a photorealistic model on a computer

monitor.

Contents

List of Figures

List of Tables

Acknowledgments

First of all, I would like to express my sincere and deep gratitude to my advisor, Professor Murali Subbarao. He offered me the great opportunity to do my research and full freedom to pursue my interest. Without his excellent guidance and constant encouragement, I can not finish this work.

I am also very grateful to Professor Petar M. Djurić, Professor Gene Gindi and Professor Xianfeng David Gu for serving on my thesis committee and reviewing this dissertation.

I would like to thank the former and current members of Computer Vision Lab: Dr. Soon-Yong Park, Dr. Tao Xian, Dr. Satyaki Dutta, Mr. Youn-Sik Kang, Mr. Shekhar Sastry, Mr. Karthik Sathyanarayana and Mr. Devadutta Ghat. It has been great pleasure to work with them. In particular, I want to thank Dr. Tao Xian for his valuable suggestions during the early stage of my research and Mr. Shekhar Sastry for writing camera-control software.

There are several other people I would like to thank for their support and encouragement. The staff of our department, Ms. Deborah Kloppenburg and Ms. Judy Eimer, have helped me to deal with numerous paperwork. I also want to thank Peng Dai, Mingyi Hong, Yao Li, Ting Lu, Wei Xu, Jiansong Chen, etc. for their precious friendships.

And special thanks goes to my parents for supporting me unconditionally in the past years. Finally, I want to thank my boyfriend, Weidong Cai, who is always there for me no matter what happens.

The support of this research in part by the Olympus Corporation is gratefully acknowledged.

Chapter 1

Introduction

With the development of high-quality and inexpensive digital cameras in recent decades, image-based sensing techniques have become increasingly important in many applications such as 3D modeling and autofocusing. In this dissertation, we present different image-based sensing techniques including Depth from Defocus (DFD), Shape from Shift-variant Blurring and Stereopsis. We also address issues related to 3D modeling including camera calibration, point cloud registration, shape integration and photo-realistic model rendering.

This chapter begins with describing a set of problems related to 3D sensing and modeling. We then briefly review some existing techniques and conclude with an overview of the dissertation.

1.1 Motivation

The core task of 3D sensing and modeling is to retrieve 3D surface information of an object and build a digital geometric model. Along with the development of new digitizing techniques, high power computing, and inexpensive hardware, 3D sensing technology has very wide applications. Autofocusing is one example that requires a camera to automatically determine the distance of an object and move the lens system to corresponding focus position. Reverse engineering is another application. It usually measures object surface through laser scanner or structured light system and then converts the obtained point cloud into polygonal models. In virtual reality, the 3D models are often further processed with texture mapping in order to be rendered photo-realistically on a computer screen.

In the research area of 3D sensing and modeling, the following steps are involved:

- Camera Calibration

  Camera calibration is the process to retrieve camera parameters such as focal length, optical center, aperture diameter, etc. It is the very first step of 3D sensing. In most applications, camera is considered as pin-hole model where only focal length, pixel size, optical center and camera position need to be calibrated. Some other applications (like autofocusing) adopt a more elaborate camera model where more parameters need to be retrieved.

- Depth-map Retrieval

  Depth-map represents the geometric 3D structure of the scene as a 2D image, where the value at specific pixel is the distance from the sensor to corresponding scene point. Techniques for depth-map retrieval are mainly divided into two classes: active techniques and passive techniques. Image-based sensing techniques belong to the latter.

- Multi-view Registration

The range image obtained from a single sensor provides only a partial surface of the object. To get a complete 3D structure, we need several range images from different view points. These range images must be brought into a common coordinate system so that geometric and photometric structures from multi-view range images could be fused into a single complete model. This process is called multi-view registration. It requires determining rigid body transformations between different coordinate systems.

- Multi-view Integration

  Multi-view integration is the technique to merge registered range images into a single model. Usually partial surfaces do not mesh with each other perfectly even after registration, therefore integration algorithm has to extract an iso-surface from layered partial surfaces.

- Texture Mapping

  After multi-view integration, we obtain only the geometric information of the 3D object which is usually represented as a set of polygons. We have to add surface reflectance and color to reconstruct a photo-realistic model. This is termed as texture mapping. It is akin to wrapping a gift paper onto a white box.

The ultimate goal of our research project is to build a real-time camera system to reconstruct a 3D photo-realistic model of the object from its 2D images. In this dissertation, we discuss the underlying theory, techniques, and our experiments.

1.2 Literature Review

In this section, we review some existing techniques in 3D sensing and modeling literature.

1.2.1 Camera Calibration

Camera calibration can be mainly divided into two categories: photogrammetric method and self calibration. Photogrammetric method determines camera parameters by observing a known geometry object (like checker board pattern) [1, 2], while self calibration can recover camera parameters up to a scale by exploring different constraints in arbitrary scene [3, 4]. 3D sensing tasks often require precisely measuring the surface structure, where photogrammetric methods are more suitable. Commonly adopted photogrammetric methods include DLT [5], Tsai [2] and Zhang's methods [1]. Direct Linear Method (DLT) first computes general projective matrix and then uses RQ factorization to retrieve intrinsic camera matrix and extrinsic transformation matrix. Tsai's calibration method assumes some parameters are provided by manufactures and use them as initial guess. It requires n feature points ($n >= 8$) per image and solves the problem with a set of n linear equations based on radial alignment constraint. Zhang's method requires a planar checkerboard grid to be placed at different orientations in front of the camera. It first provides a closed-form solution by exploring constraints from homography and then refine it through minimizing reprojected geometric errors. Both Tsai and Zhang's methods can incorporate lens distortion estimation in iterative refinement step, while DLT corrects the lens distortion separately and conducts calibration algorithm on "distortion free" images. It employs the property that perspective projective projection retains linearity to estimate distortion , i.e., a straight line in 3D space should remain as straight after imaging process if lens distortion effect dose not exist. Most methods use sum of line fitting residues as error measure and estimate distortion parameters through minimizing the residue [6], [7], [8]. Recently, some researchers propose to incorporate lens distortion into fundamental matrix computation [9] [10] and solve the distortion values based on 8-point algorithm [11].

1.2.2 Depth-map Retrieval

There are mainly two groups of techniques to acquire depth maps: active techniques and passive techniques.

Active techniques can be understood as a modulation and demodulation procedure. A carrier signal is projected onto the object surface, and the modulated reflected light is detected by the sensor. The measurement result is obtained from demodulation calculation. Laser range imaging and structured light techniques are the most common active techniques. Some recent investigation [12, 13, 14] on these techniques show that they can measure the depth very accurately in real time. However, active techniques are much more expensive than passive techniques.

In contrast, passive techniques work with only naturally formed images produced by reflected light from object. Some common methods include stereopsis, motion parallax, shape from shading, depth from focus and depth from defocus.

Stereopsis is a widely used technique for 3D shape reconstruction without active illumination. It is similar to human visual perception in that object depth is sensed from two slightly different projections on the retinas. If two cameras are fixed in parallel facing the object, the depth of certain object point can be recovered through triangulation from its corresponding points in two images. There are two main different classes of algorithms to perform stereo matching: local (window-based) algorithms and global algorithms. Local (window-based) algorithms computes the disparity at a given pixel by searching for the point with highest similarity within a finite window. Sum-of-Squared-Differences (SSD)[15, 16], normalized cross-correlation [17] are two popular criteria to measure similarity. Other criteria include binary matching costs (i.e. match/no match) [18, 19], rank transform [20] and gradient based measures [21]. On the other hand, global algorithms make explicit smoothness assumptions and seek a disparity (distance between corresponding points) assignment minimizing a global cost function. The main distinction between these algorithms is the minimization method used, e.g., simulated annealing [22], probabilistic (mean-field) diffusion [23] or graph cuts [24]. In between these two broad classes are certain iterative algorithms, which do not explicitly state a global function but whose behavior mimics closely that of iterative optimization algorithm [18, 23, 25]. Hierarchical (coarse-to-fine) algorithms are one example. They typically operate on an image pyramid, where results from coarser levels are used to constrain a more local search at finer levels [26, 27, 28].

Motion parallax is similar to stereopsis, which also utilizes disparity between corresponding points to recover depth, except that the disparity is from relative movement. Most stereopsis techniques can be all applied to motion-structure estimation. Unlike stereopsis, the movement can be either caused by the object or the camera, so there exists an ambiguity. In most cases, motion parallax can only reconstruct the scene geometry up to a a scale unless more information is provided

[5, 29].

Shape from shading (SFS) recovers object shape from a gradual variation of shading in the image. SFS technique adopts a Lambertian reflectance model, where the gray level at a pixel depends on the light source direction and the surface normal. Given a set of images captured with different light sources, SFS usually first recovers the surface normal (or gradient) and then the shape through integration [30]. There are mainly four classes of techniques in SFS literature: minimization approaches, propagation approaches, local approaches and linear approaches. Minimization approaches obtain the solution by minimizing an energy function [31, 32, 33]. Propagation approaches propagate the shape information from a set of surface points to the whole image [34, 35]. Local approaches derive shape based on the assumption of surface type [36, 37]. Linear approaches compute the solution based on the linearization of the reflectance map [38, 39].

Depth from focus and depth from defocus both retrieve depth map from a set of different blurred images captured by a single camera. Depth from focus (DFF) technique acquires many images with different lens settings, then searches for the best setting through a focus measure [40, 41, 42]. The ideal focus measure should be unimodal, monotonic, and reaches its maximum when the object is in focus. However, due to lens configuration, the residual error of focus motor and device noise, there exist many local maxima. A more complicated searching strategy that combines Fibonacci search and quadratic curve fitting has been demonstrated to find the global maximum [42, 43]. Normally, DFF algorithm is slow because of the multiple image acquisition.

Depth from Defocus requires only two or three images, and recovers the depth information by computing the degree of blur. This makes it suitable for realtime autofocusing and distance measurement. Early algorithms extracted depth information from blur measurement of an edge [44, 45]. Later methods for arbitrary objects can be classified as frequency domain approaches [46, 47, 43, 48, 49], spatial domain approaches [50, 51, 52, 53, 54, 55], and statistical approaches [56, 57].

The frequency domain based methods tend to recover the depth through inverse filtering in Fourier domain. Subbarao [46] removes the constraint of pinhole aperture and allows several camera parameters to be varied simultaneously. To avoid the sensitivity to the frequency spectra of local scene textures, Xiong and Shafer [43] use moment filters compensate the frequency spectra within passband of each narrowband filter. Watanabe and Nayar [48] propose a class of broadband filters instead of a series of narrowband filters. They claim that the filters are sufficient to achieve high-resolution depth map of image even with complex texture information.

The statistical approach formulates the image formation process as a convolution plus additive noise. The depth information coded in Point Spread Function (PSF) can be estimated through Maximum a Posterior (MAP) criteria [56]. In [57], Rajagopalan and Chaudhuri model the image as Markov Random Field to add some smooth constraints between pixels during estimation. Statistical approach usually requires higher computation power and yields lower depth-map density.

The spatial domain approach usually involves less computation as it preserves the spatial correspondence information. Subbarao and Surya propose a spatial-domain Convolution/Deconvolution Transform (S Transform) for n-dimensional signals in [58] and apply S Transform in depth recovery [51, 59]. They approximate blurred images as cubic polynomial in local neighborhood and estimate depth-related blur parameter by solving a quadratic equation. Two variations of this method have been developed where the quadratic equation is reduced to liner equation. Ziou et al. [55, 60, 61] fits the image with Hermite polynomial basis and illustrates that the Hermite coefficients of one blurred image are functions of partial derivatives of other images and blur difference. They calculate the blur difference through a group of equations and then recover the depth map. Favaro et al. [53] infer the 3D geometry from defocused images through computing PSF related orthogonal operators which are regularized through functional singular value decomposition.

Most methods in DFD literature assume the scene (or small surface patch) is planar and parallel to lens system (so called equifocal or shift-invariant assumption), which limits the accuracy of reconstructed depth-map. Recently, Subbarao et al. [62, 63, 64] propose a new framework to model the actual shift-variant blurring process. Based on the new theory, they develop a method to recover both depth and slopes of local planar surface patches. Favaro et al. [65] convert the shift-variant integral equation to diffusion equation and solve partial differential equation through gradient flow. Namboodiri et al. [66] also convert the problem to diffusion equation but estimate the parameters through MRF-graph cuts based methods.

1.2.3 Multi-view Registration

Depth map (or point cloud) from single sensor only reveals partial surface information. To build a full surface model, more depth maps obtained from different viewing directions are needed. Usually point clouds are constructed with respect to their own camera frames, therefore they need to be aligned into a common coordinate system. Transformation between different coordinate systems can be parameterized as a rotation and a translation. Registration is the procedure to identify the transformation between each camera coordinate system and the reference system.

Registration of multi-view range images is generally done in two steps: coarse registration and registration refinement [67]. Most registration algorithms assume the initial approximate transformation is provided and concentrate on improving it. With calibrated camera systems, the transformation parameters can be derived from calibration result. If the system is uncalibrated, auto calibration [3, 4], wide baseline stereo [68] or motion parallax [29] techniques can be used to estimate the relative motion parameters between different views. Starting from initial solution, registration refinement improves the result by iteratively minimizing certain error measure. Iterative Closest Point (ICP) approach [69, 70, 71] first pairs points in one depth map with nearby points in the other, then calculates the rigid motion that better aligns the two point clouds, and iterates these steps as long as the result improves. Exhaustive search for closest points can be used to identify matching pair, but it is very computationally expensive [69, 71]. Some approaches speed up the process by projecting the point along its normal vector [70, 72] or viewing direction [73, 74] line onto the other mesh and searching within the neighborhood of the intersection. There are mainly two approaches to align point clouds: aligning points with their mates (point-point alignment) [75, 76] and aligning points with the tangent planes of their mates (point-plane alignment) [70]. Besides using pure geometrical information, Johnson and Kang [77] improve registration by combining texture information.

1.2.4 Integration

After being aligned into a common coordinate system, multiple partial shapes have to be integrated into a single 3D model. There are two common categories of multi-view modeling techniques. The first one deals with multiple point clouds obtained from different sensors [78, 79, 80, 81]. The other approach is based on processing photographic images using volumetric modeling techniques, such as voxel coloring or shape from silhouettes [82, 83, 84]. In this dissertation, we adopt the technique belonging to the first category, which works with point clouds acquired by three stereo cameras placed at different positions.

Among the point clouds merging techniques, mesh-based integration and volumetric integration are two popular approaches. One of the mesh-based integration techniques is based on stitching neighborhood meshes from one view after segmenting meshes according to 'Region of Construction' algorithm [85]. Mesh 'zippering' and mesh 'growing' are other techniques investigated by Turk et al. [86] and Rutishauser et al. [87]. Volumetric integration usually adopts an implicit representation of the object's surface [88, 79, 80, 89, 90]. Hoppe et al. [88] produce a surface mesh from a group of unorganized points assuming that every point is on object surface, which can hardly be true in practice. Curless and Levoy [79] extract the iso-surface of object based on signed distances of voxel grid. Hilton et al. [80] employ a similar procedure but assume the implicit surface function be continuous. Most of the approaches assume a noise-free data set as the range images are acquired by high-accuracy laser scanners. An approach to handle the noise problem in partial shapes is investigated by Wheeler et al. [81]. They exclude the noise data using a 'consensus surface' algorithm. For point clouds acquired by stereo camera system, stereo matching algorithms can also be combined to create 3D models [91, 92].

1.2.5 Texture Mapping

Texture mapping is the last step in 3D reconstruction. The reconstructed mesh model consists of numerous triangles on its surface. To reconstruct a photo-realistic model, we need to map the texture information onto 3D wire-frame model. The process is akin to wrapping a patterned paper onto a plain white box.

In image-based rendering application, based on several acquired texture images and scene geometry, a new synthetic image is generated according to the virtual camera position for texture mapping [93, 94]. This is termed as view-dependent texture mapping. In 3D modeling applications, view-independent texture mapping techniques are usually employed, where a big texture atlas is produced and wrapped onto the whole model [95, 96]. With OpenGL or DirectX library, such techniques can be easily implemented by establishing the mapping between vertices of triangle mesh and their corresponding pixels in texture image. The system would automatically perform perspective correction, stretching and squeezing through interpolation and filtering.

When stitching multiple texture images together, there usually present obvious seams and color difference the the 3D model. Bannai et al. propose to use several global color transform functions to normalize the overall color tone [97]. Baumberg blends the texture images in frequency domain with camera weight function [98]. Rocchini et al. blend the texture of frontier triangles using barycentric coordinate factors [99]. Zhou et al. combine several techniques including gradient filed fusion and texture impainting to generate a seamless textured 3D model [100].

1.3 Dissertation Overview

This dissertation is organized as follows:

In Chapter 2, we first briefly review the spatial-domain *depth from defocus* (DFD) algorithm [51] which can recover depth from only two different blurred images. We then present new algorithms to extend the DFD method to cameras working in macro mode used for very close objects in a distance range of 5 cm to 20 cm. We adopt a new lens position setting suitable for macro mode to avoid serious blurring. We also develop a magnification normalization algorithm to normalize images' viewing fields. In some range intervals with high error sensitivity, we use an additional image to reduce the error caused by drastic change of lens positions. Experimental results on a high-end digital camera show that the new algorithms significantly improve the accuracy of DFD in the macro mode. In terms of focusing accuracy, the RMS error is about 15 lens steps out of 1500 steps, which is around 1%.

In Chapter 3, we propose a new approach for 3D shape recovery of local planar surface patches from two shift-variant blurred images. It is based on a recently proposed technique *Rao Transform/Inverse Rao Transform*. The shape of a visible object at each pixel is specified by the distance and slopes of a local planar surface patch. Our method elegantly models the actual imaging process and recovers the shape by inverting the shift-variant blurring in camera system. It is completely localized, applicable to general Point Spread Function and facilitates fine-grain parallel implementation. Results of both simulation and real experiments indicate that the new approach is effective and useful in practical applications.

In Chapter 4, we present new camera calibration techniques for a novel designed stereo camera. The stereo camera features a digital camera and a stereo adaptor attached in front of the lens. With a pair of periscope-like mirror system, the adaptor splits the incoming light and form a pair of stereo image on left and right half sensor respectively. Straightforward extension of the traditional calibration techniques are found to be inaccurate and ineffective. Our new technique fully exploits the physical constraint that the stereo image pair have the same intrinsic camera parameters such as focal length, principle point and pixel size. Our method involves taking one image of a calibration object and estimating one set of intrinsic parameters and two sets of extrinsic parameters corresponding to the mirror adaptor simultaneously. The method also includes lens distortion correction to improve the calibration accuracy. Experimental results on a real camera system are presented to demonstrate that the new calibration technique is accurate and robust.

In Chapter 5, we first review the basic theory of stereopsis and introduce our novel stereo camera system. We then present several fast implementation methods to meet the requirements of real time processing. We combine lens distortion correction and rectification in one single step. We also adopt a multi-resolution search scheme to identify corresponding points pair using Gaussian pyramid structure. The search window size is limited based on the system design to further improve the speed. A simple object segmentation is employed to exclude erroneous results of background pixels. The experiments carried out on our stereo camera system illustrates that our fast stereo matching algorithm is reliable.

In Chapter 6, we introduce our multi-stereo camera system where three stereo cameras are placed at different positions to capture images. Each camera is able to reconstruct partial shape of the object with respect to its own coordinate system. Registration is the process to retrieve the transformation matrices between different camera frames and bring partial shapes into a common coordinate system. The coarse registration can be done using information derived from calibration result. We then present a fast Iterative Closest Surface Patch (ICSP) approach to refine the registration. It finds the closest surface patches in overlapping parts of two point clouds and compute the transformation matrices that minimize the error distance. The process is carried out iteratively until the error measure converges. The experimental results on real system data illustrate that our registration method is fast and accurate.

In Chapter 7, we first describe a volumetric method to extract the object surface from layered point clouds. It calculates the signed distances from every voxel to different range surfaces and averages them according to viewing angles. The true object surface rises at voxels whose signed distances are zero. The Marching Cube algorithm is employed to generate a triangle mesh that intersect voxel grid at zero distance. With preset lookup table, such process can be done in real time. Finally, we present the texture mapping techniques to render a photo-realistic 3D model. Among three texture images, the one with best viewing direction is selected as map source. Due to inherent manufacturing differences and viewing directions, images captured by different cameras would present serious blocking artifacts when they are stitched together. To remove the artifacts, global color normalization followed by local edge smoothing is performed. The final model appearance is satisfactory.

Finally, conclusion and future work are discussed in chapter 8.

Chapter 2

Depth from Defocus

One important passive technique for acquiring range images is *depth from defocus* (DFD) [51]. It requires the camera to capture two or three images with different lens settings and recover the depth of an object by computing the degree of blurring. There are two main categories of DFD algorithms: statistical and deterministic. Statistical approaches like Maximum likelihood[57] and Markov Random field methods[56] are computationally intensive. They formulate the task as a global estimation problem and solve it through iterative techniques. Deterministic algorithms can be classified as frequency domain approaches[43, 48] and spatial domain approaches[51, 55, 54]. The frequency domain approaches transform images to frequency domain and recover the depth through different filtering techniques. They usually yield lower depth-map resolution. In comparison, spatial domain approaches use only a small image region and therefore require less computation and generate a denser depth-map. Due to the inherent advantage of being local in nature, spatial domain approaches are more suitable for real-time autofocusing applications.

A Spatial-domain Convolution/Deconvolution Transform (S Transform)[58] was proposed for n-dimensional signals and arbitrary order polynomials. Surya and Subbarao[51] utilized S Transform to estimate the blur parameter in the spatial domain using a method named STM. There are two basic variations of STM: STM1 uses two images captured by changing the lens position, and STM2 is based on varying the aperture diameter. In practice, STM1 has been found to have good accuracy for cameras operating in the normal mode used for objects at a distance of 20 cm or more[52]. When cameras operate under macro-mode used for objects at close range between 5 cm to 20 cm, many difficulties arise, such as highly blurred images, significant magnification change for small lens displacement, and specific settings in lens parameters.

In this Chapter, we first review the basics of S Transform and STM-DFD, then we present some new techniques to extend STM1 to cameras working in macro-mode. We adopt a new lens setting suitable for macro-mode to avoid serious blurring and develop a new calibration algorithm to normalize the magnification of images captured with different lens positions. In some range intervals with high error sensitivity, we use an additional image to reduce the error caused by drastic change of lens settings. Experimental results on different test objects show that the new algorithm significantly improves the accuracy of STM1 for macro mode. The RMS error is about 15 lens steps out of 1500 steps, which is around 1%.

2.1 Camera Model

In the defocus problem, we adopt a more elaborate camera model rather than the common pinhole camera model. Figure 2.1 shows a schematic diagram of a camera system, where L1 and L2 are two lenses, OA is the optical axis, P1 and P2 are the principal planes, Q1 and Q2 are the principal points, ID is the image detector, D is the aperture diameter, s is the distance between the second principal plane and the image detector, u is the distance of the object from the first principal plane, and v is the distance of the focused image from the second principal plane. The relation

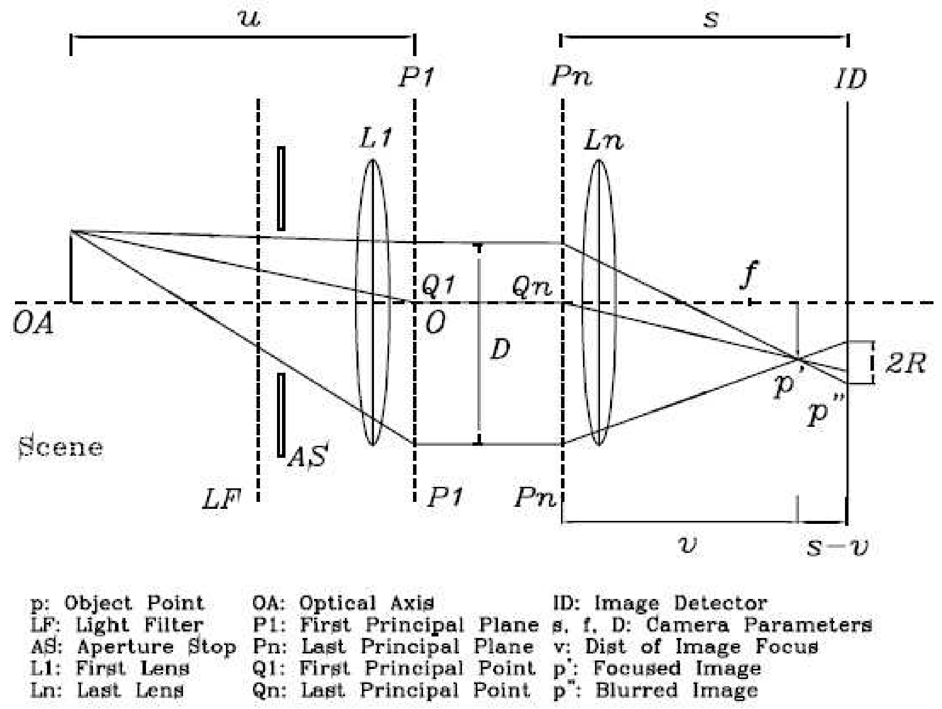

Figure 2.1: Schematic diagram of a camera system

between f, u and v is expressed by the well-known lens formula

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v} \tag{2.1}$$

The distance s, focal length f and aperture diameter D will be referred together as camera parameters and denoted by $\mathbf{e}$, i.e.,

$$\mathbf{e} = (s, f, D). \tag{2.2}$$

If the object point p is not in focus, then it gives rise to a blurred image p'' on the image detector ID. According to geometric optics, p'' has the same shape as the lens aperture but scaled by a factor. If we assume the lens system to be a circular system, p'' is then a circle with uniform brightness. Its distribution can be expressed as a cylindrical function:

$$h(x,y) = \begin{cases} \frac{1}{\pi R^2} & \text{if } x^2 + y^2 \leq R^2 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

In practice, usually blurred circle is not a crisp circular patch of constant brightness as suggested by geometric optics. It is more like a circular blob with brightness falling off gradually at the border rather than sharply. Another popular PSF model is 2D Gaussian function:

$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.4}$$

where σ is a spread parameter corresponding to the standard deviation of Gaussian distribution.

If the blur circle radius R is constant over some region on the image plane, the camera can then be modeled as a linear shift invariant system. Therefore, the observed image $g(x, y)$ is the convolution of focused image $f(x, y)$ with Point Spread Function $h(x, y)$, i.e.,

$$g(x, y) = f(x, y) * h(x, y) \tag{2.5}$$

where * denotes the convolution operation.

2.2 S Transform

A new Spatial-Domain Convolution/Deconvolution Transform (S Transform) was proposed for images and n-dimensional signals [58] for the case of arbitrary order polynomials.

Let $f(x, y)$ be an image. Within a small region, we approximate it as a two variable cubic polynomial:

$$f(x, y) = \sum_{m=0}^{3} \sum_{n=0}^{3-m} a_{m,n} x^m y^n \tag{2.6}$$

where $a_{m,n}$ are the polynomial coefficients. This assumption of f can be made valid by applying a polynomial fitting least square smoothing filter to the image.

Let $h(x, y)$ be a rotationally symmetric point spread function. The moments of the point spread function are defined by

$$h_{m,n} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^m y^n h(x, y)\, dx\, dy \tag{2.7}$$

The observed image $g(x, y)$ is the convolution of the focused image $f(x, y)$ and the PSF $h(x, y)$:

$$\begin{aligned} g(x,y) &= f(x,y) * h(x,y) \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-\zeta, y-\eta)h(\zeta,\eta)\,d\zeta\,d\eta \end{aligned} \tag{2.8}$$

Since $f(x, y)$ is a cubic polynomial, it can be expressed in a Taylor series as

$$f(x-\zeta, y-\eta) = \sum_{0\le m+n\le 3} \frac{-\zeta^m}{m!}\frac{-\eta^n}{n!} f^{m,n}(x,y) \tag{2.9}$$

where

$$f^{m,n}(x,y) = \frac{\partial^m}{\partial x^m}\frac{\partial^n}{\partial y^n} f(x,y) \tag{2.10}$$

Substituting Eq.(2.9) into Eq.(2.8), we have

$$\begin{aligned} g(x,y) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \sum_{0\le m+n\le 3} \frac{(-1)^{m+n}}{m!n!} f^{m,n}(x,y)\zeta^m\eta^n h(\zeta,\eta)\,d\zeta\,d\eta \\ &= \sum_{0\le m+n\le 3} \frac{(-1)^{m+n}}{m!n!} f^{m,n}(x,y)\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \zeta^m\eta^n h(\zeta,\eta)\,d\zeta\,d\eta \\ &= \sum_{0\le m+n\le 3} \frac{(-1)^{m+n}}{m!n!} f^{m,n}(x,y)h_{m,n} \end{aligned} \tag{2.11}$$

where

$$h_{m,n} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x,y)x^m y^n\,dx\,dy \tag{2.12}$$

Equation 2.11 expresses the convolution as a summation involving the derivatives of $f(x, y)$ and moments of $h(x, y)$. This corresponds to the forward S-Transform. Now we can use this formula to derive a deconvolution formula. Since $h(x, y)$ is circularly symmetric, it can be shown that

$$h_{0,1} = h_{1,0} = h_{1,1} = h_{0,3} = h_{3,0} = h_{2,1} = h_{1,2} = 0 \tag{2.13}$$

$$h_{2,0} = h_{0,2} \tag{2.14}$$

Also if we assume the camera to be a lossless system (i.e., no light energy is absorbed by the camera system), then

$$h_{0,0} = \iint h(x, y)\, dx\, dy = 1 \tag{2.15}$$

Therefore we obtain

$$f(x, y) = g(x, y) - \frac{h_{2,0}}{2}(f^{2,0}(x, y) + f^{0,2}(x, y)) \tag{2.16}$$

Applying $\frac{\partial^2}{\partial x^2}$ to the above equation on both sides and noting that derivatives of order higher than 3 are zero (because f is cubic), we obtain

$$f^{2,0}(x, y) = g^{2,0}(x, y) \tag{2.17}$$

Similarly applying $\frac{\partial^2}{\partial y^2}$, we get

$$f^{0,2}(x, y) = g^{0,2}(x, y) \tag{2.18}$$

Therefore

$$f(x, y) = g(x, y) - \frac{h_{2,0}}{2}\nabla^2 g(x, y) \tag{2.19}$$

where ∇^2 is the Laplacian operator, and σ_h is the spread parameter of h, by the definition of moments of h, we have $h_{2,0} = h_{0,2} = \sigma_h^2/2$, so the above deconvolution equation can be written as:

$$f(x,y) = g(x,y) - \frac{\sigma_h^2}{4}\nabla^2 g(x,y) \tag{2.20}$$

Equation (2.20) is Inverse S-Transform. In the following section, we will describe the application of this formula to distance estimation.

2.3 STM Autofocusing

Let $f(x,y)$ be the focused image of a planar object at distance u, $g_1(x,y)$ and $g_2(x,y)$ be two images of the object recorded for two different camera parameter settings $\mathbf{e}_1$ and $\mathbf{e}_2$ where

$$\mathbf{e}_1 = (s_1, f_1, D_1)$$
$$\mathbf{e}_2 = (s_2, f_2, D_2)$$

Left R be the radius of the blur circle and q be the scale factor defined as $q = 2R/D$. Using the property of similar triangles in Figure 2.1, we have

$$q = \frac{2R}{D} = \frac{s-v}{v} = s\left(\frac{1}{v} - \frac{1}{s}\right) \tag{2.21}$$

Substituting for $1/v$ from Eq. (2.1), we obtain

$$R = q\frac{D}{2} = s\frac{D}{2}\left(\frac{1}{f} - \frac{1}{u} - \frac{1}{s}\right) \tag{2.22}$$

In a practical camera system, if two images $g_i(x, y)$ for $i = 1, 2$ are captured with camera parameter settings of $\mathbf{e}_i$, then image magnification and mean image brightness may change even though nothing has changed in the scene. To compare the blurring in two images g_1 and g_2 in a correct and consistent manner, both image brightness and magnification have to be normalized. In the following discussion, we will assume that images have been normalized. Without loss of generality, we assume that the brightness normalization has been carried out by dividing the gray-scale value at every pixel by the mean brightness value, and magnification has been normalized corresponding to $s = s_0$. After magnification normalization, the normalized radius R' can be expressed as a function of camera parameters $\mathbf{e}$ and object distance u

$$R'(\mathbf{e}; u) = \frac{s_0 R}{s} = \frac{D s_0}{2}\left(\frac{1}{f} - \frac{1}{u} - \frac{1}{s}\right) \tag{2.23}$$

In practice, it is found that [101] the blur spread parameter σ is proportional to R', i.e.

$$\sigma = kR' \quad \text{for} \quad k > 0 \tag{2.24}$$

In most cases, $k = 1/\sqrt{2}$ is a good approximation [101, 45, 102].

The actual PSF of camera system is much more complicated than Cylindrical or Gaussian due to diffraction, polychromatic illumination and lens aberrations, etc. Yet the spread parameter σ can be used to characterize the extent of blurring, where

σ is the standard deviation of the PSF $h(x, y)$, i.e., the second central moment of $h(x, y)$:

$$\sigma^2 = \iint \left((x - \bar{x})^2 + (y - \bar{y})^2\right) h(x, y)\, dx\, dy \tag{2.25}$$

where $(\bar{x}, \bar{y})$ is the position of the centroid of PSF $h(x, y)$.

For a planar object perpendicular to the optical axis, the camera acts as a linear shift invariant system. Therefore g_i will be the convolution of focused image $f(x, y)$ and the corresponding point spread function $h_i(x, y)$. The spread parameter σ is linear to reciprocal of object distance u^{-1} (see Eq.3.25 and Eq.2.24) [102]. Let the spread parameter σ for h_1 be σ_1 and for h_2 be σ_2. From the polar coordinate system, and equation (3.25), we obtain:

$$\sigma_1 = m_1 u^{-1} + c_1 \tag{2.26}$$

$$\sigma_2 = m_2 u^{-1} + c_2 \tag{2.27}$$

where

$$\begin{aligned} m_1 &= -\frac{D_1 s_0}{2\sqrt{2}} && (2.28)\\ c_1 &= \frac{D_1 s_0}{2\sqrt{2}} \left(\frac{1}{f_1} - \frac{1}{s_1}\right) && (2.29)\\ m_2 &= -\frac{D_2 s_0}{2\sqrt{2}} && (2.30)\\ c_2 &= \frac{D_2 s_0}{2\sqrt{2}} \left(\frac{1}{f_2} - \frac{1}{s_2}\right) && (2.31) \end{aligned}$$

Then σ_1 can then be expressed in terms of σ_2 as

$$\sigma_1 = \alpha\sigma_2 + \beta \tag{2.32}$$

where

$$\alpha = \frac{m_1}{m_2} \qquad \text{and} \qquad \beta = c_1 - c_2\frac{m_1}{m_2} \tag{2.33}$$

We assume that in a small image neighborhood the focused image $f(x, y)$ can be adequately approximated by a cubic polynomial. Now we can apply the inverse S Transform (Eq. (2.20)) on both observed images g_1 and g_2:

$$f = g_1 - \frac{1}{4}\sigma_1^2\nabla^2 g_1 \tag{2.34}$$

$$f = g_2 - \frac{1}{4}\sigma_2^2\nabla^2 g_2 \tag{2.35}$$

Equating the right sides of both equations, we obtain

$$g_1 - \frac{1}{4}\sigma_1^2\nabla^2 g_1 = g_2 - \frac{1}{4}\sigma_2^2\nabla^2 g_2 \tag{2.36}$$

If we take partial derivatives on both sides, together with the fact that f is cubic polynomial, it can be easily verified that $\nabla^2 g_1 = \nabla^2 g_2$. Therefore, the above equations can be expressed as

$$g_1 - g_2 = \frac{1}{4}G\nabla^2 g \tag{2.37}$$

where

$$G = \sigma_1^2 - \sigma_2^2 \tag{2.38}$$

and

$$\nabla^2 g = \nabla^2 g_1 = \nabla^2 g_2 = \frac{\nabla^2 g_1 + \nabla^2 g_2}{2} \tag{2.39}$$

Substituting for σ_1 in terms of σ_2 from Equation (2.32) and using the definition of G in Equation (2.38), we obtain

$$\sigma_2^2(\alpha^2 - 1) + 2\alpha\beta\sigma_2 + \beta^2 = G \tag{2.40}$$

where α and β are defined in Equation (2.33).

In STM1, the aperture diameter is not changed but the lens position is changed during capture of two images g_1 and g_2, (i.e., $f_1 \neq f_2$ and $s_1 \neq s_2$, but $D_1 = D_2$). Therefore

$$\alpha = \frac{m_1}{m_2} = \frac{D_1}{D_2} = 1 \tag{2.41}$$

so the quadratic equation is reduced to a linear equation which can be solved directly:

$$\sigma_2 = \frac{G}{2\beta} - \frac{\beta}{2} \tag{2.42}$$

and object distance can be retrieved using Eq. (2.27).

2.4 Extension to Macromode

In practical implementation of STM autofocusing, especially when camera is operating in macromode, a many changes are needed to make the method applicable.

2.4.1 SNR Mask

Ideally it should be possible to compute the value of σ_2 at each pixel (x, y) and obtain an estimate of the specific distance. However, because of digitization and noise, it is necessary to combine information from its neighboring pixels. From Equation (2.38), we notice that Laplacian of observed images $\nabla^2 g$ plays an important role in σ_2 computation. Unfortunately, Laplacian estimates are very much sensitive to camera noise and quantization. Its low Signal-to-Noise Ratio (SNR) will lead to large errors in estimation of σ_2. Therefore, we introduce a binary mask [52] to improve the robustness of STM1 algorithm. The binary mask is defined by thresholding the Laplacian values:

$$M_0(x, y) = \begin{cases} 1 & |\nabla^2 g| \geq T, \\ 0 & \text{otherwise.} \end{cases} \tag{2.43}$$

where T is a threshold on the Laplacian. It can be determined experimentally. We then compute the average G within a small neighborhood W:

$$G = \frac{4}{N} \sum_{(x,y)\in W}\sum M_0(x, y) \frac{g_1(x, y) - g_2(x, y)}{\nabla^2 g(x, y)} \tag{2.44}$$

where

$$N = \sum_{(x,y)\in W}\sum M_0(x, y) \tag{2.45}$$

Upon getting average G, the average σ_2 and average inverse distance $1/u$ for the small image patch can be calculated. This operation will remove unreliable points with low SNR and optimize the estimation of depth, but at the cost of reducing spatial resolution.

2.4.2 Magnification Normalization

As stated earlier, the observed images g_1 and g_2 should be normalized in terms of both brightness and magnification. In practical implementation, as magnification change is less than 3% in most cases for consumer digital cameras, it is usually ignored in favor of computational speed. However, when a camera is working in macro-mode (for objects closer than 20 cm) instead of the standard mode (for objects farther than 20 cm), such change (shown in Fig. 2.2) has to be taken into account as the shift between corresponding points could be up to 20 pixels near the image borders. To ensure the correct calculation of the blur parameter σ_2 at each pair of corresponding pixels, image magnification normalization becomes a necessity.

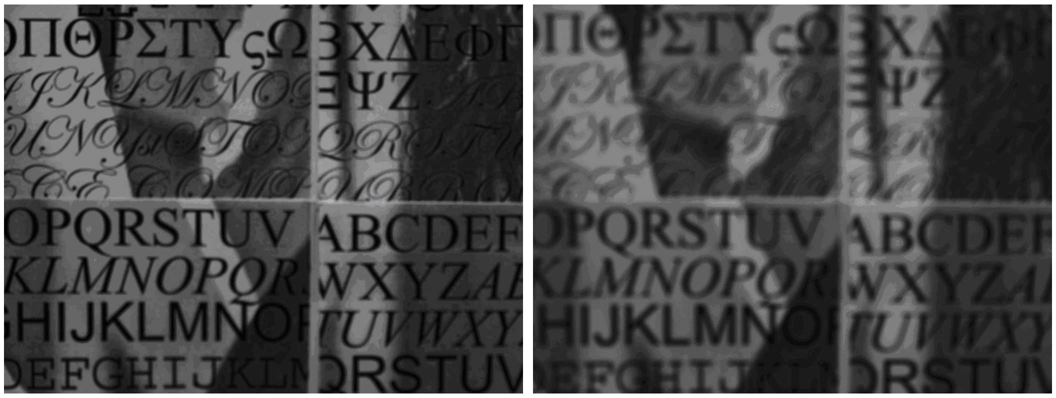

Figure 2.2: Images taken with different camera settings in macro mode

The relation between two images' magnifications can be expressed as

$$\begin{bmatrix} x^{(2)} \\ y^{(2)} \\ z^{(2)} \end{bmatrix} = \begin{bmatrix} s & 0 & s_x \\ 0 & s & s_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^{(1)} \\ y^{(1)} \\ z^{(1)} \end{bmatrix} \tag{2.46}$$

where $[x^{(1)}, y^{(1)}, z^{(1)}]^t$ and $[x^{(2)}, y^{(2)}, z^{(2)}]^t$ are homogeneous coordinates of corresponding pixels in observed images g_1 and g_2.

To compute the transformation matrix between magnification of two images, we need to calculate three coefficients s, s_x, s_y from calibration. One method is to take pictures of a simple pattern object (like black dots on white paper), and get two sets of corresponding pixel coordinates from pattern images g_1 and g_2. Then we can form a group of over constrained equations as shown below, which can be solved through least square criterion.

$$\begin{bmatrix} x_1^{(1)} & 1 & 0 \\ y_1^{(1)} & 0 & 1 \\ x_2^{(1)} & 1 & 0 \\ y_2^{(1)} & 0 & 1 \\ x_3^{(1)} & 1 & 0 \\ y_3^{(1)} & 0 & 1 \\ & \vdots & \\ x_n^{(1)} & 1 & 0 \\ y_n^{(1)} & 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ s_x \\ s_y \end{bmatrix} = \begin{bmatrix} x_1^{(2)} \\ y_1^{(2)} \\ x_2^{(2)} \\ y_2^{(2)} \\ x_3^{(2)} \\ y_3^{(2)} \\ \vdots \\ x_n^{(2)} \\ y_n^{(2)} \end{bmatrix} \tag{2.47}$$

If we write it as $Ax = b$, then x could be solved through pseudo inverse method:

$$x = (A^t A)^{-1} A^t b \tag{2.48}$$

After getting the transformation matrix, the next step is to warp the target image (e.g., g_1, usually the one with larger field of view). As forward warping usually maps integer pixel coordinates to sub pixel coordinates, here a backward warping scheme is applied. Integer pixel coordinates in transformed image are projected back to the original image, usually resulting in sub pixel position. Then its gray scale value will be computed by linear interpolation. The results of magnification correction is shown in Figure 2.3. we can clearly notice that the viewing field are about the same in both images.

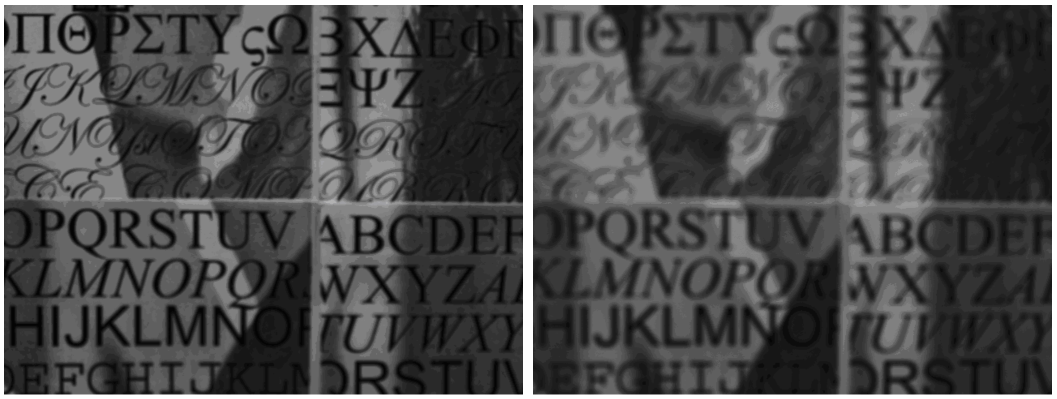

Figure 2.3: Images with Normalized Magnification

2.4.3 New Lens Setting

Besides magnification change, another problem in macro-mode DFD is that images are extremely blurred if the same lens positions are adopted as that for the standard mode. In this case, the estimated Laplacian of observed image $\nabla^2 g$ is noisy and unreliable (SNR will be low as Laplacian magnitude is reduced due to high levels of blur). We have to change the camera settings so that the images used in estimating σ have a moderate or low blur.

Another problem is that the relation between σ and distance u is not exactly linear (Eq. (2.27)) due to change of multiple unknown parameters in the lens system (e.g. lens position and focal length). Therefore, we calibrate the camera to obtain a new lookup table specifically for macro-mode autofocusing. The lookup table has two entries: step number and blur parameter σ. Step number is a measure of lens position. It is approximately linear with reciprocal of object distance $1/u$. We use Depth-from-Focus (DFF) [42] method to obtain the step number corresponding to the object distance that is in best focus. DFF acquires a number of images and searches for the best camera parameter that maximizes a focus measure. It needs much more computation than DFD but is usually more accurate. The results are stored in a table indexed by step number that gives focused object distance.

We place objects at several specified distances and compute the corresponding blur parameter σ through Eq.(2.42) and the focusing step number by DFF method. These results were stored in a table indexed by σ. After establishing this lookup table, we use linear interpolation to get the focusing step number for any blur parameter σ. The lens step number is then indexed into another table to obtain the depth.

Due to some special change of lens position when camera operates in macro-mode, mapping from σ to step number is not linear or even monotonic in a particular sensitive region (shown in Figure 2.4). This makes the looking up task ambiguous and difficult. In such a case, we capture an additional image with a new camera setting and use a new lookup table to get the step number. The new lookup table will be such that its sensitive region differs from the previous one. This strategy of using an additional image and two lookup tables will effectively improve the accuracy as the sensitive region of both lookup tables are avoided. Results of this 3 image DFD approach is illustrated in Section 2.5.

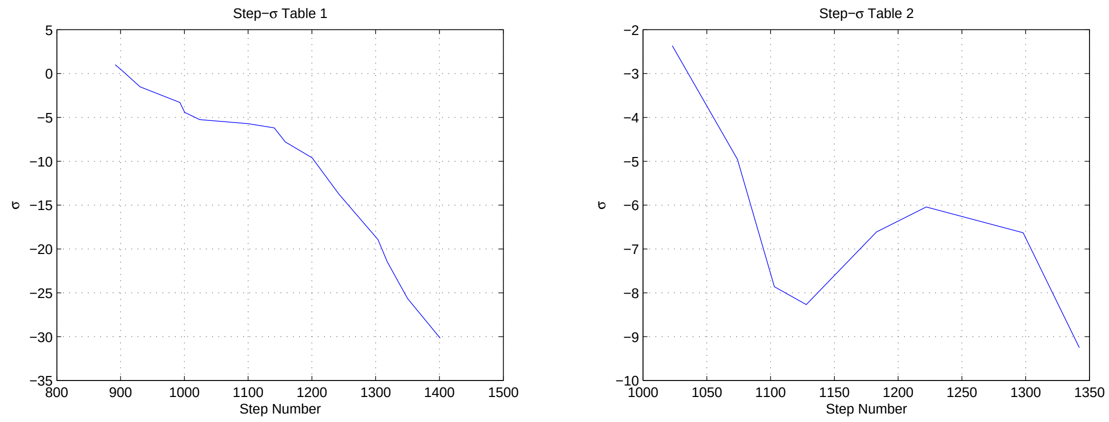

Figure 2.4: Step-σ Table for Different Camera Settings

2.5 Experimental Results

We conducted our experiments on Olympus E-1 digital camera. The camera is controlled by a host computer (Pentium 4 2.4GHz) from a USB port. The lens' focus motor position range from step 0 to step 1500, where step 0 corresponds to focusing at infinity and step 1500 corresponds to focusing at a nearby object.

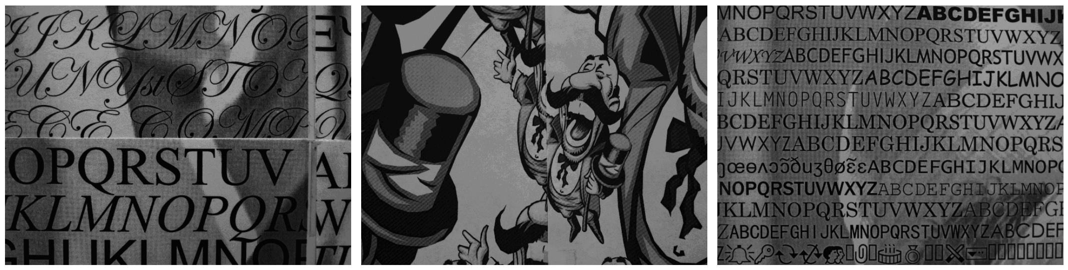

Figure 2.5: Test Objects

The performance of modified DFD algorithms for macro-mode was evaluated with experiments using different objects shown in Figure 2.5. Each object is placed at eight different distances in the range of 5 cm to 20 cm at roughly 2 cm intervals. The distance and the corresponding focused step numbers are listed in Table 2.1. The steps are obtained using the DFF algorithm. Two images are taken with camera set to step 800 and step 1100. If the focus step calculated by DFD is between 1000 and 1150, then a third image will be taken with the camera set to step 1250. In this case a different lookup table is used to retrieve the depth. The focusing window is set to 96×96 located at the center of the scene. Before performing DFD, all the images are normalized with respect to magnification and brightness and smoothed by a Gaussian filter. The image Laplacians are thresholded to weed out low contrast pixels with low SNR.

Position	1	2	3	4	5	6	7	8	9
Distance(mm)	250	225	200	180	150	120	100	80	55
Step	786	806	893	902	993	1099	1158	1217	1401

Table 2.1: Distance-Step Table

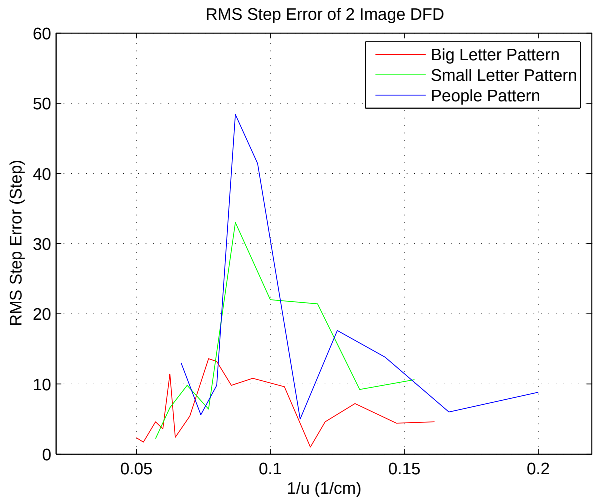

Figure 2.6: RMS Error of 2 Images DFD

First we present the RMS errors for 2 Image DFD and 3 Image DFD for objects at different distances. All test objects are flat and parallel to the lens plane. Therefore, computing depth from central focus window is sufficient. For 2 Image DFD, RMS errors are quite small for most regions, except in the range u = 10 cm to 12.5 cm, or focus step position around 1100. This is the sensitive region shown in Figure 2.4. If we combine the result from the third image, we find that the results are significantly improved. Most RMS errors are less than 15 steps out of 1500 steps. The maximum error is just 22 steps and the average error is about 1%.

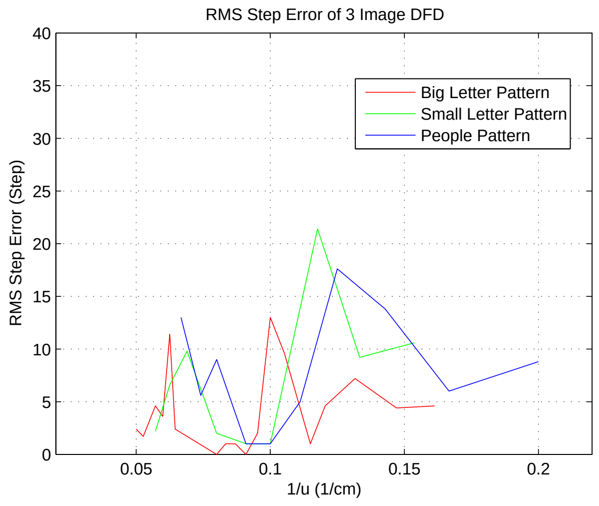

Figure 2.7: RMS Error of 3 Images DFD

Next we present the shape reconstruction results for planar objects. We split the image into smaller blocks and compute depth separately in each block. The magnification normalization procedure plays an important role here as magnification can change up to 20 pixels near image borders.

Figure 2.10 displays the estimated shape of object in Figure 2.2. With magnification normalization, the results around image borders are improved significantly.

The next example is to an inclined poster shown in Figure 2.9. Once again, magnification normalization is found to be very effective in improving accuracy.

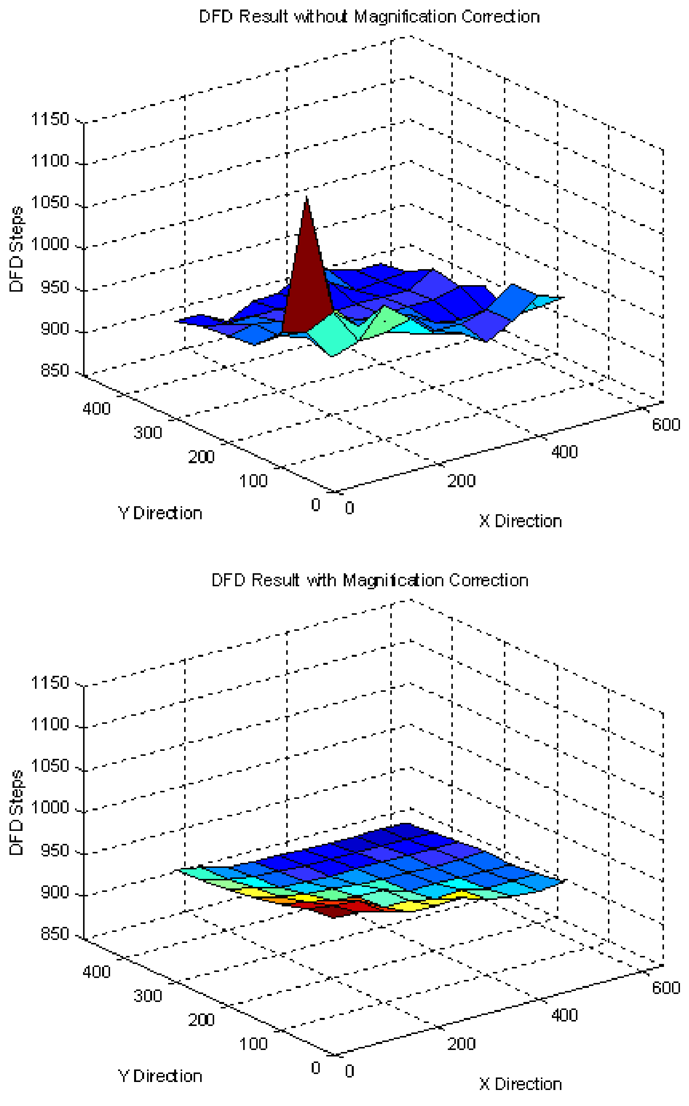

Figure 2.8: Shape of Flat Poster without and with Magnification Normalization

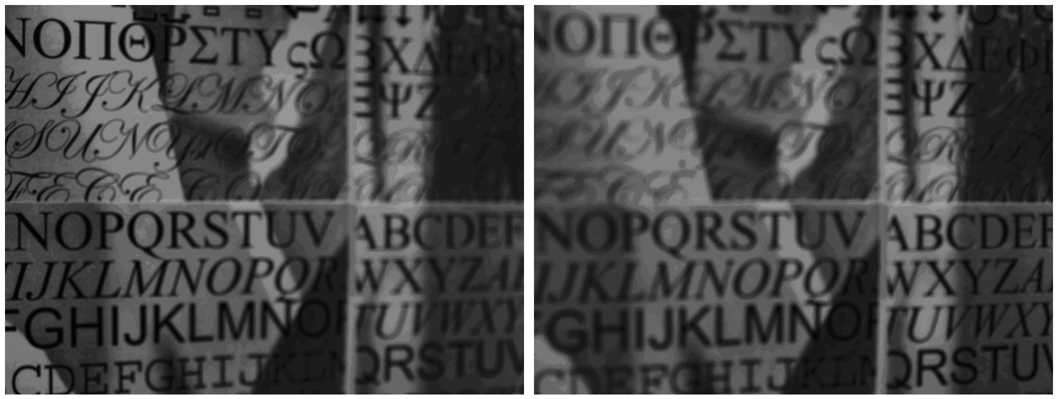

Figure 2.9: Images of Inclined Flat Poster

2.6 Summary

In this chapter, a modified DFD algorithm is presented for cameras operating in macro-mode. Before performing DFD, we first normalize the magnification of both images to ensure the correct comparison between corresponding pixels. The results are substantially better, especially for areas far away from the image center. We also use new lens settings and lookup table specifically for macro-mode. In some range of object distance, a third image is taken and depth is determined through a second lookup table. Such 3 image DFD scheme is found to be very accurate. The RMS error in this case is around 15 steps out of 1500 steps, which is about 1% on average.

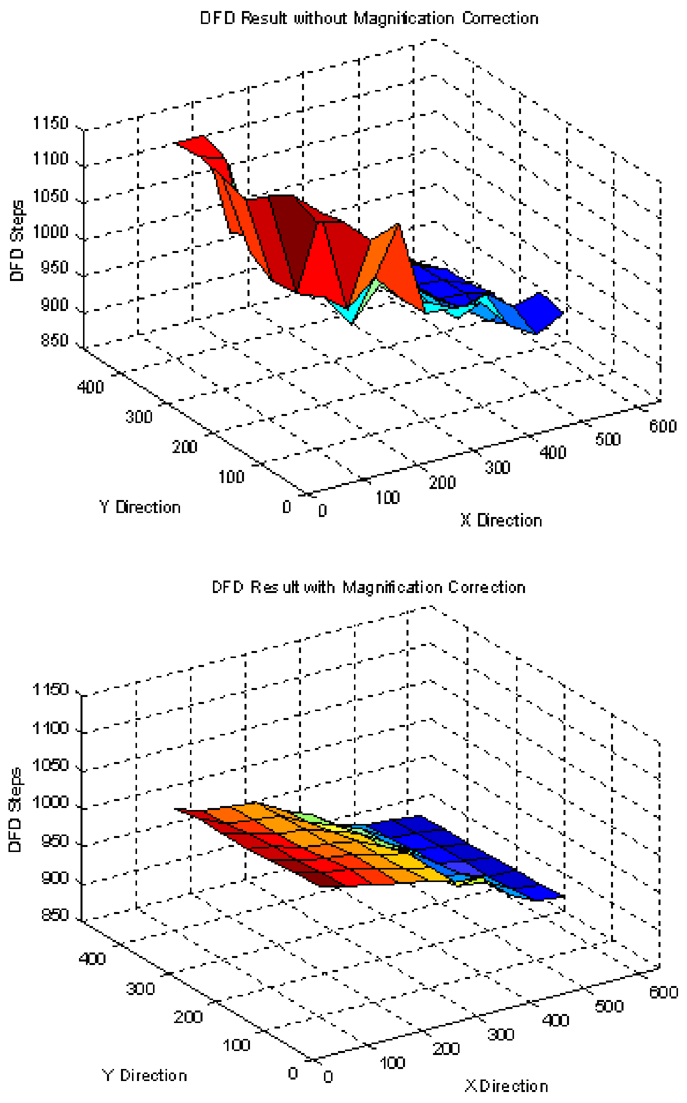

Figure 2.10: Shape of Inclined Poster without and with Magnification Normalization

Chapter 3

Shape from Shift-variant Blurred Images

In Chapter 2, we model the camera system as a linear shift-invariant system. This model is valid only when the object is planar and perpendicular to the optical axis. If an object is of arbitrary shape or various optical abberations are present in the view, this assumption does not hold. One way to approximate shift-variance is to split the image into small blocks and model each block as a shift-invariant system [48, 54]. Others convert shift variant integral equation to inhomogeneous diffusion equation and solve shape related diffusion coefficients through different numerical methods [65]. In this Chapter, we present a new method to extend the S Transform approach to shift-variant case. It is based on the recently developed Rao Transform (RT) [62, 63] for image restoration and linear integral equations. In the new method, the computations are completely localized (i.e. solution at a pixel is computed using image data in a small neighborhood around that same pixel), efficient, and suitable for fine-grain parallel implementation. The effectiveness of the new approach is demonstrated with both simulation and real experiments.

3.1 Rao Transform (RT)

We first illustrate Rao Transform for one dimensional signals. Let us express the shift variant system as

$$g(x) = \int_r^s k(x,\alpha) f(\alpha)\, d\alpha \tag{3.1}$$

We define:

$$h(x,\alpha) = k(x+\alpha, x) \tag{3.2}$$

and

$$\beta = x - \alpha \tag{3.3}$$

If we substitute α with $x-\beta$ in Eq. (3.1), we have:

$$\begin{aligned} g(x) &= \int_r^s k(x,\alpha) f(\alpha)\, d\alpha \\ &= \int_r^s k(x, x-\beta)) f(x-\beta)\, d(x-\beta) \end{aligned} \tag{3.4}$$

where

$$k(x, x-\beta) = k((x-\beta)+\beta, x-\beta) = h(x-\beta, \beta) \tag{3.5}$$

Substituting Eq. (3.5) into Eq. (3.4), we have

$$\begin{aligned} g(x) &= \int_{r}^{s} h(x-\beta,\beta) f(x-\beta)\, d(x-\beta) \\ &= \int_{x-r}^{x-s} h(x-\beta,\beta) f(x-\beta)\, d(\beta) \end{aligned} \tag{3.6}$$

The above transform is termed as Rao Transform (RT), which localizes a global integral equation within a small region around x. Similarly, a two dimensional Rao Transform to model shift-variant image blurring process can be expressed as:

$$g(x,y) = \int_{x-b}^{x-a} \int_{y-d}^{y-c} h(x-u, y-v, u, v) f(x-u, y-v)\, du\, dv \tag{3.7}$$

where $f(x,y)$ is focused image, $h(x,y,u,v)$ is shift-variant Point Spread Function (PSF), and $g(x,y)$ is blurred image.

Eq. (3.7) provides an intuitive and elegant way to model the actual blurring process that takes place in a physical linear shift variant system. It also facilitates the derivation of an explicit closed-form formula for inverting the transform.

3.2 Inverse Rao Transform (IRT)

The shift-variant blurring in Eq. (3.7) can be inverted to solve for the focused image $f(x,y)$ and related shape information. A summary of the inversion are presented below.

We use superscript (m,n) to denote the m^{th} partial derivative with respect x and n^{th} partial derivative with respect to y and the subscript p,q to denote $(p,q)^{th}$

moments with respect to u, v:

$$g^{(m,n)} = \frac{\partial^m}{\partial x^m}\frac{\partial^n}{\partial y^n}g(x,y) \tag{3.8}$$

$$f^{(m,n)} = \frac{\partial^m}{\partial x^m}\frac{\partial^n}{\partial y^n}f(x,y) \tag{3.9}$$

$$h^{(m,n)} = \frac{\partial^m}{\partial x^m}\frac{\partial^n}{\partial y^n}h(x,y,u,v) \tag{3.10}$$

$$h_{p,q}^{(m,n)} = \int_{x-b}^{x-a}\int_{y-d}^{y-c} u^p v^q \frac{\partial^m}{\partial x^m}\frac{\partial^n}{\partial y^n}h(x,y,u,v)\,du\,dv \tag{3.11}$$

for $m, n, p, q = 0, 1, 2, \ldots$.

Using the above notation, the truncated Taylor series expansion of $f(x-u, y-v)$ around (x, y) up to order N and $h(x-u, y-v, u, v)$ around the point (x, y, u, v) up to order M are given by

$$f(x-u, y-v) \approx \sum_{n=0}^{N} a_n \sum_{i=0}^{n} C_i^n u^{n-i} v^i f^{(n-i,i)} \tag{3.12}$$

$$h(x-u, y-v, u, v) \approx \sum_{m=0}^{M} a_m \sum_{j=0}^{m} C_j^m u^{m-j} v^j h^{(m-j,j)} \tag{3.13}$$

where C_i^n and C_j^m denote the binomial coefficients

$$C_p^k = \frac{k!}{p!(k-p)!}$$

and a_n, a_m are defined as

$$a_k = (-1)^k / k! \tag{3.14}$$

Substituting the truncated Taylor-series expansions of h and f into 2D Rao Transform in Eq.(3.7) and simplifying, we get

$$g(x,y) \approx \sum_{n=0}^{N} a_n \sum_{i=0}^{n} C_i^n f^{(n-i,i)} \sum_{m=0}^{M} a_m \sum_{j=0}^{m} C_j^m h_{m+n-i-j,i+j}^{(m-j,j)} \tag{3.15}$$

The above equation can be rewritten as

$$g(x,y) \approx \sum_{n=0}^{N} \sum_{i=0}^{n} S_{n,i} f^{(n-i,i)} \tag{3.16}$$

where

$$S_{n,i} = a_n C_i^n \sum_{m=0}^{M} a_m \sum_{j=0}^{m} C_j^m h_{m+n-i-j,i+j}^{(m-j,j)} \tag{3.17}$$

We can now write expressions for the various partial derivatives of g with respect to x, y as

$$g^{(p,q)} \approx \sum_{n=0}^{N} \sum_{i=0}^{n} \frac{\partial^p}{\partial x^p} \frac{\partial^q}{\partial y^q} [S_{n,i} f^{(n-i,i)}] T(n+p+q) \tag{3.18}$$

where

$$T(n+p+q) = \begin{cases} 1 & \text{if } n+p+q \leq N \\ 0 & \text{otherwise} \end{cases} \tag{3.19}$$

As we assume that $f(x,y)$ can only be expanded up to certain order of Taylor Series, then the terms with derivatives of f of order greater than N are set to zero, for $p+q = 0, 1, 2, \ldots, N$. Note that

$$S_{n,i}^{(p,q)} = \frac{\partial^p}{\partial x^p} \frac{\partial^q}{\partial y^q} S_{n,i} = a_n \mathrm{C}_i^m \sum_{m=0}^{M-(p+q)} a_m \sum_{j=0}^{m} \mathrm{C}_j^m h_{m+n-i-j,i+j}^{m-j+p,j+p} \tag{3.20}$$

and

$$\frac{\partial^p}{\partial x^p}\frac{\partial^q}{\partial y^q}f^{n-i,i} = f^{(n-i+p,i+q)} \tag{3.21}$$

The above equation for $g^{(p,q)}$ for $p, q = 0, 1, 2, \cdots, N$ and $0 \leq p+q \leq N$ constitute $(N+1)(N+2)/2$ equations with as many unknowns $f^{(p,q)}$. The system of equations for $g^{(p,q)}$ can be expressed in a vector-matrix form as

$$\begin{bmatrix} g^{(0,0)} \\ g^{(1,0)} \\ \vdots \\ g^{(0,N)} \end{bmatrix} = \begin{bmatrix} k_{00} & k_{01} & \cdots & \cdots \\ k_{10} & k_{11} & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ & \cdots & & \end{bmatrix} \begin{bmatrix} f^{(0,0)} \\ f^{(1,0)} \\ \vdots \\ f^{(0,N)} \end{bmatrix} \tag{3.22}$$

or

$$\mathbf{g}_{x,y} = \mathbf{K}_{x,y}\ \mathbf{f}_{x,y} \tag{3.23}$$

where the subscripts (x, y) make explicit the dependence of the vectors/matrix on (x, y). This matrix equation can be solved to obtain $f^{(p,q)}$, by inverting the kernel matrix $\mathbf{K}_{x,y}$. The solution can be written in the form

$$\mathbf{f}_{x,y} = \mathbf{K}'_{x,y}\ \mathbf{g}_{x,y} \tag{3.24}$$

where $\mathbf{K}'_{x,y} = \mathbf{K}^{-1}_{x,y}$.

3.3 Shape Recovery

In this section we derive an interesting result that the blur circle radius R corresponding to points on the same planar surface is a linear function of image coordinates. This result provides the crucial relation between the shift-variant blur parameter σ (or blur circle radius R) and the 3D shape parameters of the planar surface patch. Using this relation, we obtain the 3D shape parameters given the blur parameters at each pixel.

Figure 2.1 in Chapter 2 shows a schematic diagram of image formation by a thin lens L with diameter D, focal length f, object point P at distance u, focused image p' at distance v, image detector ID at distance s, and blur circle with radius R. From Chapter 2, we derive that the normalized blur circle radius is :

$$R' = s_0 \frac{D}{2}\left(\frac{1}{f} - \frac{1}{u} - \frac{1}{s}\right) \tag{3.25}$$

where image magnification is normalized with respect to uniform sensor distance s_0. In order to compare images captured with different parameters s, image magnification is normalized with $s = s_0$.

Using a world coordinate system with its origin at the optical center Q and Z-axis along the optical axis, the image p'' of the object point P at (X, Y, Z) has the image coordinates (x, y) given by perspective projection relations:

$$\begin{aligned} x &= s_0 \frac{X}{Z} \\ y &= s_0 \frac{Y}{Z} \end{aligned} \tag{3.26}$$

The tangent plane of object surface at point P can be expressed as

$$Z(X, Y) = Z_0 + Z_X X + Z_Y Y \tag{3.27}$$

where Z_0 is the distance of the plane along the optical axis at P, and Z_X and Z_Y are slopes along the X and Y axes respectively. The three parameters Z_0, Z_X, and Z_Y, are the 3D shape parameters to be determined. Substituting X and Y in Eq. (3.27) with its corresponding image projection x and y using Eq. (3.26), we can express Z as a function of x and y:

$$\begin{aligned} Z &= Z_0 + Z_X X + Z_Y Y \\ &= Z_0 + Z_X \frac{xZ}{s_0} + Z_Y \frac{yZ}{s_0} \end{aligned} \tag{3.28}$$

Rearrange equation:

$$Z\left(1 - \frac{Z_X}{s_0}x - \frac{Z_Y}{s_0}y\right) = Z_0 \tag{3.29}$$

Finally, we obtain:

$$Z(x,y) = \frac{Z_0}{1 - \frac{Z_X}{s_0}x - \frac{Z_Y}{s_0}y} \tag{3.30}$$

Substitute $Z(x,y)$ as object distance u in Eq.(3.25), we can derive the relation between normalized blur circle radius R' and shape parameters Z_0, Z_X and Z_Y:

$$\begin{aligned} R'(x,y) &= s_0\frac{D}{2}(\frac{1}{f} - \frac{1}{Z(x,y)} - \frac{1}{s}) \\ &= s_0\frac{D}{2}(\frac{1}{f} - \frac{1 - Z_X x/s_0 - Z_Y y/s_0}{Z_0} - \frac{1}{s}) \\ &= s_0\frac{D}{2}(\frac{1}{f} - \frac{1}{Z_0} - \frac{1}{s}) + \frac{DZ_X}{2Z_0}x + \frac{DZ_Y}{2S_0}y \\ &= R_0 + R_x x + R_y y \end{aligned} \tag{3.31}$$

where

$$R_0 = \frac{Ds_0}{2}\left(\frac{1}{f} - \frac{1}{Z_0} - \frac{1}{s}\right) \tag{3.32}$$

$$R_x = \frac{DZ_X}{2Z_0} \tag{3.33}$$

$$R_y = \frac{DZ_Y}{2Z_0} \tag{3.34}$$

The normalized blur circle radius R' is a linear function of the image coordinates (x, y) within a planar surface patch. The cylindrical PSF within a planar surface follows:

$$h(x, y, u, v) = \begin{cases} \frac{1}{\pi R'^2(x,y)} & \text{for } \ u^2 + v^2 \leq R'^2 \\ 0 & \text{otherwise} \end{cases}$$

Also note that the blur spread parameter σ, defined as square root of the second central moment of PSF:

$$\sigma^2(x, y) = \iint h(x, y, u, v) u^2 v^2 \, du \, dv \tag{3.35}$$

For real camera systems, σ is known to be proportional to blur circle radius R'[**?**], therefore σ is also a linear function of the image coordinates. The shift-variant Gaussian PSF can be expressed as:

$$h(x, y, u, v) = \frac{1}{2\pi\sigma^2(x, y)} \exp\left(-\frac{u^2 + v^2}{2\sigma^2(x, y)}\right) \tag{3.36}$$

Within the planar surface patch, different moments of Shift-Variant PSF (SV-PSF) at each pixel (x, y) can be explicitly expressed as functions of shape parameters Z_0,

Z_X and Z_Y. From inverse RT (Eq. (3.24)), we can establish two groups of equations for two different blurred images $g_1(x, y)$ and $g_2(x, y)$, which are recored with different camera parameter settings (e.g., $s = s_1$ and $s = s_2$):

$$\mathbf{f} = \mathbf{K}'_1 \ \mathbf{g}_1 \tag{3.37}$$

$$\mathbf{f} = \mathbf{K}'_2 \ \mathbf{g}_2 \tag{3.38}$$

Equating the left hand sides, we obtain

$$\mathbf{K}'_1 \ \mathbf{g}_1 = \mathbf{K}'_2 \ \mathbf{g}_2. \tag{3.39}$$

Given the camera parameters D_i, s_i, and f_i, associated with the blurred images g_i for $i = 1, 2$, the matrix $\mathbf{K}'_i$ (different terms of moments of SV-PSF)can be expressed in terms of the shape parameters Z_0, Z_X and Z_Y. The theory here can be extended to include higher order shape parameters such as surface curvatures (e.g. Z_{XX}, Z_{YY} and Z_{XY}) but in practice such extension does not seem to be useful because of the limited gray level and spatial resolution of image data and noise. Curved surfaces can be adequately approximated by planar polygons (e.g. triangles) on local tangent planes at each pixel. Therefore solving Eq. (3.39) at each pixel (x, y) will provide a solution for 3D shape of the whole object in the scene.

3.4 Experiments

In practice image derivatives are very noise sensitive. Therefore it is necessary to truncate the Taylor Series expansion of the image function f. Experimental results indicated that $N = 2$ and $M = 1$ would provide good results. We also assume that SV-PSF $h(x, y, u, v)$ to be rotationally symmetric, then all odd moments of h become zero:

$$h_{i,j}^{(1,0)} = h_{i,j}^{(0,1)} = 0 \quad \text{if } i \text{ is odd or } j \text{ is odd} \tag{3.40}$$

Also,

$$h_{0,0}^{(0,0)} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x, y, \alpha, \beta)\, d\alpha\, d\beta = 1 \tag{3.41}$$

Since $M = 1$, all derivatives of h of order more than 1 with respect to x and y are zero. In summary,

$$h_{0,0}^{(0,0)} = 1 \tag{3.42}$$

$$h_{1,0}^{(0,0)} = h_{0,1}^{(0,0)} = h_{1,1}^{0,0} = 0 \tag{3.43}$$

$$h_{1,0}^{(1,0)} = h_{1,1}^{(1,0)} = h_{2,1}^{(1,0)} = h_{1,2}^{(1,0)} = h_{3,0}^{(1,0)} = 0 \tag{3.44}$$

$$h_{0,1}^{(0,1)} = h_{1,1}^{(0,1)} = h_{2,1}^{(0,1)} = h_{1,2}^{(0,1)} = h_{0,3}^{(0,1)} = 0 \tag{3.45}$$

Therefore $g^{(0,0)}$ can be express as

$$g^{(0,0)} = f^{(0,0)} + f^{(1,0)} h_{2,0}^{(1,0)} + f^{(0,1)} h_{0,2}^{(0,1)} + \frac{1}{2} f^{(2,0)} h_{2,0}^{(0,0)} + \frac{1}{2} f^{(0,2)} h_{0,2}^{(0,0)} \tag{3.46}$$

In this case, inverse RT in Eq. (3.24) provides the following simplified expression for the focused image $f(x,y) = f^{(0,0)}$:

$$\begin{aligned}
f^{(0,0)} &= g^{(0,0)} - h_{2,0}^{(1,0)} g^{(1,0)} - h_{0,2}^{(0,1)} g^{(0,1)} + 2h_{2,0}^{(1,0)} h_{0,2}^{(0,1)} g^{(1,1)} \\
&+ \left(\frac{3}{2}(h_{2,0}^{(1,0)})^2 + \frac{1}{2}h_{0,2}^{(0,1)} h_{2,0}^{(0,1)} - \frac{1}{2}h_{2,0}\right) g^{(2,0)} \\
&+ \left(\frac{1}{2}h_{2,0}^{(1,0)} h_{0,2}^{(1,0)} + \frac{3}{2}(h_{0,2}^{(0,1)})^2 - \frac{1}{2}h_{0,2}\right) g^{(0,2)} \qquad (3.47) \\
f^{(1,0)} &= g^{(1,0)} - \frac{3}{2}h_{2,0}^{(1,0)} g^{(2,0)} - h_{0,2}^{(0,1)} g^{(1,1)} - \frac{1}{2}h_{0,2}^{(1,0)} g^{(0,2)} \qquad (3.48) \\
f^{(0,1)} &= g^{(0,1)} - \frac{1}{2}h_{2,0}^{(0,1)} g^{(2,0)} - h_{2,0}^{(1,0)} g^{(1,1)} - \frac{3}{2}h_{0,2}^{(0,1)} g^{(0,2)} \qquad (3.49) \\
f^{(2,0)} &= g^{(2,0)} \qquad (3.50) \\
f^{(0,2)} &= g^{(0,2)} \qquad (3.51) \\
f^{(1,1)} &= g^{(1,1)} \qquad (3.52)
\end{aligned}$$

Savitzky-Golay filters are used to compute image derivatives $g_i^{(m,n)}(x,y)$ after smoothing the images with Gaussian filter. As second order derivatives are highly noise sensitive, especially when the image contrast is low, we use data at only those pixels where the image derivative magnitudes were above a preset threshold. For each pixel with reliable data in a small image block, we equate the right sides of the above equations and form a set of non-linear equations with unknowns Z_0, Z_X and Z_Y. Using data at more than 3 nearby pixels will result in an over-constrained set of non-linear equations. These were solved iteratively in a least square sense to solve for the unknowns.

Both simulation and real experiments on simple objects were carried-out. In simulation experiments, a slanted planar object with a chessboard pattern was

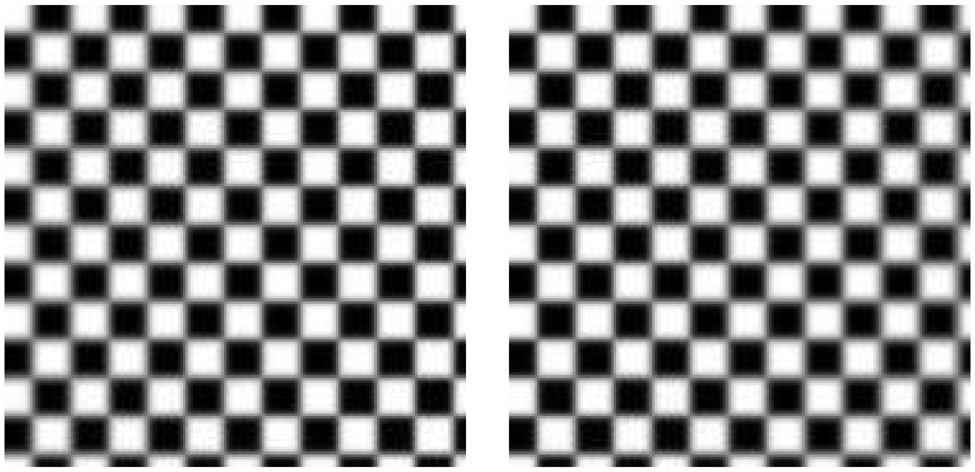

Figure 3.1: Simulated blur images

assumed, and two shift-variant blurred images were generated using corresponding SV-PSFs (shown in Figure 3.1). Then an exhaustive search of the shape parameter space was carried-out to find a solution that minimized an error measure. From the piecewise STM1 method [51], we can get a rough estimation on depth, then from depth between neighboring patches, we could estimate the local slope. Starting from these initial values, we can compare the results between different parameter values around the initials to get the optimum. Solutions in small overlapping image blocks were averaged to obtain the final results. The depth map is presented in Figure 3.4 along with its ground truth for comparison. The average and maximum error in depth were around 1% and 3% respectively.

In real experiments a slanted textured poster on a plane was recorded with a calibrated digital camera at different focus settings (See Figure 3.3). Then, using the same approach as in the simulation experiments, the depth map of the object was computed. The result is shown in Figure 3.4. Images of size 640×480 were processed in blocks of size 30×30 on a 1.6GHz laptop computer with unoptimized code. The

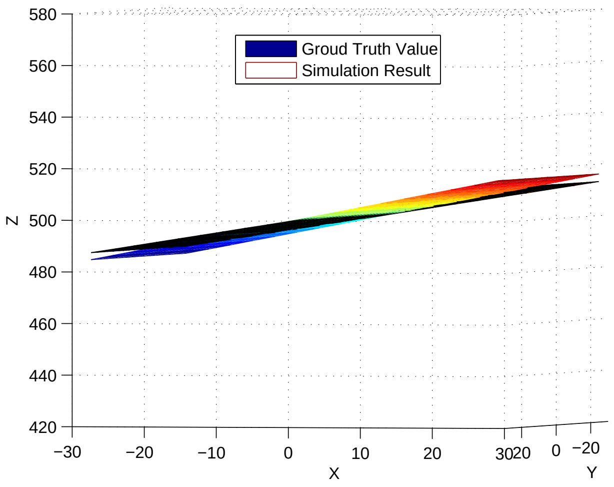

Figure 3.2: Depth map of Recovered scene and Ground truth

processing time was about 5 minutes, most of which was spent on searching for the final solution. This step could be improved substantially.

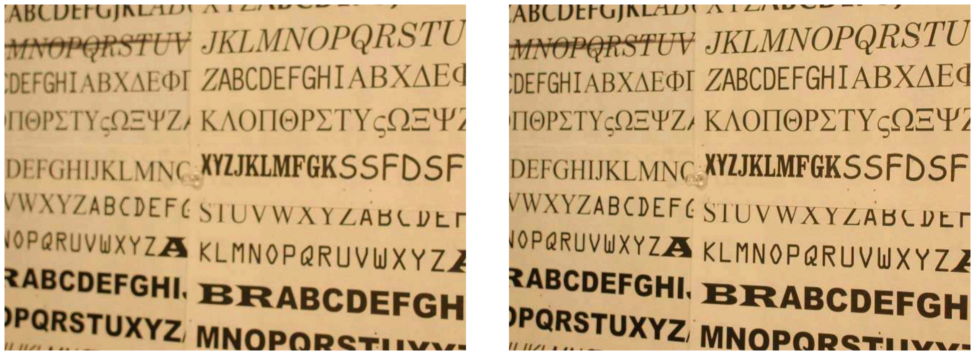

Figure 3.3: Images taken with different camera settings

Results of our experiments show that the new approach presented here is effective and efficient for computing the 3D shape of a planar object using shift-variant blurred images. Unlike approaches based on piecewise constant distance approximation (local convolution or shift-invariant blurring) the results of the new method does not exhibit blocking artifacts in the shape.

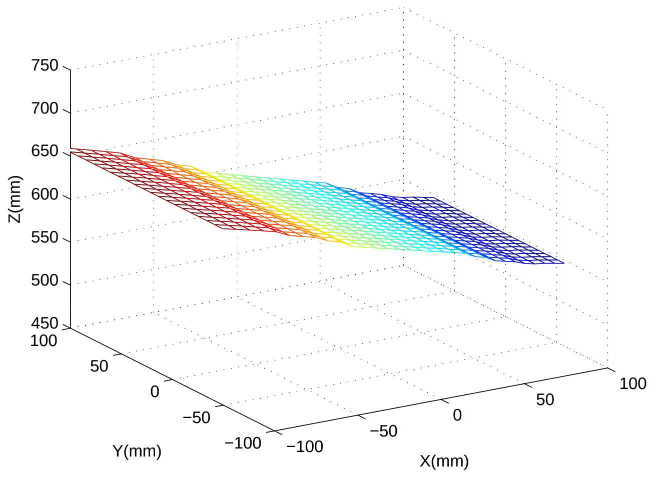

Figure 3.4: Depth Map of Inclined Plane

This method can be extended from planar to more complex 3D shapes. For example, curvature parameters (second-order partial derivatives Z_{XX}, Z_{XY}, and Z_{YY}) can be included in Eq. (3.27) by approximating the surface patch as

$$Z(X, Y) = Z_0 + Z_X X + Z_Y Y + Z_{XX} X^2 + Z_{XY} XY + Z_{YY} Y^2. \tag{3.53}$$

In this case, with two images captured at two different camera parameter settings, image derivatives at six or more (non-degenerate) pixels will need to be used to determine the six shape parameters. Six simultaneous equations corresponding to Eq. (3.39), one at each pixel, is solved. As an alternative, image derivatives at a single pixel can be used by using seven defocused images as follows. A set of simultaneous equations correponding to Eq. (3.39) of the form

$$\mathbf{K}'_i \ \mathbf{g}_i = \mathbf{K}'_{i+1} \ \mathbf{g}_{i+1} \tag{3.54}$$

could be solved for $i = 1, 2, \cdots, 6$ by processing 7 images g_i recorded with 7 different (non-degenerate) camera parameter settings. Note that, in addition to the six shape parameters, the focused image at a pixel is the seventh parameter to be determined. In the case of a planar surface patch, there is an alternative to the method presented earlier based on two defocused images at three or more pixels. Four images captured at four different camera parameter settings and image derivatives at a single pixel can be used to obtain three equations from Eq. (3.54). They can be solved to determine the three shape parameters. These approaches are under investigation.

3.5 Summary

In this Chapter, we present a new theoretical framework to recover 3D shape of object from shift-variant blurred images. It represents a basic theoretical and computational advance in the literature. Experimental results indicate that the method is effective and efficient. In addition to shape, the new method can be used to compute the focused image of the object from its blurred images [63]. The method here will be further improved using regularization and better iterative techniques for solving non-linear system of equations.

Chapter 4

Stereo Camera Calibration

Stereopsis is widely used in many applications such as surface reconstruction and robot vision. Traditional stereo vision systems usually consist of two cameras attached to a mount to implement the epipolar geometry. Recently, some researchers have designed a novel stereo camera as an alternative to the two-camera stereo system. This new architecture features a single digital camera with an adaptor attached in front of the camera lens. The adaptor has a pair of periscopes, where the incoming light is split to form two stereo images as shown in Figure 4.1. Figure 4.2 is a sample stereo image captured by an actual camera with the new architecture. In Figure 4.2, left half and the right half are pictures of the same object from different views.

The calibration of a stereo camera requires estimating two sets of external transformation parameters and one set of intrinsic camera parameters. One straightforward strategy is to separate the left half and the right half image as if they were captured with two distinct cameras, and calibrate them independently using existing state-of-the-art techniques [5, 2, 1]. In practice, we have found that this direct

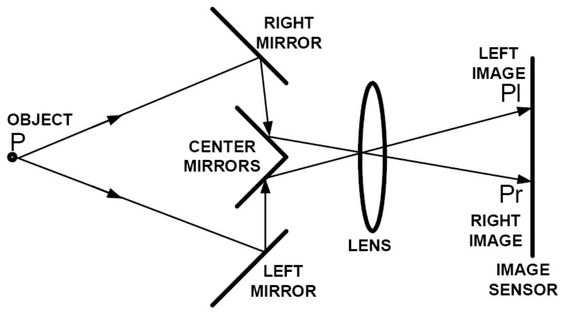

Figure 4.1: Optics of Stereo Adaptor

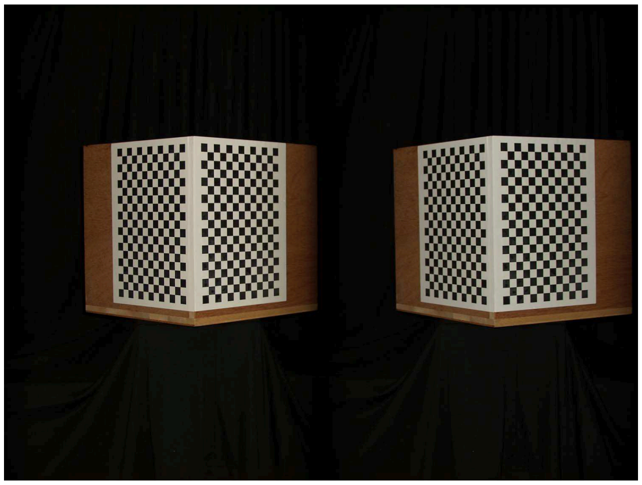

Figure 4.2: Stereo Image of Calibration Pattern

extension is not accurate and reliable, because using information from only half the image sensor is very sensitive to noise. Instead, we present an effective new technique which fully exploits the physical constraints. Our method first provides a closed-form solution for all camera parameters (both intrinsic and extrinsic) from a single stereo image. The initial results are then refined by minimizing reprojected errors between model-computed image points and observed points. We also provide some pre-processing techniques including corner detection and lens distortion correction to improve calibration accuracy. Experiments on a real camera system is shown to validate the effectiveness of our new calibration technique.

4.1 Camera Calibration

Our goal is to find a fast and reliable method to determine all the camera parameters from a single image captured by the new stereo camera system. It is well-known from the principles of projective geometry [5] that reliable calibration requires the use of a 3D non-planar calibration object to avoid degenerate configurations. We use a 3D calibration object where checkerboard patterns are pasted on two perpendicular planar surfaces as shown in Figure 4.2.

4.2 Camera Model

We adopt the conventional pin-hole camera model which assumes that object is in perfect focus. The perspective projection between 3D object and its 2D image projection is modeled by a 3×3 matrix $\mathbf{C}$ as:

$$\mathbf{C} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

where f_x and f_y denote focal lengths in pixel units along x and y axes, u_0 and v_0 specify the position of the optical center, and s denotes the skew factor. Rigid transformation between the world coordinate system and the camera frame is modeled by a rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$, where

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{4.2}$$

is a 3×3 orthogonal matrix and

$$\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{4.3}$$

is a 3×1 vector.

Given the world coordinates of scene points as $\mathbf{X} = (X, Y, Z, W)^t$ and their corresponding image points $\mathbf{u} = (u, v, w)^t$, where W and w are homogeneous scalars, the projection relation between them can be expressed as:

$$\mathbf{u} = \mathbf{C}\,[\mathbf{R}\;\mathbf{t}]\,\mathbf{X}. \tag{4.4}$$

In the case of our novel stereo camera system, image points on the left half and the right half are modeled separately (due to the split optical path) as:

$$\mathbf{u}_l = \mathbf{C}\,[\mathbf{R}_l\;\mathbf{t}_l]\,\mathbf{X} \tag{4.5}$$

$$\mathbf{u}_r = \mathbf{C}\,[\mathbf{R}_r\;\mathbf{t}_r]\,\mathbf{X} \tag{4.6}$$

Note that the intrinsic camera matrices $\mathbf{C}$ for the left and the right half are the same because the stereo image is formed using a single lens and a single sensor. This important constraint will be fully exploited in our calibration scheme.

4.2.1 Closed-form Solution

Given a group of measured 3D scene coordinates $\mathbf{X}_i = (X_i, Y_i, Z_i)$ and their corresponding left and right image coordinates $\mathbf{u}_{li} = (u_{li}, v_{li})$ and $\mathbf{u}_{ri} = (u_{ri}, v_{ri})$, two general 3 projective matrices $\mathbf{P}_l$ and $\mathbf{P}_r$ can be recovered through Direct Linear Transform (DLT) [5] which satisfy:

$$\begin{bmatrix} u_{li} \\ v_{li} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{l1} & \mathbf{p}_{l2} & \mathbf{p}_{l3} & \mathbf{p}_{l4} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{4.7}$$

and

$$\begin{bmatrix} u_{ri} \\ v_{ri} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{r1} & \mathbf{p}_{r2} & \mathbf{p}_{r3} & \mathbf{p}_{r4} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{4.8}$$

where $\mathbf{p}_{lj}$ and $\mathbf{p}_{rj}$ are column vectors of $\mathbf{P}_l$ and $\mathbf{P}_r$ respectively. From Eq. (4.6), we have

$$\begin{bmatrix} \mathbf{p}_{l1} & \mathbf{p}_{l2} & \mathbf{p}_{l3} & \mathbf{p}_{l4} \end{bmatrix} = \lambda_l \mathbf{C} \begin{bmatrix} \mathbf{r}_{l1} & \mathbf{r}_{l2} & \mathbf{r}_{l3} & \mathbf{t}_l \end{bmatrix} \tag{4.9}$$

and

$$\begin{bmatrix} \mathbf{p}_{r1} & \mathbf{p}_{r2} & \mathbf{p}_{r3} & \mathbf{p}_{r4} \end{bmatrix} = \lambda_r \mathbf{C} \begin{bmatrix} \mathbf{r}_{r1} & \mathbf{r}_{r2} & \mathbf{r}_{r3} & \mathbf{t}_r \end{bmatrix} \tag{4.10}$$

where λ_l and λ_r are arbitrary scalars. Employing the orthogonal property of rotation matrix $\mathbf{R}_l$ and $\mathbf{R}_r$, we can derive

$$\begin{aligned}
\mathbf{p}_{l1}^t \mathbf{C}^{-t}\mathbf{C}^{-1}\mathbf{p}_{l2} &= 0 &(4.11)\\
\mathbf{p}_{l2}^t \mathbf{C}^{-t}\mathbf{C}^{-1}\mathbf{p}_{l3} &= 0 &(4.12)\\
\mathbf{p}_{l1}^t \mathbf{C}^{-t}\mathbf{C}^{-1}\mathbf{p}_{l1} &= \mathbf{p}_{l2}^t \mathbf{C}^{-t}\mathbf{C}^{-1}\mathbf{p}_{l2} &(4.13)\\
\mathbf{p}_{l2}^t \mathbf{C}^{-t}\mathbf{C}^{-1}\mathbf{p}_{l2} &= \mathbf{p}_{l3}^t \mathbf{C}^{-t}\mathbf{C}^{-1}\mathbf{p}_{l3} &(4.14)\\
\mathbf{p}_{r1}^t \mathbf{C}^{-t}\mathbf{C}^{-1}\mathbf{p}_{r2} &= 0 &(4.15)\\
\mathbf{p}_{r2}^t \mathbf{C}^{-t}\mathbf{C}^{-1}\mathbf{p}_{r3} &= 0 &(4.16)\\
\mathbf{p}_{r1}^t \mathbf{C}^{-t}\mathbf{C}^{-1}\mathbf{p}_{r1} &= \mathbf{p}_{r2}^t \mathbf{C}^{-t}\mathbf{C}^{-1}\mathbf{p}_{r2} &(4.17)\\
\mathbf{p}_{r2}^t \mathbf{C}^{-t}\mathbf{C}^{-1}\mathbf{p}_{r2} &= \mathbf{p}_{r3}^t \mathbf{C}^{-t}\mathbf{C}^{-1}\mathbf{p}_{r3} &(4.18)
\end{aligned}$$

where $\mathbf{C}^{-t}$ denotes $(\mathbf{C}^{-1})^t$. Note that matrix $\mathbf{B} = \mathbf{C}^{-t}\mathbf{C}^{-1}$ is symmetric, whose degree of freedom is only 6. The above 8 linear equations can be easily solved to determine $\mathbf{B}$ through Singular Value Decomposition (SVD) or Pseudo-Inverse.

¿From $\mathbf{B} = \mathbf{C}^{-t}\mathbf{C}^{-1}$, we can easily derive that $\mathbf{B}^{-1} = \mathbf{C}\mathbf{C}^t$. Because of the scaling factor in projective matrices $\mathbf{P}_l$ and $\mathbf{P}_r$, the actual relation between $\mathbf{B}$ and $\mathbf{C}$ should be $\mathbf{B}^{-1} = \lambda\mathbf{C}\mathbf{C}^t$, i.e.

$$\mathbf{B}^{-1} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} = \lambda \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ u_0 & v_0 & 1 \end{bmatrix} \tag{4.19}$$

The unknowns in camera matrix **C** can then be solved as follows:

$$\lambda = b_{33} \tag{4.20}$$

$$u_0 = b_{13}/\lambda \tag{4.21}$$

$$v_0 = b_{23}/\lambda \tag{4.22}$$

$$f_y = \sqrt{b_{22} - {v_0}^2} \tag{4.23}$$

$$s = (b_{12} - u_0 v_0)/f_y \tag{4.24}$$

$$f_x = \sqrt{b_{11} - s^2 - {u_0}^2} \tag{4.25}$$

After solving the intrinsic camera matrix **C**, the next step is to calculate rotation matrices and translation vectors. From Eq.(4.9) and Eq.(4.10), we have

$$\mathbf{R}_l = 1/\lambda_l \, \mathbf{C}^{-1} \begin{bmatrix} \mathbf{p}_{l1} & \mathbf{p}_{l2} & \mathbf{p}_{l3} \end{bmatrix} \tag{4.26}$$

$$\mathbf{t}_l = 1/\lambda_l \, \mathbf{C}^{-1} \mathbf{p}_{l4} \tag{4.27}$$

$$\mathbf{R}_r = 1/\lambda_r \, \mathbf{C}^{-1} \begin{bmatrix} \mathbf{p}_{r1} & \mathbf{p}_{r2} & \mathbf{p}_{r3} \end{bmatrix} \tag{4.28}$$

$$\mathbf{t}_r = 1/\lambda_r \, \mathbf{C}^{-1} \mathbf{p}_{r4} \tag{4.29}$$

with $\lambda_l = ||\mathbf{p}_{l1}|| = ||\mathbf{p}_{l2}|| = ||\mathbf{p}_{l3}||$ and, similarly, $\lambda_r = ||\mathbf{p}_{r1}|| = ||\mathbf{p}_{r2}|| = ||\mathbf{p}_{r3}||$. In practice, usually, norms of different column vectors would not be equal to each other and thus the resulting rotation matrix **R** would not satisfy the orthogonality constraint. Therefore a standard method [1] to estimate the best rotation matrix from a general 3×3 matrix is used.

4.2.2 Iterative Refinement

The closed-form solution described above tends to minimize the algebraic error of equations (Eq.(4.11)-Eq.(4.18)), which is not physically meaningful. We can further refine the initial result by minimizing geometric errors between reprojected image points and actual image points. Radial distortion can also be incorporated in this process to improve calibration accuracy.

Due to complex design of lens system and imperfect manufacturing, practical lens system usually can not be modeled as linear system (described by camera matrix **C**). It usually shifts image points to new positions along their normalized radius (which is termed as radial distortion). Here we use simple polynomials to model the shift to achieve numerical stability [1].

Let (x_j, y_j) be ideal normalized image projections of scene point (X, Y, Z) through stereo adaptor, where

$$w_j \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_j & \mathbf{t}_j \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.30}$$

and the distorted normalized image coordinates are:

$$r_j = \sqrt{{x_j}^2 + {y_j}^2} \tag{4.31}$$

$$\hat{x}_j = x_j + x_j(k_1 {r_j}^2 + k_2 {r_j}^4) \tag{4.32}$$

$$\hat{y}_j = y_j + y_j(k_1 {r_j}^2 + k_2 {r_j}^4) \tag{4.33}$$

where $j = l, r$, denoting coordinates from left and right half of stereo image and (k_1,

k_2) are radial distortion coefficients. The final observed image coordinates in pixel unit are:

$$\begin{bmatrix} \hat{u}_j \\ \hat{v}_j \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_j \\ \hat{y}_j \\ 1 \end{bmatrix} \tag{4.34}$$

We assume that each detected image coordinate $\hat{\mathbf{u}}'$ is corrupted Gaussian noise with zero mean and uniform standard deviation σ that $\hat{\mathbf{u}}' = \hat{\mathbf{u}} + \Delta u$ and

$$P(\Delta u) = \frac{1}{2\pi\sigma^2} \exp\,(-\frac{||\Delta u||^2}{2\sigma^2}) \tag{4.35}$$

We also assume noises are independent and identically distributed, then the Probability Density Function (PDF) of noise perturbed data is

$$P(\hat{\mathbf{u}}'_1 \hat{\mathbf{u}}'_2 \ldots \hat{\mathbf{u}}'_n) = \prod_{i=1}^{n} \frac{1}{2\pi\sigma^2} \exp\,(-\frac{||\hat{\mathbf{u}}'_i - \hat{\mathbf{u}}_i(\mathbf{C}, k_1, k_2, \mathbf{R}_l, \mathbf{t}_l, \mathbf{R}_r, \mathbf{R}_r)||^2}{2\sigma^2}) \tag{4.36}$$

Then Maximum Likelihood estimate of (MLE) of camera parameters $(\mathbf{C}, \mathbf{R}_l, \mathbf{t}_l, \mathbf{R}_r, \mathbf{t}_r)$ can be obtained by minimizing the following objective function:

$$E = \sum_{i}^{n} ||\hat{\mathbf{u}}'_{li} - \hat{\mathbf{u}}_{li}(\mathbf{C}, k_1, k_2, \mathbf{R}_l, \mathbf{t}_l)||^2 + \sum_{i}^{n} ||\hat{\mathbf{u}}'_{ri} - \hat{\mathbf{u}}_{ri}(\mathbf{C}, k_1, k_2, \mathbf{R}_r, \mathbf{t}_r)||^2 \tag{4.37}$$

where l and r denote where image points lie, on left or right half of stereo image. Because of orthogonality, rotation matrix $\mathbf{R}$ has only 3 degrees of freedom, which can be parameterized as a unit 4-vector quaternion $\mathbf{Q} = (q_0, q_x, q_y, q_z)$, that is:

$$
\begin{aligned}
w &= \sqrt{1 + r_{ii} + r_{jj} + r_{kk}} && (4.38)\\
q_0 &= \frac{r_{kj} - r_{jk}}{2w} && (4.39)\\
q_x &= \frac{w}{2} && (4.40)\\
q_y &= \frac{r_{ij} - r_{ji}}{2w} && (4.41)\\
q_z &= \frac{r_{ki} - r_{ik}}{2w} && (4.42)
\end{aligned}
$$

where r_{ii} is the largest diagonal element of $\mathbf{R}$ and ijk is an even permutation of 123 (i.e., 123, 231 or 312).

Up till now, we assumed that the 3D measurement of corner points on the calibration object is precise, but in practice, such assumption rarely holds due to limited accuracy of mesurements. It has been demonstrated in [1] and [103] that the systematic error from measurement has more effect than the random error. To further improve the accuracy, we can also estimate 3D coordinates $\mathbf{X}_i$ in the refinement step. As measurement errors are within certain range, it is natural to limit the search range of $\mathbf{X}_i$ in the parameter space to improve the speed. Now the new objective function becomes

$$E = \sum_i^n ||\mathbf{u}_{li} - \hat{\mathbf{u}}_{li}(\mathbf{C}, k_1, k_2, \mathbf{Q}_l, \mathbf{t}_l, \mathbf{X}_i)|| + \sum_i^n ||\mathbf{u}_{ri} - \hat{\mathbf{u}}_{ri}(\mathbf{C}, k_1, k_2, \mathbf{Q}_r, \mathbf{t}_r, \mathbf{X}_i)|| \quad (4.43)$$

subject to

$$||\mathbf{X}_i - \mathbf{X}_{i0}|| \leq \delta \quad (4.44)$$

and

$$||\mathbf{Q}_l|| = 1 \tag{4.45}$$

$$||\mathbf{Q}_r|| = 1 \tag{4.46}$$

where $\mathbf{X}_i$ are estimated 3D coordinates of corners and $\mathbf{X}_{i0}$ are the originally measured ones.

We use Interior Point algorithm [104] to solve this constrained optimization problem. Interior Point method starts from a group of initial guesses of estimated parameters and improves the results step by step by evaluating the Jacobian $\mathbf{J}_E$ and Hessian matrix $\mathbf{H}_E$ of the objective function. Directly computing the large Hessian matrix could be slow, but its sparse structure can be utilized to speed up the process. As there is no interaction between different corners, we obtain

$$\frac{\partial E}{\partial X_i}\frac{\partial E}{\partial X_j} = 0 \tag{4.47}$$

which results in a large number of zero entries in the Hessian matrix $\mathbf{H}_E$. Its sparse structure is illustrated in Figure 4.3, where nonzero entries exist only along diagonal, first few horizontal, and vertical bands. These few nonzero entries can be computed analytically and the overall computation cost is acceptable.

The initial values of camera parameters ($\mathbf{C}_0$, $\mathbf{R}_{l0}$, $\mathbf{t}_{l0}$, $\mathbf{R}_{r0}$, and $\mathbf{t}_{r0}$) can be calculated using the linear method described in Section 4.2.1. (k_1, k_2) can be roughly estimated using methods described in Section 4.3.2, and initial values of 3D corners $\mathbf{X}_{i0}$ are measured. The optimization converges after a few iterations and generate reliable results (illustrated in Section 4.4).

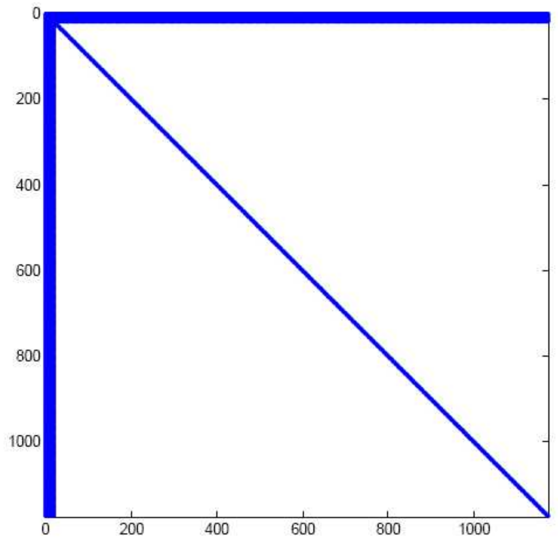

Figure 4.3: Sparse Structure of Hessian Matrix

4.3 Pre-processing

Before applying the above calibration method, we have to detect all the inner corners in the stereo image. The following part describes the corner detection scheme and the estimation method for initial distortion coefficients.

4.3.1 Corner Detection

We adopt a Harris corner detector based algorithm [105] to detect inner corner points on a checkerboard pattern. The first step is to compute the function:

$$c(u, v) = det(M) - \lambda(trace(M))^2 \tag{4.48}$$

with

$$M = \begin{bmatrix} (\frac{\partial I}{\partial u})^2 \otimes g(u,v) & (\frac{\partial I}{\partial u}\frac{\partial I}{\partial v}) \otimes g(u,v) \\ (\frac{\partial I}{\partial u}\frac{\partial I}{\partial v}) \otimes g(u,v) & (\frac{\partial I}{\partial v})^2 \otimes g(u,v) \end{bmatrix} \tag{4.49}$$

where $g(u, v)$ is a smoothing Gaussian filter, $I(u, v)$ is the intensity of checkerboard pattern image and λ is a tuning operator.

Function $c(u, v)$ will show prominent positive peaks in correspondence to corners, including T, X, and L junctions. Thresholding followed by local maximum search can reveal integer corner positions $\mathbf{p}(u, v)$. To exclude the L-junctions, a heuristic method is adopted. Let $I_{n\times n}$ be a small $n \times n$ block with detected corner $\mathbf{p}$ as its center, and I_{ne}, I_{se}, I_{nw}, I_{sw} be the intensity of its four vertices. If any of absolute intensity differences between two orthogonal vertices (e.g., $|I_{ne} - I_{sw}|$) is larger than that between adjacent vertices (e.g., $|I_{ne} - I_{nw}|$, $|I_{ne} - I_{se}|$), then this corner will be marked as L-junction and removed from the detected corner list.

Subpixel refinement can be achieved by quadratic fitting of the intensity space within a small neighborhood $(s, t) \in [u - \Delta u, u + \Delta u] \times [v - \Delta v, v + \Delta v]$. Minimizing the following equation can retrieve the quadratic parameters a, b, c, d and e.

$$E = ||as^2 + bst + ct^2 + ds + et + f - I(u, v)|| \tag{4.50}$$

The X-junction is the critical point (saddle point) of this hyperbolic paraboloid, which is also the intersection of two lines:

$$2as + bt + d = 0 \tag{4.51}$$

$$bs + 2ct + e = 0 \tag{4.52}$$

Such quadratic fitting scheme requires a preliminary interpolation to achieve certain accuracy, which is computationally expensive. Instead, we use a simpler approximation to compute subpixel saddle point. Assuming the Gaussian smoothed image around $\mathbf{p}(u, v)$ can be exactly fitted by the quadratic equation, the coefficients a, b, c, d and e will be partial derivatives of intensity I at (u, v):

$$a = \frac{1}{2}I_{uu} \tag{4.53}$$

$$b = I_{uv} \tag{4.54}$$

$$c = \frac{1}{2}I_{vv} \tag{4.55}$$

$$d = I_u \tag{4.56}$$

$$e = I_v \tag{4.57}$$

and the subpixel saddle point $\mathbf{p}(u + s_0, v + t_0)$ lies at

$$s_0 = \frac{I_v I_{uv} - I_u I_{vv}}{I_{uu} I_{vv} - I_{uv}^2} \tag{4.58}$$

$$t_0 = \frac{I_v I_{uu} - I_u I_{uv}}{I_{uu} I_{vv} - I_{uv}^2} \tag{4.59}$$

The partial derivatives can be computed by Savitzky-Golay filter, and the subpixel position can be solved directly. In Figure 4.4, all detected inner corners are marked with a red cross.

4.3.2 Lens Distortion Correction

The multi-parameter space being searched during refinement step (described in Section 4.2.2) is rarely convex in practice and the iterative process can easily

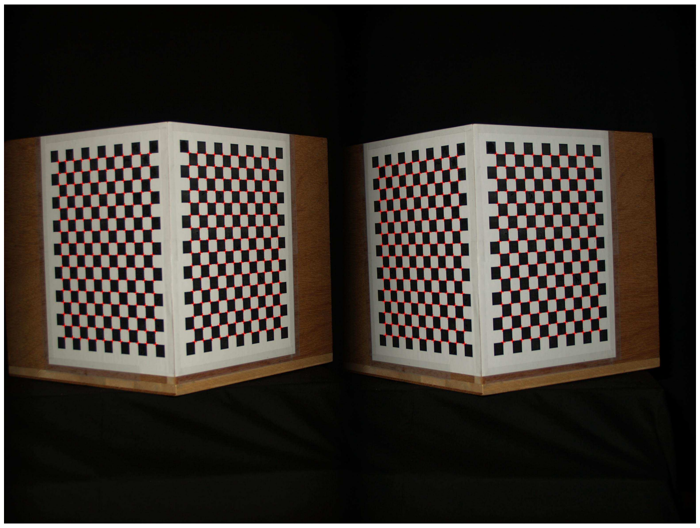

Figure 4.4: Pattern Image with Inner Corners Marked

converge to a local minimum. Therefore the accuracy of initial values becomes crucial. The main limitation of our closed form solution (which served as initial guess for refinement) is that it models camera as a linear system. To make this assumption valid, we have to first roughly remove the distortion effect from the stereo image.

As described in Section 4.2.2, radial lens distortion is modeled as a simple polynomial:

$$\hat{r} = r(1 + k_1 r^2 + k_2 r^4) \tag{4.60}$$

where $\hat{r}$ is distorted normalized radius, and r is ideal normalized radius. Recovering ideal image position from distorted one requires solving the above quintic equation. There is no direct analytic method to find the roots, so we use Taylor Expansion to derive an approximate solution. We denote

$$\begin{aligned} f(r) =& \hat{r} - r = k_1 r^3 + k_2 r^5 \\ =& f(\hat{r} + (r - \hat{r})) \\ =& f(\hat{r}) + f'(\hat{r})(r - \hat{r}) \end{aligned} \tag{4.61}$$

Substituting $f(\hat{r})$ with $k_1\hat{r}^3 + k_2\hat{r}^5$ and $f'(\hat{r})$ with $3k_1\hat{r}^2 + 5k_2\hat{r}^4$, we obtain

$$\hat{r} - r = k_1\hat{r}^3 + k_2\hat{r}^5 + (3k_1\hat{r}^2 + 5k_2\hat{r}^4)(r - \hat{r}) \tag{4.62}$$

We can then easily obtain the inverse lens distortion model:

$$r = \frac{\hat{r} + 2k_1\hat{r}^3 + 4k_2\hat{r}^5}{1 + 3k_1\hat{r}^2 + 5k_2\hat{r}^4} \tag{4.63}$$

If expressed in pixel unit, they are:

$$u = u_0 + \hat{u}\frac{1 + 2k_1\hat{r}^2 + 4k_2\hat{r}^4}{1 + 3k_1\hat{r}^2 + 5k_2\hat{r}^4} \tag{4.64}$$

$$v = v_0 + \hat{v}\frac{1 + 2k_1\hat{r}^2 + 4k_2\hat{r}^4}{1 + 3k_1\hat{r}^2 + 5k_2\hat{r}^4} \tag{4.65}$$

where (u_0, v_0) is the principle point, $(\hat{u}, \hat{v})$ are distorted (actual observed) image coordinates and (u, v) are the ideal image coordinates.

As perspective projection retains collinearity, lens distortion is the sole factor making straight lines appear curved. We exploit this characteristic to estimate distortion coefficients k_1 and k_2. To exclude errors from optical misalignment, we detach the stereo adaptor and take an image of a planar object shown in Figure 4.5.

There are n straight lines in the image, where $(\hat{u}_{ij}, \hat{v}_{ij})$ are the j^{th} point lying on the i^{th} line. They satisfy:

$$u_{ij} = u_0 + \hat{u}_{ij} \tfrac{1+2k_1\hat{r}_{ij}^2+4k_2\hat{r}_{ij}^4}{1+3k_1\hat{r}_{ij}^2+5k_2\hat{r}_{ij}^4} \tag{4.66}$$

$$v_{ij} = v_0 + \hat{v}_{ij} \tfrac{1+2k_1\hat{r}_{ij}^2+4k_2\hat{r}_{ij}^4}{1+3k_1\hat{r}_{ij}^2+5k_2\hat{r}_{ij}^4} \tag{4.67}$$

$$a_i u_{ij} + b_i v_{ij} + c_i = 0 \tag{4.68}$$

Naturally, we can formulate an objective function:

$$F = \sum\nolimits_{i,j} ||a_i u_{ij} + b_i v_{ij} + c_i|| \tag{4.69}$$

Minimizing F with regard to $(k_1, k_2, a_i, b_i, c_i)$ $(i = 1, 2, \ldots, n)$ can retrieve the distortion coefficients. The initial values of a_i, b_i and c_i can be calculated through line fitting of $(\hat{u}_{ij}, \hat{v}_{ij})$, and (k_1, k_2) can be set to zero. Principle point (u_0, v_0) is assumed to coincide with image center, and focal length can be read from EXIF data.

After getting the lens distortion coefficients k_1 and k_2, we apply inverse model Eq.(4.64) and Eq.(4.65) to recover ideal corner coordinates (u_i, v_i) from the detected ones $(\hat{u}_i, \hat{v}_i)$. Then use (u, v) to compute camera parameters through closed-form method (described in Section 4.2.1). Because of coupling effects between optical center and distortion, k_1 and k_2 estimated from straight line based method has to be adjusted after getting the new optical center. They serve as initial values in refinement step (described in Section 4.2.2). Figure 4.5 is an original image and Figure 4.6 shows the distortion correction result.

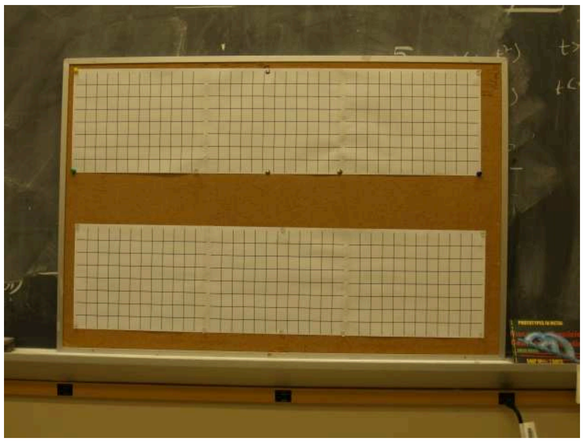

Figure 4.5: Original Image

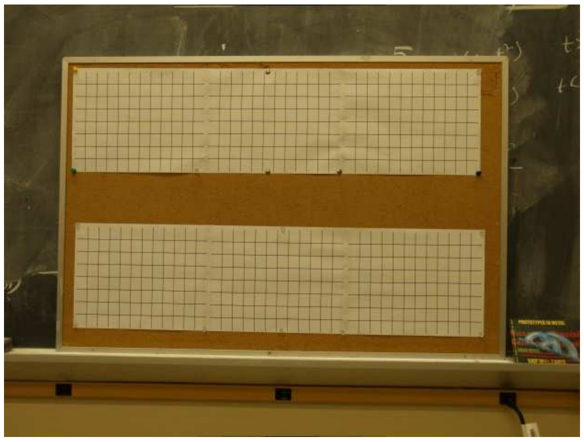

Figure 4.6: Distortion Corrected Image

4.4 Experimental Results

The calibration algorithm described above is implemented and evaluated on our stereo camera system. We first detach the stereo adaptor and estimate lens distortion using straight line based method (described in Section 4.3.2). The distortion corrected image of figure 4.5 is shown in figure 4.6, where we can easily notice that curved lines become straight. This also illustrates that this correction step is necessary to make valid the linear camera model used in the closed-form method.

Next we attach the stereo adaptor and capture stereo images of a chessboard pattern (see Figure 4.2). The inner corners' 3D world coordinates are carefully measured. Their 2D projections are detected using the method described in Section 4.3.1 and corrected using inverse lens distortion model (Eq. (4.64-4.65)). We obtain initial estimation of camera parameters $\mathbf{C}_0$, $\mathbf{R}_{l0}$, $\mathbf{t}_{l0}$, $\mathbf{R}_{r0}$ and $\mathbf{t}_{r0}$ through closed-form formula. They are then refined together with lens distortion coefficients and 3D corner positions by minimizing reprojected errors (described in Section 4.2.2).

The calibration pattern is placed on a desk about 1.2 m away from the stereo camera. We take five images (camera's resolution set at 3136×2352) of the pattern with slightly different viewing directions. Results of each trial are presented in Table 4.1 and Table 4.2.

Table 4.1: Intrinsic Parameters

Trial	f_x	f_y	s	u_0	v_0	k_1	k_2
1	4457.9	4446.5	-8.17	1595.3	1156.1	-0.130	0.101
2	4463.2	4453.1	-8.13	1599.8	1160.5	-0.121	0.011
3	4460.6	4447.9	-7.79	1597.9	1155.0	-0.132	0.112
4	4467.3	4457.6	-7.81	1600.0	1157.4	-0.121	0.018
5	4463.4	4453.4	-7.83	1596.5	1161.6	-0.123	0.033
Mean	4462.49	4451.69	-7.95	1597.92	1158.11	-0.125	0.055
STD	3.51	4.50	0.19	2.07	2.85	0.005	0.048

The rotation and translation vectors obtained from calibration are with respect to the world coordinate system, but this world coordinate system is fixed on the calibration object (origin and three axes are fixed on the object). Therefore, this coordinate system varies with respect to the camera coordinate system from one experimental trial to another because the object position and orientation are changed.

We present the extrinsic parameters in a fixed coordinate system [106] with respect to the camera to evaluate their consistency. The new coordinate system's origin is the center of left virtual camera. Its X axis is parallel to baseline, Y axis is orthogonal to X axis and original left Z axis, and Z axis is orthogonal to XY plane. Within this new coordinate system, left virtual camera sits at the origin and right virtual camera is on new X axis at (Baseline, 0, 0), and they are both rotated. Then there are 7 extrinsic parameters in total, 2×3 parameters to describe left and right rotation (rotation is parameterized as yaw, pitch, roll) and baseline. Conversion from rotation matrix to yaw α, pitch β and roll γ can be calculated according to following equations:

$$\alpha = \arctan\left(\frac{r_{21}}{r_{11}}\right) \tag{4.70}$$

$$\beta = \arctan\left(\frac{-r_{31}}{\sqrt{{r_{32}}^2 + {r_{33}}^2}}\right) \tag{4.71}$$

$$\gamma = \arctan\left(\frac{r_{32}}{r_{33}}\right) \tag{4.72}$$

The choice of quadrants for inverse tangent function is decided by the sign of numerator and denominator of the argument.

Table 4.2: Extrinsic Parameters

Trial	$\alpha_l(^\circ)$	$\beta_l(^\circ)$	$\gamma_l(^\circ)$	$\alpha_r(^\circ)$	$\beta_r(^\circ)$	$\gamma_r(^\circ)$	Baseline(mm)
1	0.14	12.40	0.03	0.14	-10.59	-0.15	97.56
2	0.08	12.68	0.02	0.06	-10.31	-0.15	97.91
3	0.15	12.54	0.03	0.15	-10.42	-0.15	97.75
4	0.05	12.66	0.01	0.05	-10.31	-0.16	98.01
5	0.07	12.47	0.01	0.05	-10.52	-0.15	97.81
Mean	0.10	12.55	0.02	0.09	-10.43	-0.15	97.81
STD	0.04	0.12	0.01	0.05	0.13	0.00	0.17

Results presented in Table 4.1 and Table 4.2 show that our calibration method is very reliable. The standard deviation of all parameters are very small. We also notice that the designed baseline value of stereo adaptor is about 100 mm, while the mean value of our result is 97.81mm, showing that the system error is small. Another way to evaluate the accuracy is to compare the reconstructed 3D positions of corner points with their actual 3D coordinates. The average RMS error between corresponding reconstructed point and original point is about 2 mm. The result is presented in Figure 4.7.

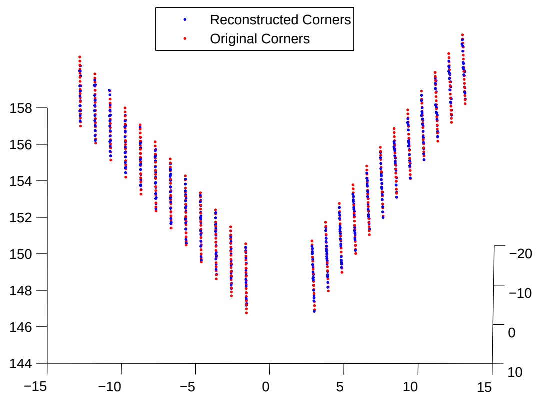

Figure 4.7: Reconstructed 3D Corners

Besides evaluating the consistency of our algorithm, we also check the convergence speed of iterative refinement. Figure 4.8 plots the RMS residual during iteration. We find that the mean error distance quickly decreases to less 1 pixel after several iterations.

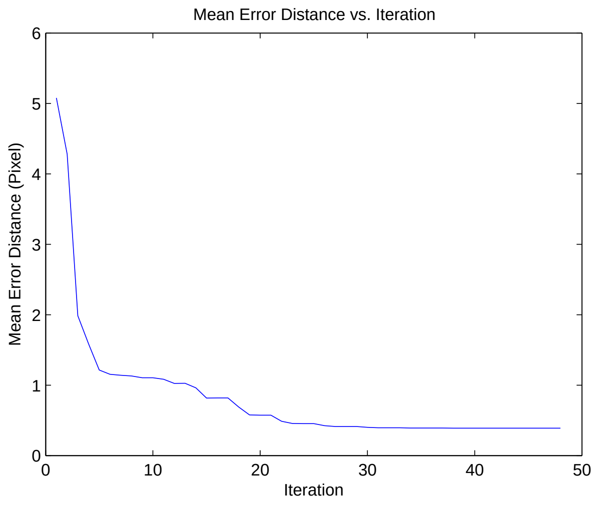

Figure 4.8: Convergence of Mean Error Distances

4.5 Summary

In this Chapter, we presented a new calibration technique for a novel stereo camera architecture and demonstrated its effectiveness as a stable, accurate, and useful technique in practical applications. We proposed a closed-form solution utilizing the constraint that intrinsic parameters are the same. Then we refined the initial result by minimizing the reprojected geometric errors incorporating both lens distortion and 3D corner position estimation. We exploit the sparse structure of the objective function's Hessian matrix to improve the speed. We also proposed some pre-processing techniques including feature corner identification and lens distortion correction to decrease the system error of initial estimation. Experimental results on a real camera system show that our method is accurate and reliable.

Chapter 5

Stereopsis

In Chapter 2 and 3, we introduced Depth from Defocus and Shape from Shift-variant Blurring. These techniques require the camera to take images with different lens settings and recover depth by analyzing the extent of image blur. In this Chapter, we present our research on another shape recovery technique: Stereopsis. As in the human visual system which has two eyes, stereopsis involves two cameras placed at different positions with fixed lens setting (as shown in Figure 5.1). Images of the same object are taken with two cameras. After identifying corresponding image points p and p' in each frame of a common object point P, we can recover the 3D position of P by intersecting Op and $O'p'$.

5.1 Epipolar Geometry

Let us consider a stereo rig composed of two pinhole cameras (Figure 5.2). Let C_1 and C_2 be the optical centers of the two cameras. Object point W is projected onto both image planes at M_1 and M_2. Its epipolar plane is defined by intersecting

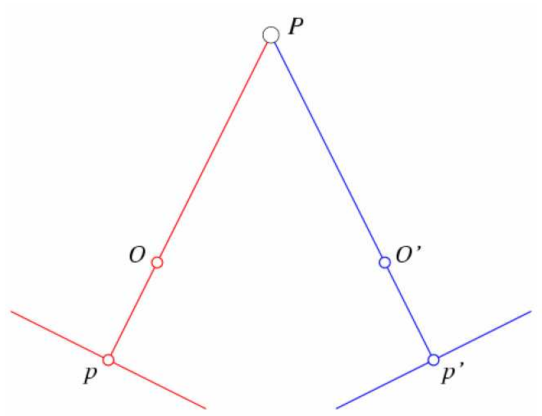

Figure 5.1: Simple Triangulation

rays WC_1 and WC_2. Epipolar line is the intersection of the epipolar plane and the image plane, i.e., line M_1E_1 and M_2E_2. C_1C_2 is the baseline connecting two optical centers. E_1 and E_2 are intersections of baseline and two image planes termed as epipoles. Given a point M_1 in one image, its corresponding point M_2 in the other image should lie on the epipolar line. All the epipolar lines pass through the epipole.

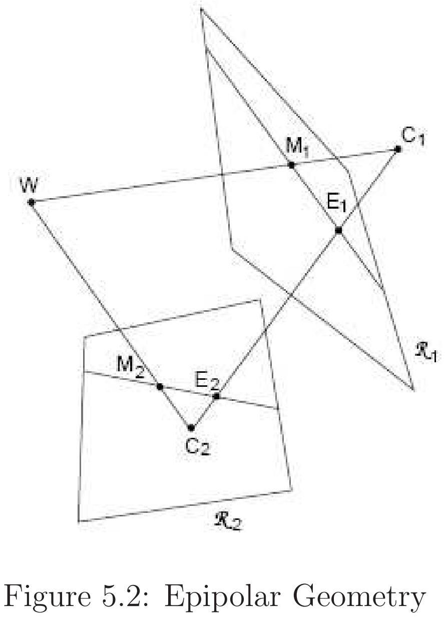

Figure 5.2: Epipolar Geometry

5.2 Stereo Camera System

Traditional stereo vision systems consist of two vergence cameras (as illustrated in Figure 5.2). Even if two cameras of the same type with fixed settings are used, they still present slight differences in image characteristics such as brightness, contrast, and noise level. This difference often introduces errors in stereo matching and texture mapping. To overcome these problems, Olympus Corporation developed a new type of stereo camera featuring an ordinary digital camera and an attached stereo adapter (shown in Figure 5.3).

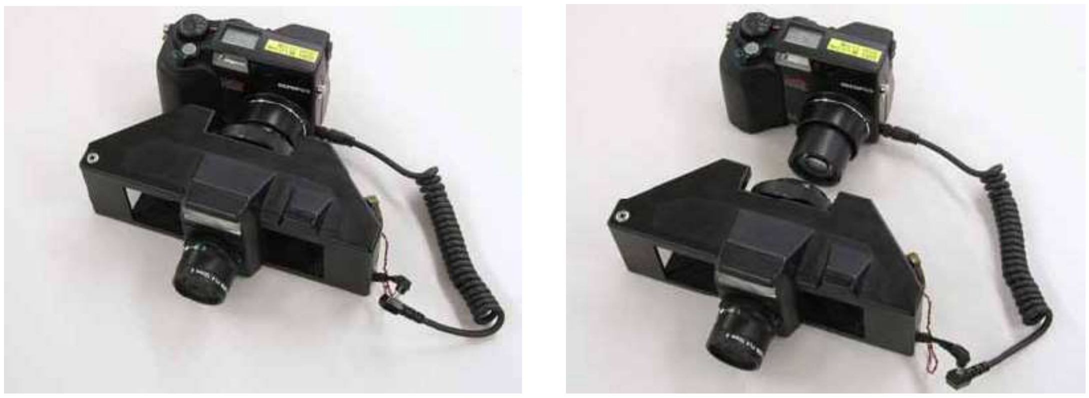

Figure 5.3: Stereo Camera

As shown in Figure 4.1 in Chapter 4, the stereo adapter features a pair of parallel mirrors which splits the incoming light and forms a stereo image pair on the single sensor. This new architecture has many advantages over traditional two-camera stereo system in terms of compactness, cost and accuracy. The stereo adaptor also carries a small pattern projector which can project a random dot pattern onto the object surface during image capture. The camera is programmed to shoot two images consecutively within a short time period, one with pattern projected and another one an ordinary texture image. The pattern image is used for shape recovery as the random dots add more contrast on texture-less surfaces (like human face), and the texture image is mainly used for photorealistic rendering. Example of pattern image and texture are shown in Figure 5.4 and Figure 5.5.

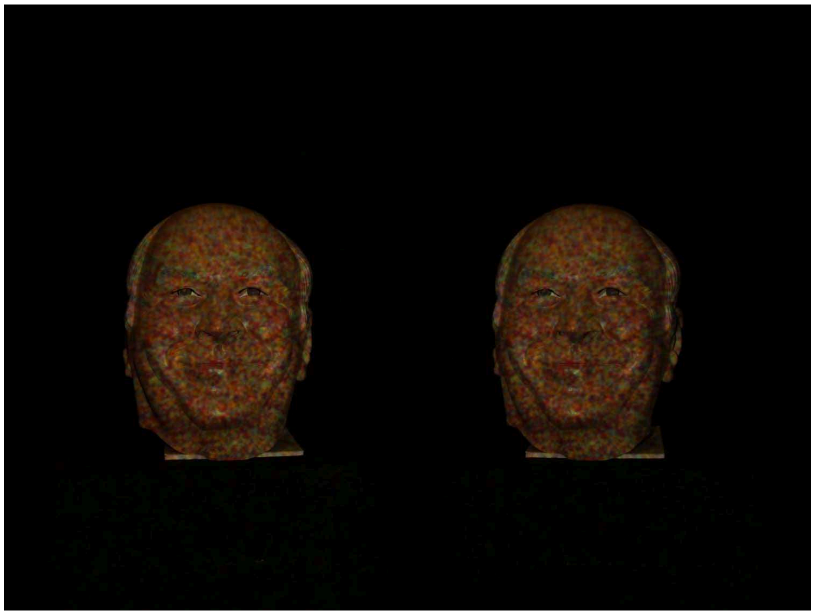

Figure 5.4: Stereo Images with Pattern Projection

The brightness of the projected pattern has an effective working range between 1.2 m and 1.8 m. Beyond this range, the pattern fades away and therefore objects outside the range appear very dark in the image. This feature together with a dark background setting largely simplifies the object segmentation from the background in stereo matching.

5.3 Shape Reconstruction

The stereo camera is first calibrated using techniques presented in Chapter 4. As described before, one set of intrinsic camera parameters and two sets of extrinsic

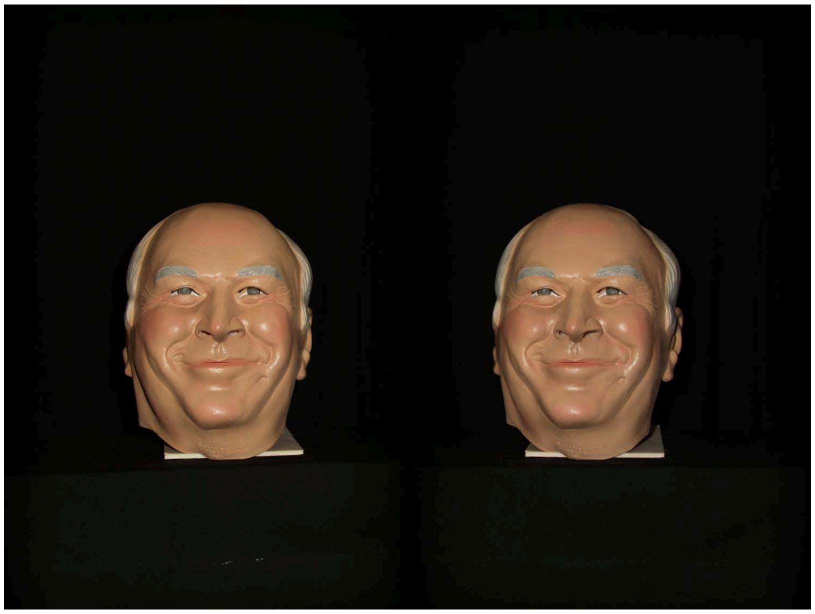
Figure 5.5: Normal Textured Stereo Images

parameters are estimated. If $\mathbf{X} = (X, Y, Z, W)^t$ is 3D object point, $\hat{\mathbf{u}}_l = (\hat{u}_l, \hat{v}_l, \hat{w}_l)^t$ and $\hat{\mathbf{u}}_r = (\hat{u}_r, \hat{v}_r, \hat{w}_r)^t$ are corresponding images points in stereo image, then their relation follows:

$$\mathbf{u}_l = \mathbf{C}\,[\mathbf{R}_l\,\mathbf{t}_l]\,\mathbf{X} \tag{5.1}$$

$$\mathbf{u}_r = \mathbf{C}\,[\mathbf{R}_r\,\mathbf{t}_r]\,\mathbf{X} \tag{5.2}$$

where $\mathbf{C}$ is camera matrix, $\mathbf{R}_l$ and $\mathbf{R}_r$ are rotation matrices, $\mathbf{t}_l$ and $\mathbf{t}_r$ are translation vectors, $\mathbf{u}_l$ and $\mathbf{u}_r$ are ideal image points.

Because of radial lens distortion effects, the actual observed image coordinates are shifted along the radius:

$$\hat{u}_j = u_j + (u_j - u_0)\left[k_1({x_j}^2 + {y_j}^2) + k_2({x_j}^2 + {y_j}^2)^2\right] \quad (5.3)$$

$$\hat{v}_j = u_j + (v_j - v_0)\left[k_1({x_j}^2 + {y_j}^2) + k_2({x_j}^2 + {y_j}^2)^2\right] \quad (5.4)$$

where j = l, r denotes left and right half of stereo image, k_1 and k_2 are distortion coefficients, (x_j, y_j) are ideal normalized image coordinates and (u_0, v_0) is principle point in pixel unit.

5.3.1 Rectification

In Section 5.1, it is proved that corresponding point in the other image must lie on the epipolar line. If baseline is parallel to both image planes, then the epipoles would be at infinity and epipolar lines are a bundle of parallel horizontal lines. With this special configuration, correspondence search is largely simplified as its location is constrained on the same horizontal scanline in the other image.

In practice, we usually adopt the vergence setting (two cameras are placed with an angle, illustrated in Figure 5.2) rather than parallel setting to retain certain length of baseline as well as large viewing filed. Our stereo camera can be viewed as two virtual cameras with vergence setting. However, the stereo image pair can be transformed so that epipolar lines are parallel and horizontal in each image. Such procedure is termed as image rectification [106].

The idea of rectification is to define two new Perspective Projection Matrices (PPM) $\mathbf{P}_{ln}$ and $\mathbf{P}_{rn}$ obtained by rotating the old ones around their optical centers until the focal planes become coplanar (as illustrated Figure 5.6). This ensures that epipoles are at infinity and therefore the epipolar lines will be parallel. To get horizontal epipolar lines, the baseline must be parallel to the new X axis of both cameras. In addition, to have a proper rectification, corresponding points must have the same vertical coordinate. This is obtained by requiring that the new cameras have the same intrinsic parameters.

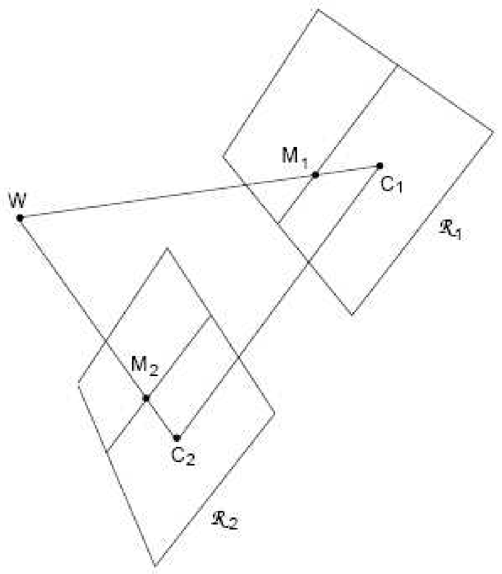

Figure 5.6: Rectified Cameras

The calibrated PPMs for stereo camera are:

$$\mathbf{P}_l = \mathbf{C}\,[\mathbf{R}_l\,\mathbf{t}_l] \tag{5.5}$$

$$\mathbf{P}_r = \mathbf{C}\,[\mathbf{R}_r\,\mathbf{t}_r] \tag{5.6}$$

Two virtual cameras' optical centers in world coordinate system are at

$$\mathbf{c}_l = -\mathbf{R}_l{}^{-1}\mathbf{t}_l \tag{5.7}$$

$$\mathbf{c}_r = -\mathbf{R}_r{}^{-1}\mathbf{t}_r \tag{5.8}$$

To satisfy the requirements of rectification, both virtual cameras are rotated to the same pose, which means that they share the same rotation matrix $\mathbf{R}_n$ that:

$$\mathbf{R}_n = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix} \tag{5.9}$$

The row vectors $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ specify the new X, Y, and Z axis of the new camera frame with respect to world coordinate system. They can be constructed in the following way:

- The new X axis is parallel to the baseline: $\mathbf{r}_1 = (\mathbf{c}_1 - \mathbf{c}_2)/\|\mathbf{c}_1 - \mathbf{c}_2\|$.
- The new Y axis is orthogonal to new X axis and an arbitrary vector $\mathbf{k}$: $\mathbf{r}_2 = \mathbf{k} \wedge \mathbf{r}_1$.
- The new Z axis is orthogonal to new XY plane: $\mathbf{r}_3 = \mathbf{r}_1 \wedge \mathbf{r}_2$.

Usually, we set arbitrary vector $\mathbf{k}$ as old left camera's Z axis.

Then the new PPMs for both virtual cameras can be written as:

$$\mathbf{P}_{ln} = \mathbf{C}\left[\mathbf{R}_n \quad -\mathbf{R}_n\mathbf{c}_1\right] \tag{5.10}$$

$$\mathbf{P}_{rn} = \mathbf{C}\left[\mathbf{R}_n \quad -\mathbf{R}_n\mathbf{c}_2\right] \tag{5.11}$$

The transformation matrices mapping the old image coordinates to rectified image coordinates are

$$\mathbf{T}_l = \mathbf{C}\mathbf{R}_n\mathbf{R}_l^{-1}\mathbf{C}^{-1} \tag{5.12}$$

$$\mathbf{T}_r = \mathbf{C}\mathbf{R}_n\mathbf{R}_r^{-1}\mathbf{C}^{-1} \tag{5.13}$$

5.3.2 Image Warping

If we directly map the old image coordinates to the new ones, usually we get non-integer result, which would be difficult to construct a new image. Instead, we map the integer pixel position in the rectified image back to the original image at sub-pixel position, and obtain the color information through bilinear interpolation.

Radial lens distortion is also present in the original stereo image. Before proceeding to any other step, lens distortion effects have to be removed from the original image, which also requires image warping. Actually, lens distortion removal and rectification can be combined together to speed up the process. We can formulate an overall transformation equation which projects the distortion free rectified coordinates to original image coordinates. In this way, bilinear interpolation only needs to be performed once.

We denote $\mathbf{u}_n = (u_n, v_n)$ as rectified image coordinates and $\hat{\mathbf{u}} = (\hat{u}, \hat{v})$ as actual observed image coordinates. Combining Eq. (5.4) and Eq. (5.13), we can derive that the transformation from $\mathbf{u}_n$ to $\hat{\mathbf{u}}$ follows:

$$w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{R}_j \mathbf{R}_n^{-1} \mathbf{C}^{-1} \begin{bmatrix} u_n \\ v_n \\ 1 \end{bmatrix} \tag{5.14}$$

and

$$\hat{x} = x\big[1 + k_1(x^2 + y^2) + k_2(x^2 + y^2)^2\big] \tag{5.15}$$

$$\hat{y} = y\big[1 + k_1(x^2 + y^2) + k_2(x^2 + y^2)^2\big] \tag{5.16}$$

and finally

$$\begin{bmatrix} \hat{u} \\ \hat{v} \\ 1 \end{bmatrix} = \mathbf{C} \begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix} \tag{5.17}$$

where k_1 and k_2 are distortion coefficients computed from calibration.

Figure 5.8 is the distortion free rectified pair of Figure 5.7. We use a planar checkerboard pattern for easy illustration. After image warping, we can see that the corresponding points (such as square corners) lie on the same horizontal line in both images.

5.3.3 Matching

After rectification, the next step is to identify the pair of corresponding points in two images. From epipolar geometry, we can derive that they lie on the same horizontal line in rectified images. This constraint largely reduces the computation

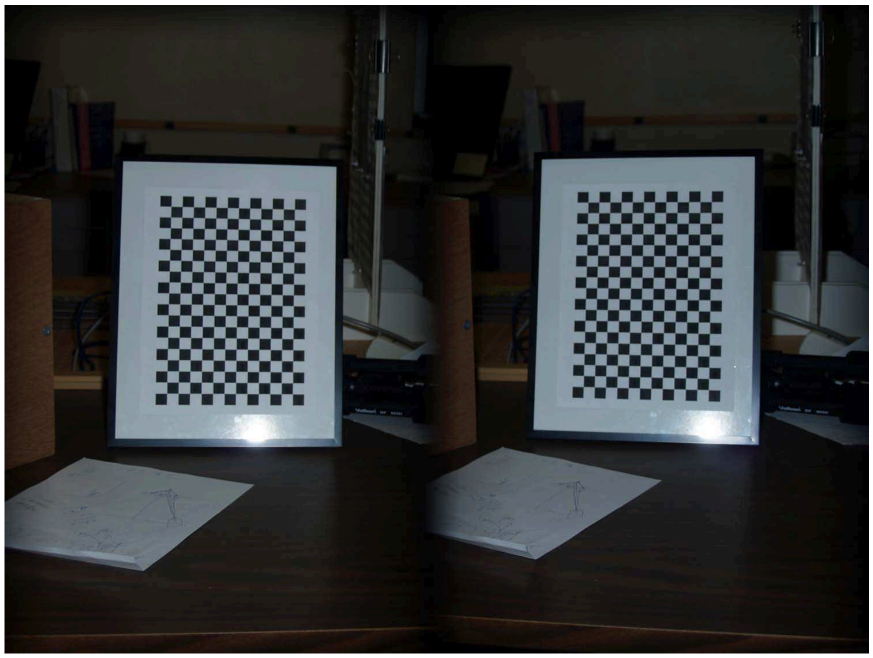

Figure 5.7: Original Stereo Image

complexity. For an arbitrary point (u_l, v_l) in left rectified image, its corresponding point (u_r, v_l) in right rectified image is defined as

$$u_r = \arg\max_u \frac{\sum_{i=-N}^{N}\sum_{j=-N}^{N} I_l(u_l+i, v_l+j) I_r(u+i, v_l+j)}{\sqrt{\sum_{i=-N}^{N}\sum_{j=-N}^{N} I_l(u_l+i, v_l+j)}\sqrt{\sum_{i=-N}^{N}\sum_{j=-N}^{N} I_r(u+i, v_l+j)}} \tag{5.18}$$

which means that the centers of two small image patch with largest cross-correlation are marked as corresponding points.

Exhaustive search for matching image patch along the whole scan-line in high resolution image is time consuming. To improve the speed, we adopt a multi-resolution search scheme, where we form a group of images with different resolution from original image and the search is conducted from coarse level to fine level. Figure 5.9 shows an example of Gaussian pyramid, where the original image I is sub-sampled

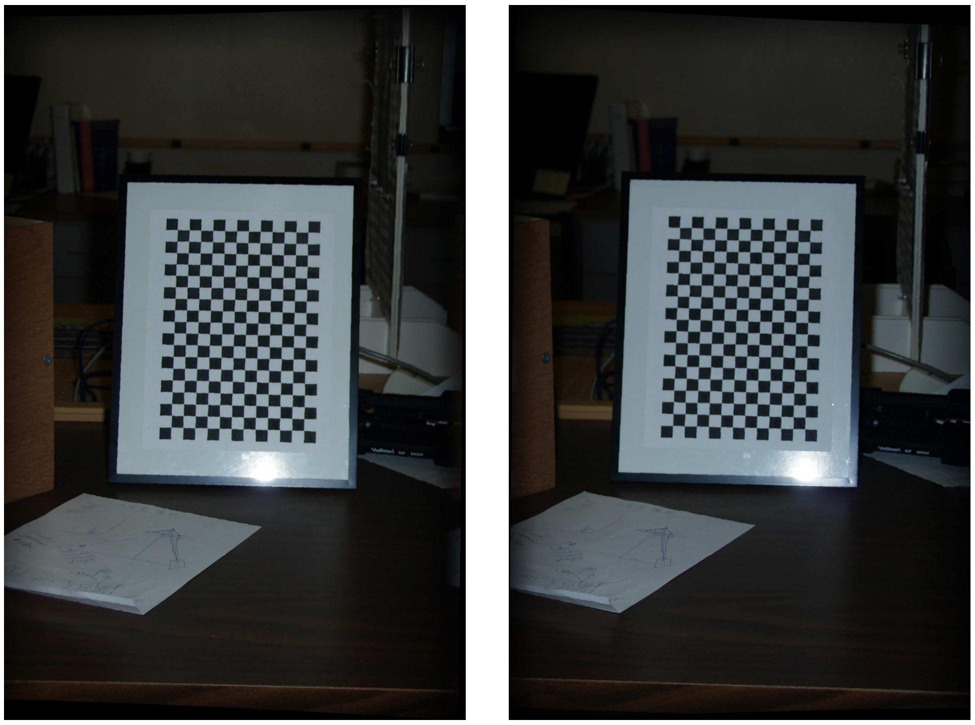

Figure 5.8: Distortion-free Rectified Image Pairs

to half resolution several times. Each level in Gaussian pyramid is calculated as

$$g_0(i,j) = I(i,j) \tag{5.19}$$

$$g_k(i,j) = \sum_{m=-2}^{2}\sum_{n=-2}^{2} w(m,n)g_{(k-1)}(2i+m, 2j+n) \tag{5.20}$$

where $w(m,n)$ is a 5×5 size Gaussian kernel.

In our application, we adopt a three level gaussian pyramid spanning from g_0 to g_2. For any point $\mathbf{u}_{l0}(u_{l0}, v_{l0})$ in original left rectified image g_{l0}, its corresponding point search is conducted in the following way:

1. Identify $\mathbf{u}_{l0}$'s projections in g_{l1} and g_{l2} using Eq. (5.20)

2. For $\mathbf{u}_{l2}(u_{l2}, v_{l2})$, search along the scan-line v_{l2} in g_{r2} within range $[u_s, u_b]$ for

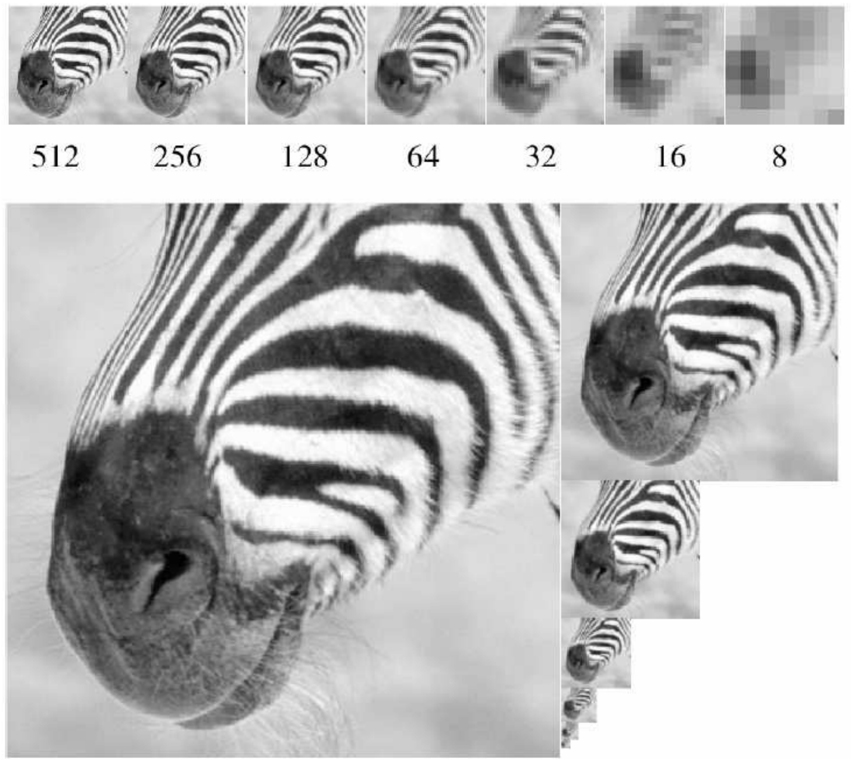

Figure 5.9: Example of Gaussian Pyramid

optimal match $\mathbf{u}_{r2}(u_{r2}, v_{r2})$

3. $\mathbf{u}_{l1}(u_{l1}, v_{l1})$'s correspondence $\mathbf{u}_{r1}(u_{r1}, v_{r1})$ is searched within $[2u_{r2}-1, 2u_{r2}+1]$ on horizontal line v_{r1} in g_{1r}

4. The final search is performed between $[2u_{r1}-1, 2u_{r1}+1]$ in original right rectified image g_{2r} to identify $\mathbf{u}_{r0}(u_{r0}, v_{r0})$

In practice, we also search along the vertical direction from v_r-1 to v_r+1 as rectified images may not be perfectly aligned due to errors from calibration.

5.3.4 Triangulation

After identifying corresponding points $\mathbf{u}_l(u_l, v_l)$ and $\mathbf{u}_r(u_r, v_r)$, we can recover the object point's position in 3D space. Its 3D coordinates with respect to left rectified camera frame is denoted as $\tilde{\mathbf{X}} = (\tilde{X}, \tilde{Y}, \tilde{Z})$. From perspective projection geometry, we have

$$w_l \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \mathbf{C}\tilde{\mathbf{X}} \tag{5.21}$$

Since right rectified camera frame is translated along X axis to $(B, 0, 0)^t$ (B is the length of baseline), the perspective projection for right image is:

$$w_r \begin{bmatrix} u_r \\ v_r \\ 1 \end{bmatrix} = \mathbf{C}(\tilde{\mathbf{X}} + \mathbf{t}) \tag{5.22}$$

where $\mathbf{t} = (-B, 0, 0)^t$.

Ideally, v_l, v_r should be the same and $\mathbf{u}_l$, $\mathbf{u}_r$ and $\tilde{\mathbf{X}}$ should form a perfect triangle. In practice, such assumption usually does not hold that two rays $\mathbf{u}_l\mathbf{c}_l$ and $\mathbf{u}_r\mathbf{c}_r$ do not intersect in 3D space. To calculate $\tilde{\mathbf{X}}$, we can rearrange Eq.(5.21) and Eq. (5.22) to form a group of linear equations:

$$\begin{bmatrix} f_u & s & u_0 - u_l \\ 0 & f_v & v_0 - v_l \\ f_u & s & u_0 - u_r \\ 0 & f_v & v_0 - v_r \end{bmatrix} \begin{bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ f_u B \\ 0 \end{bmatrix} \tag{5.23}$$

where f_u and f_v are focal length in pixel unit, s is skew factor, (u_0, v_0) is principle point and B is length of baseline. They are all obtained through calibration described in Chapter 4. The above over-constrained equation can be solved through pseudo-inverse method.

5.4 Experimental Results

Our experiment is carried out on the novel stereo camera system. The camera is connected through USB cable to a host computer, which controls the image capturing and transferring process. Figure 5.4 and Figure 5.5 are sampled stereo pattern and texture images of resolution 3136×2352. They are corrected from lens distortion and rectified using methods described in Section 5.3.1 and Section 5.3.2. Then we generate a three level Gaussian pyramid to perform multi-level search to identify all corresponding point pairs. For background pixels, matching process is time consuming and meaningless. As pattern projector's working range is within certain range and we use a dark background, usually object is much brighter than background (as shown in Figure 5.4) so that it can be segmented through simple thresholding and morphological operations. A binary image can be generated using:

$$M(u,v) = \begin{cases} 1 & I(u,v) \geq T \\ 0 & I(u,v) < T \end{cases} \tag{5.24}$$

Simple thresholding may filter out some object parts with dark colors (like human beard). The initial binary mask M can first be dilated to fill the holes inside and then eroded to remove unreliable boundary pixels. Figure 5.10 shows the left rectified pattern image of Figure 5.4and the segmentation result.

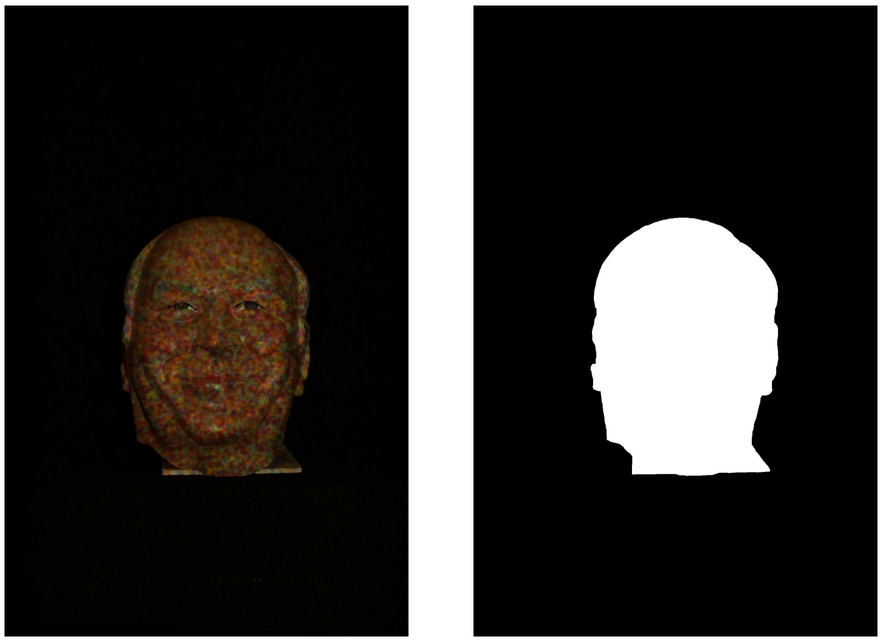

Figure 5.10: Left Rectified Image and Segmentation Result

Then the correspondence search algorithm is only performed on object pixels. We can limit the search range in coarsest level of Gaussian pyramid to further improve the speed. From Eq. (5.21) and Eq. (5.22), we have

$$u_l = \frac{f_u\tilde{X} + s\tilde{Y} + u_0\tilde{Z}}{\tilde{Z}} \tag{5.25}$$

$$u_r = \frac{f_u(\tilde{X} - B) + s\tilde{Y} + u_0\tilde{Z}}{\tilde{Z}} \tag{5.26}$$

which means

$$u_r = u_l - \frac{f_u B}{\tilde{Z}} \tag{5.27}$$

In our experiment setting, the object is of limited size (usually human face model) and placed in the working range of pattern projector (from 1.2 m to 1.8 m), then $\tilde{Z}$

should be between $[\tilde{Z}_s, \tilde{Z}_b]$ (where in implementation, we can set $\tilde{Z}_s$ as 1.2 and $\tilde{Z}_b$ as 1.8). So for point (u_l, v_l), the search window $[u_s, u_b]$ in the second level Gaussian pyramid g_{2r} can be set as

$$u_s = \frac{u_l}{4} - \frac{f_u B}{4\tilde{Z}_b} \tag{5.28}$$

$$u_b = \frac{u_l}{4} - \frac{f_u B}{4\tilde{Z}_s} \tag{5.29}$$

The point cloud generated from stereo image Figure 5.4 is shown in Figure 5.11

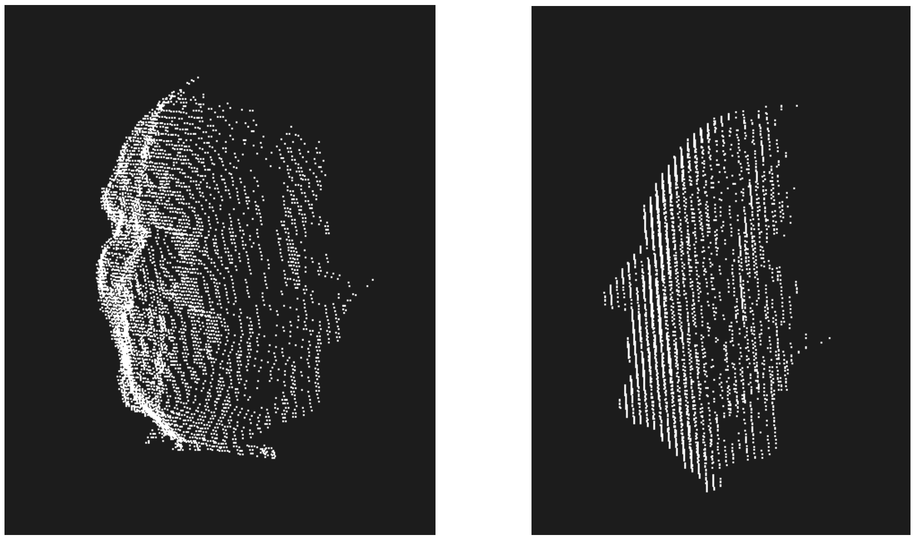

Figure 5.11: Point Cloud of Sample Object

5.5 Summary

In this Chapter, we introduced the basic theory of stereo vision and presented our novel stereo camera system. We combine lens distortion correction and rectification in a single process so that image warping has to be conducted only once. We adopt a multi-level resolution search scheme and limit the search range derived from practical system design to improve the speed. We first segment the object from background and then perform our fast correspondence search algorithm to recover its depth information. The generated point cloud illustrates that our method is fast and reliable.

Chapter 6

Registration

In Chapter 5, we presented our novel stereo camera. Due to limited viewing field, a single stereo camera can reconstruct only a partial shape of an object. To recover the object's full shape, we have built a multi-stereo camera system. It consists of three stereo cameras with one placed in front of the object and two at the sides about 45^0 apart from the center one (as shown in Figure 6.1).

The partial shape reconstructed by each stereo camera is expressed in camera's own coordinate system. To combine three partial shapes together, the first step is to bring them into a common coordinate system, which is termed as *Registration*. For convenience, we choose the center camera's frame as the reference coordinate system.

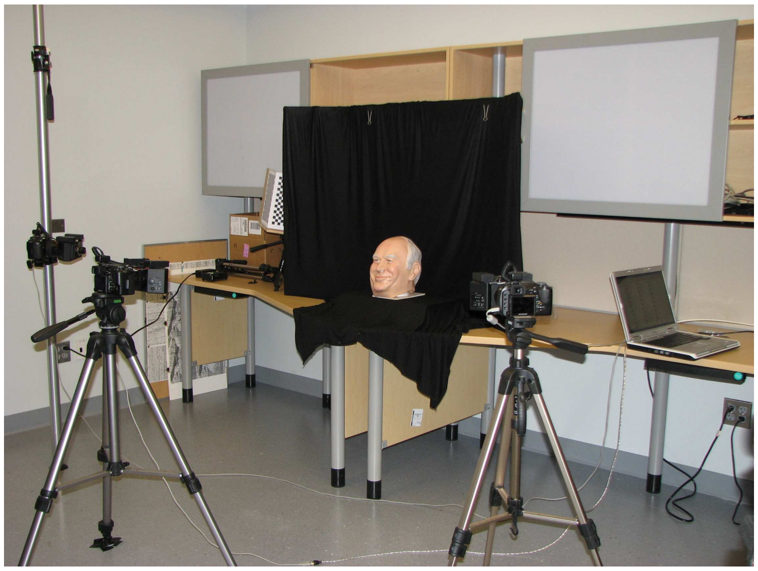

Figure 6.1: Multi Stereo Camera System

6.1 Initial Registration

The transformation between different camera coordinate systems can be described by a rotation matrix and a translation vector:

$$\mathbf{X}_r = \mathbf{R}\mathbf{X}_o + \mathbf{t} \tag{6.1}$$

where $\mathbf{X}_o = (X_o, Y_o, Z_o)$ is object point's 3D coordinates in its original camera frame, $\mathbf{R}$ is a 3×3 rotation matrix, $\mathbf{t}$ is a 3×1 translation vector, and $\mathbf{X}_r = (X_r, Y_r, Z_r)$ is the new coordinate in the common reference coordinate system. They can also be written in homogeneous coordinates form:

$$\begin{bmatrix} X_r \\ Y_r \\ Z_r \\ W_r \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X_o \\ Y_o \\ Z_o \\ W_o \end{bmatrix} \tag{6.2}$$

where W_r and W_o are homogeneous scalars, and $\mathbf{0}$ is a row vector that is $\mathbf{0} = [0,\ 0,\ 0]^t$.

If the calibration object is placed facing two cameras so that both cameras can get good views (as shown in Fig.), then the transformation between two cameras and the common world coordinate system (set according to calibration object) can be retrieved using methods described in Chapter 4.

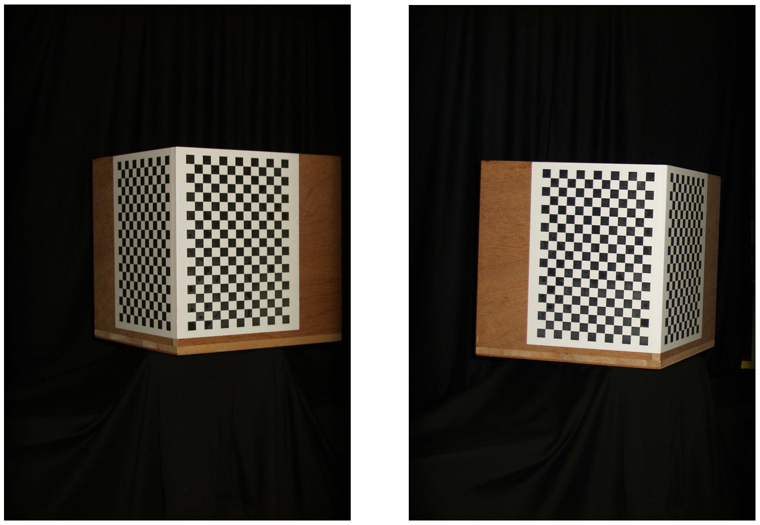

Figure 6.2: Images of Calibration Object taken by Left and Center Cameras

If we take left camera and center camera as example, we have

$$\begin{bmatrix} X_l \\ Y_l \\ Z_l \\ W_l \end{bmatrix} = \begin{bmatrix} \mathbf{R}_l & \mathbf{t}_l \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ W_w \end{bmatrix} \tag{6.3}$$

and

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ W_c \end{bmatrix} = \begin{bmatrix} \mathbf{R}_c & \mathbf{t}_c \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ W_w \end{bmatrix} \tag{6.4}$$

We can easily derive the transformation between left camera and center camera as:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ W_c \end{bmatrix} = \begin{bmatrix} \mathbf{R}_c & \mathbf{t}_c \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_l & \mathbf{t}_l \\ \mathbf{0} & 1 \end{bmatrix}^{-1} \begin{bmatrix} X_l \\ Y_l \\ Z_l \\ W_l \end{bmatrix} \tag{6.5}$$

which can also be written as

$$\mathbf{X}_c = \mathbf{R}_{lc}\mathbf{X}_l + \mathbf{t}_{lc} \tag{6.6}$$

where

$$\mathbf{R}_{lc} = \mathbf{R}_c{\mathbf{R}_l}^{-1} \tag{6.7}$$

and

$$\mathbf{t}_{lc} = -\mathbf{R}_c{\mathbf{R}_l}^{-1}\mathbf{t}_l + \mathbf{t}_c \tag{6.8}$$

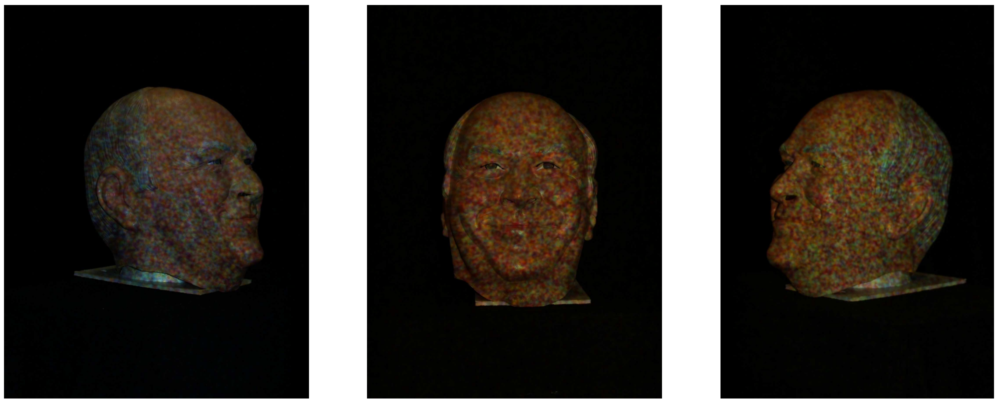

Figure 6.3: Left Rectified Texture Images taken by Three Cameras

Similar procedures can be applied to right stereo camera and center camera, so that we can also obtain the transformation matrix $\mathbf{R}_{rc}$ and $\mathbf{t}_{rc}$. Applying Eq. (6.3) to left point cloud and right point cloud, we can roughly align three partial shapes in one coordinate system. Figure 6.3 shows the left rectified texture image captured by different cameras, and Figure 6.4 shows the initial registration result.

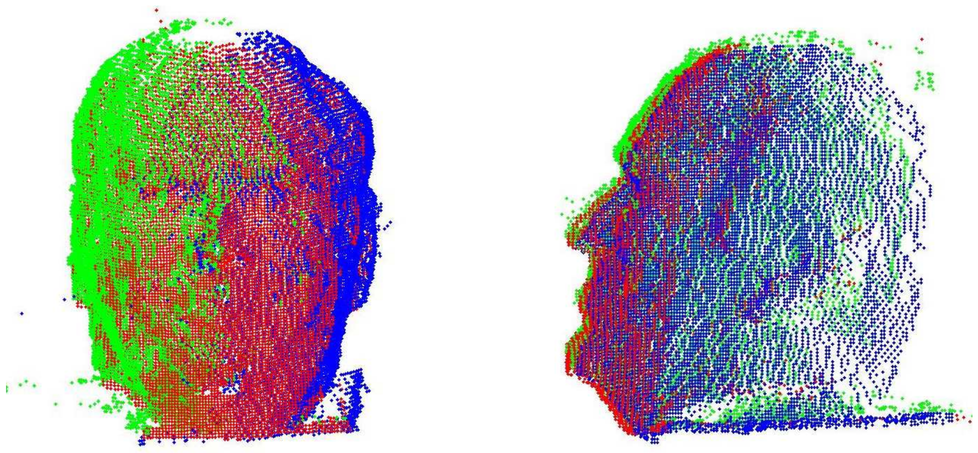

Figure 6.4: 3 Point Clouds after Initial Registration

6.2 Registration Refinement

In Figure 6.4, we can easily observe that point clouds do not align with each other perfectly. The transformation matrices $\mathbf{R}_{lc}$, $\mathbf{t}_{lc}$, $\mathbf{R}_{rc}$ and $\mathbf{t}_{rc}$ calculated from calibration result need to be further refined to minimize the alignment errors.

6.2.1 Corresponding Point Registration

Assume that the point set $\mathbf{X}_{oi}$ in left (or right) point cloud correspond to point set $\mathbf{X}_{ri}$ in center view, then the optimal transformation matrices $\mathbf{R}$ and $\mathbf{t}$ are the ones that minimize the following objective function:

$$F(\mathbf{R},\mathbf{t}) = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{X}_{ri} - \mathbf{R}\mathbf{X}_{oi} - \mathbf{t}||^2 \tag{6.9}$$

The closed form solution of $\mathbf{R}$ and $\mathbf{t}$ that minimizes F can be derived as follows. Let $\mathbf{U}_r$ and $\mathbf{U}_o$ be center of mass for each point set:

$$\mathbf{U}_o = \frac{1}{N}\sum_{i=1}^{N} \mathbf{U}_{oi} \tag{6.10}$$

$$\mathbf{U}_r = \frac{1}{N}\sum_{i=1}^{N} \mathbf{U}_{ri} \tag{6.11}$$

The cross-covariance matrix Σ_{or} of point set $\mathbf{X}_{oi}$ and $\mathbf{X}_{ri}$ is defined as

$$\begin{aligned}\Sigma_{or} =& \frac{1}{N}\sum_{i=1}^{N} (\mathbf{X}_{oi} - \mathbf{U}_o)(\mathbf{X}_{ri} - \mathbf{U}_r)^t \\ =& \frac{1}{N}\sum_{i=1}^{N} \mathbf{X}_{oi}\mathbf{X}_{ri}^t - \mathbf{U}_o\mathbf{U}_r^t\end{aligned} \tag{6.12}$$

An anti-symmetric matrix $\mathbf{A}$ is defined as

$$\mathbf{A} = \Sigma_{or} - \Sigma_{or}^t \tag{6.13}$$

then a symmetric 4×4 matrix $\mathbf{Q}$ can be formed

$$\mathbf{Q} = \begin{bmatrix} trace(\Sigma_{or}) & \Delta^t \\ \Delta & \Sigma_{or} + \Sigma_{or}^t - trace(\Sigma_{or})\mathbf{I}_3 \end{bmatrix} \tag{6.14}$$

where $\Delta^t = (A_{23}, A_{31}, A_{12}^t)$ and $\mathbf{I}_3$ is a 3×3 identity matrix. The unit eigenvector $\mathbf{q} = (q_0, q_1, q_2, q_3)$ corresponding to the maximum eigenvalue of matrix $\mathbf{Q}$ is selected as the quaternion to generate optimal rotation matrix $\mathbf{R}$, where

$$\mathbf{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix} \tag{6.15}$$

and the optimal translation vector $\mathbf{t}$ is given by

$$\mathbf{t} = \mathbf{U}_r - \mathbf{R}\mathbf{U}_o \tag{6.16}$$

The detailed proof of the optimality of the above closed form solution can be found in [76].

6.2.2 Iterative Closest Point Algorithm

Based on the closed-form solution of optimal transformation matrices derived in Section 6.2.1 (see Eq. (6.15) and Eq. (6.16)), the Iterative Closest Point Algorithm (ICP) registration refinement is stated as the following:

1. Bring left (or right) point cloud $\mathbf{X}_o$ to center camera's frame using the initial transformation matrices $\mathbf{R}_{or0}$ and $\mathbf{t}_{or0}$ calculated from calibration result. The new coordinates of $\mathbf{X}_o$ is denoted as $\mathbf{X}_{oj}$, where $j = 0$ in this initialization step.

2. For each point in $\mathbf{X}_{oj}$, find its closest point in center point cloud where Euler distance d_j is used for error measure. Then we establish the correspondence between two point sets $\mathbf{X}_{oj}$ and $\mathbf{X}_r$.

3. Compute the optimal registration matrices $\mathbf{R}_{orj}$ and $\mathbf{t}_{orj}$ between $\mathbf{X}_{oj}$ and $\mathbf{X}_r$ using methods described in Section 6.2.1.

4. Apply the registration: $\mathbf{X}_{o(j+1)} = \mathbf{R}_{orj}\mathbf{X}_{oj} + \mathbf{t}_{orj}$.

5. Terminate the iteration if the change of mean error distance between two point sets $\mathbf{X}_{o(j+1)}$ and $\mathbf{X}_r$ falls below a preset threshold τ (i.e., $d_j - d_{j+1} < \tau$), otherwise go back to step 2 to continue.

The convergence of the above algorithm is elaborately proved in [69].

6.2.3 Iterative Closest Surface Patch Approach

The ICP algorithm described above has obvious disadvantages when applied to our camera system. With the wide angle system setting, two point clouds to be registered do not completely overlap with each other and therefore correspondences only exist among a part of them. Exhaustive search for correspondence for every point is not feasible and extremely expensive. Instead, we propose an Iterative Closest Surface Patch (ICSP) approach to accelerate the process.

Instead of establishing correspondences between every pair of points, we try to find the matching surface patches in overlapping parts. From epipolar geometry, we know that approximately the left half of right point cloud overlaps with the center one (similar derivation applies to left point cloud). The surface patches are selected in equal spacing from the overlapping area on depth map. Figure 6.5 illustrates the surface patches to be registered in right point cloud. As point cloud can also

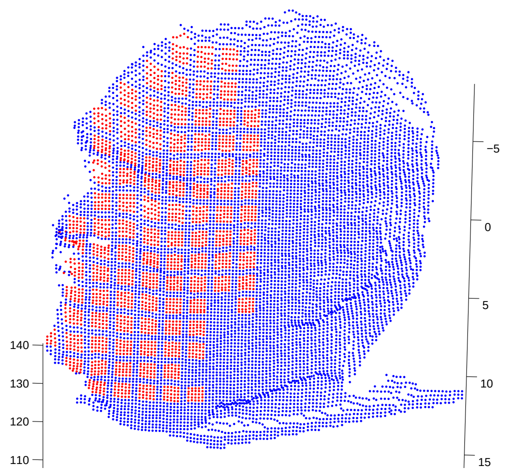

Figure 6.5: Surface Patches in Right Point Cloud

be viewed as the depth map of a down-sampled image, each surface patch can be indexed by its corresponding 2D coordinates. $\mathbf{SP}_{u,v}$ denotes the surface patch whose center is projected onto (u, v) in 2D image and $\mathbf{X}_{u,v}$ denotes the 3D coordinate of object point projected onto (u, v). For a surface patch $\mathbf{SP}_{u_r,v_r}$ in right point cloud, its corresponding patch in center point cloud is defined as

$$\mathbf{SP}_{u_c,v_c} = \underset{(u,v)\in[u_0-W,u_0+W]\times[v_0-W,v_0+W]}{\arg\min} d(u,v) \tag{6.17}$$

where

$$d(u,v) = \frac{1}{(2N+1)^2}\sum_{i=-N}^{N}\sum_{j=-N}^{N}||\mathbf{X}_{u_l+i,v_l+i} - \mathbf{X}_{u+i,v+i}|| \tag{6.18}$$

where $(2N+1)\times(2N+1)$ is the patch size, $[u_0-W, u_0+W]\times[v_0-W, v_0+W]$ is a preset search range, and the center of search window (u_0, v_0) can be calculated through the spatial geometrical relationship. Assume that the current estimated transformation matrices between two cameras are $\mathbf{R}$ and $\mathbf{t}$, then projection of $\mathbf{X}_{u_r,v_r}$ in center image is at:

$$\mathbf{u}_0 = \mathbf{C}_c[\mathbf{R}\,\mathbf{t}]\mathbf{X}_{u_r,v_r} \tag{6.19}$$

where $\mathbf{C}_c$ is the intrinsic matrix of center camera, $\mathbf{u}_0 = (u_{0w}, v_{0w}, w)$ is homogeneous image coordinates, and

$$u_0 = u_{0w}/w \tag{6.20}$$

$$v_0 = v_{0w}/w \tag{6.21}$$

Though we have preset the overlapping areas to select surface patches, the ones around borders may fail to find its correct correspondences. Such mismatch would prevent the iterative process from converging to the correct solution. An outlier removal procedure is needed when establishing corresponding point sets. If the mean error distance d between two surface patches are larger than certain threshold τ_d, both of them have to be removed from the list. The rest of our approach remains the same with ICP algorithm.

Our surface patch matching strategy is similar to the image patch matching in stereopsis. The inherent ordering constraint within the small patch helps to capture more local geometrical features, which is more reliable than measuring error distance between pair of points.

6.3 Experimental Results

We apply our ICSP approach on the left and right point clouds generated by stereo camera to align them into the center camera's frame. The coarse registration result is shown in Figure 6.4. The refined result is shown in Figure 6.6. A close-up

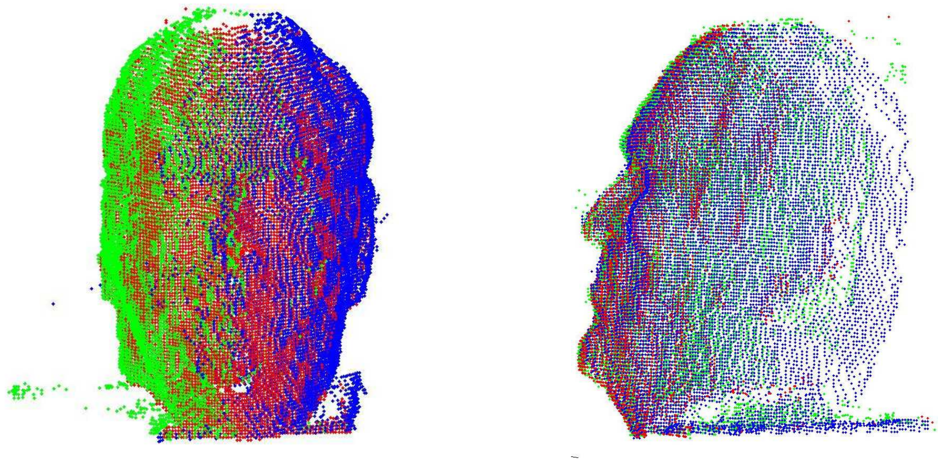

Figure 6.6: Registration Refinement Result

view of right point cloud and center point cloud is shown in Figure 6.7. From Figure 6.6 and Figure 6.7, we can clearly see that the point clouds are almost perfectly aligned together after the refinement step.

Another way to check the accuracy is to project the center of matched surface

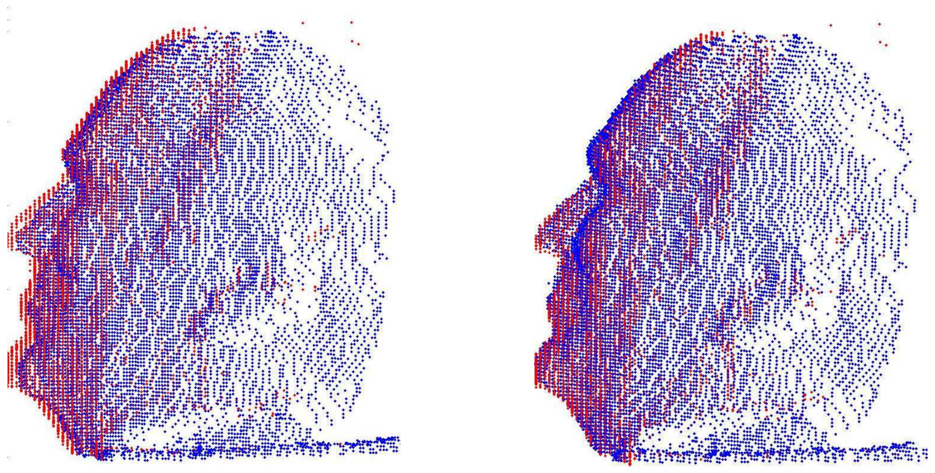
Figure 6.7: Alignment of Right and Center Point Clouds before and after Refinement

patches (which we term as control points) to their corresponding texture images. If perfectly registered, corresponding control points should lie on the same texture feature points. Figure 6.8 shows the positions of these control points in rectified texture images.

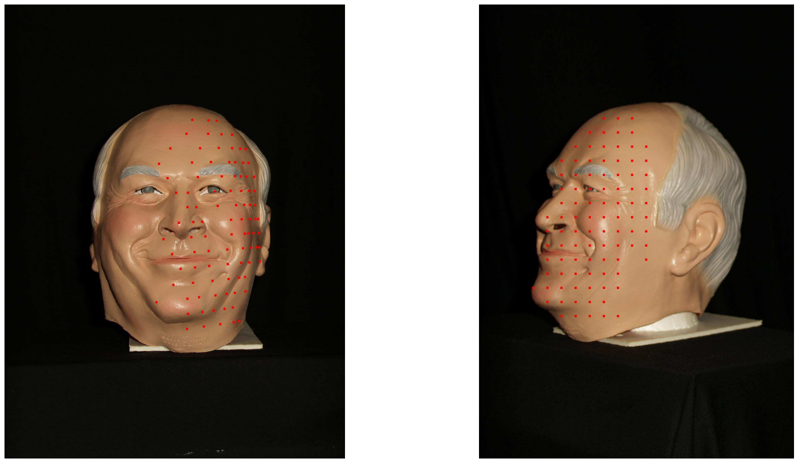
Figure 6.8: Corresponding Control Points in Texture Images

The convergence speed of ICSP is displayed in Figure 6.9 and Figure 6.10, where the mean error distance quickly drops after several iterations.

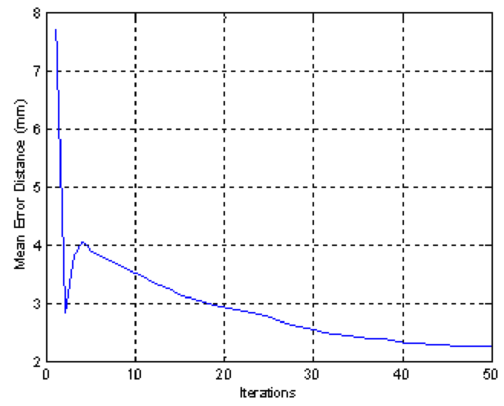

Figure 6.9: Convergence of Mean Error Distance between Left and Center Point Clouds

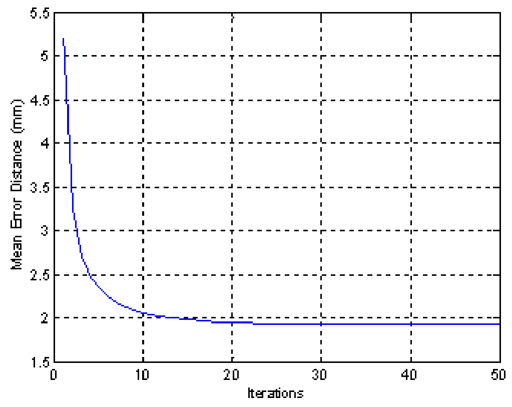

Figure 6.10: Convergence of Mean Error Distance between Right and Center Point Clouds

Experimental results on other data sets are displayed in Figure 6.11 - Figure 6.22.

6.4 Summary

In this Chapter, we introduced our multi-camera system which consists of three stereo cameras. Each camera can reconstruct partial shape of the object from different viewing directions. We also presented our registration method that aligns three point clouds into one common coordinate system. For convenience, we take the center rectified camera frame as reference. The initial coarse registration is performed based on the calibration results. Then we refine the transformation matrices through our Iterative Closest Surface Patch (ICSP) approach, where small surface patches are selected in left (and right) point clouds and their correspondences in center cloud are identified. The optimal transformation matrices are calculated from two point sets by minimizing a least squares error measure. Such procedures are carried out iteratively until the error measure converges. The experimental results illustrate that our method is fast and robust.

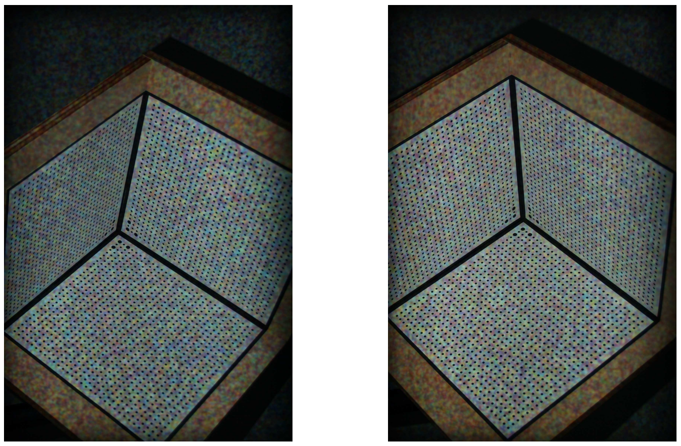

Figure 6.11: Pattern Images from Left and Center Camera

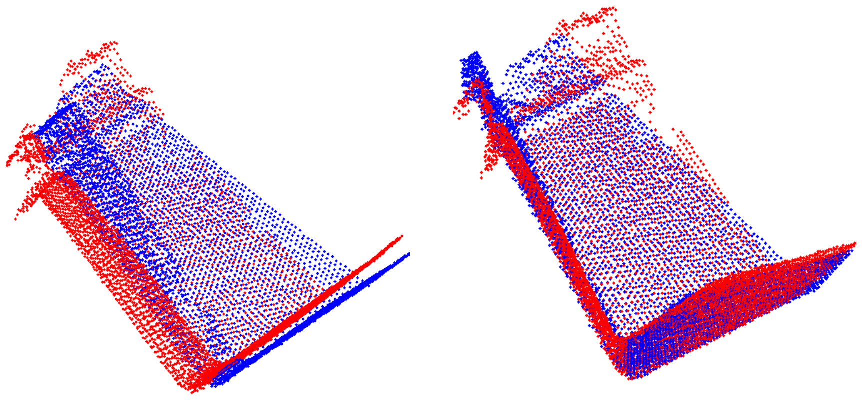

Figure 6.12: Point Clouds Alignment Before and After Registration Refinement

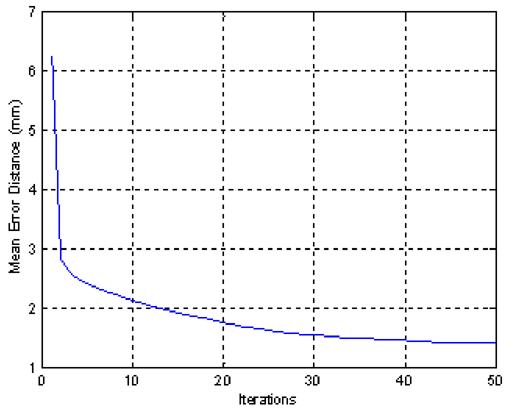

Figure 6.13: Convergence of Mean Error Distance between Two Point Clouds

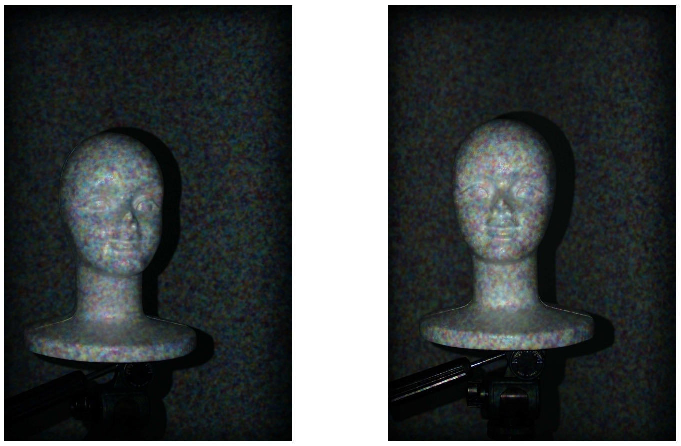

Figure 6.14: Pattern Images from Left and Center Camera

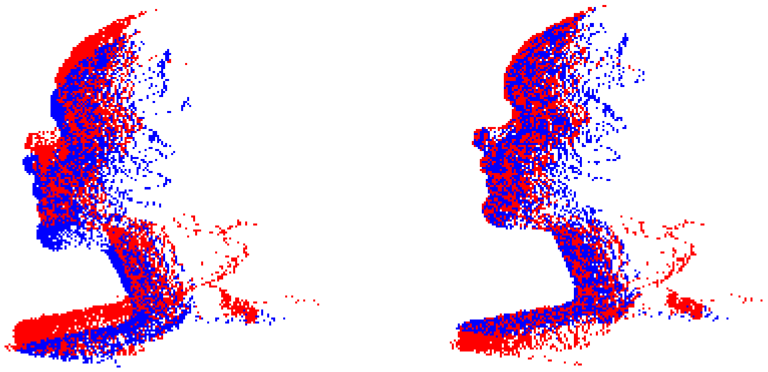

Figure 6.15: Point Clouds Alignment Before and After Registration Refinement

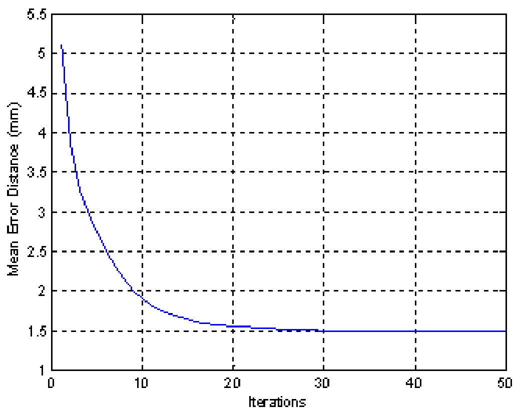

Figure 6.16: Convergence of Mean Error Distance between Two Point Clouds

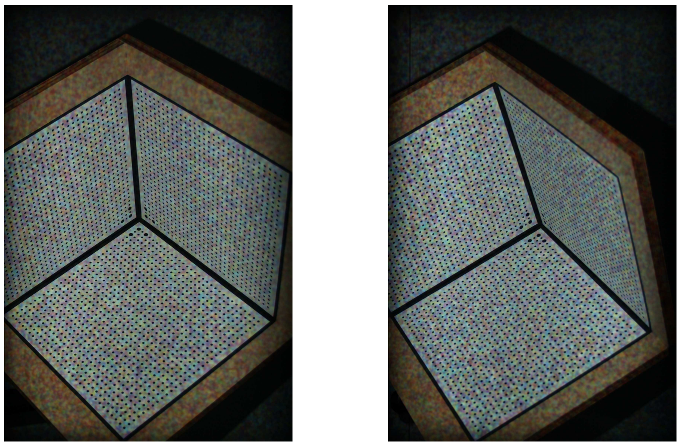

Figure 6.17: Pattern Images from Left and Center Camera

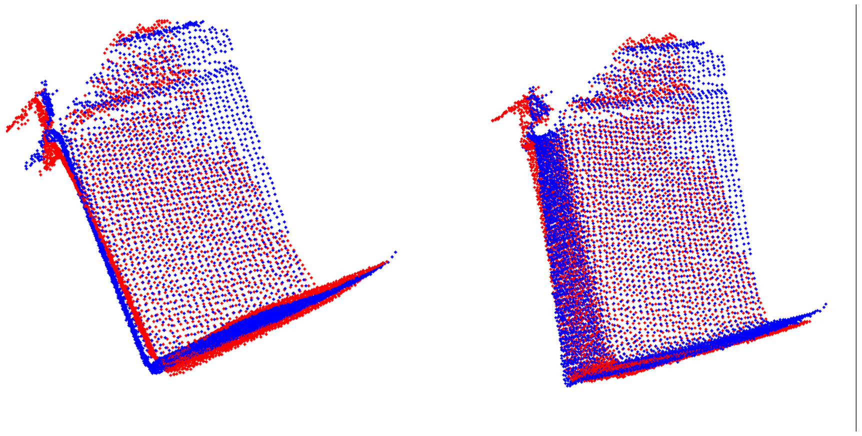

Figure 6.18: Point Clouds Alignment Before and After Registration Refinement

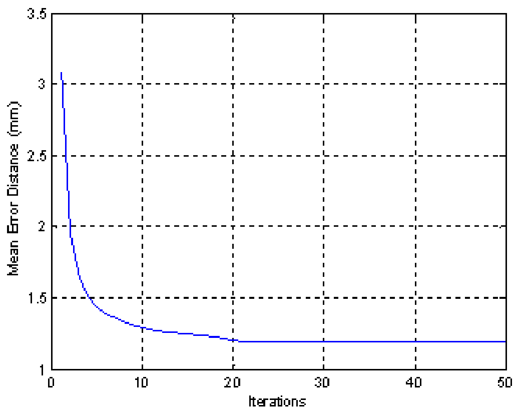

Figure 6.19: Convergence of Mean Error Distance between Two Point Clouds

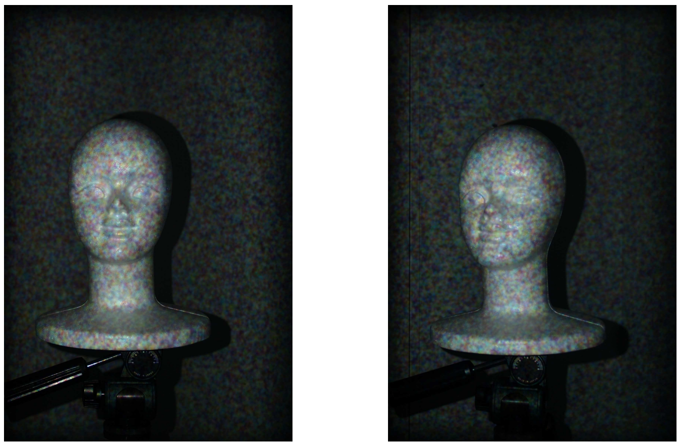
Figure 6.20: Pattern Images from Left and Center Camera

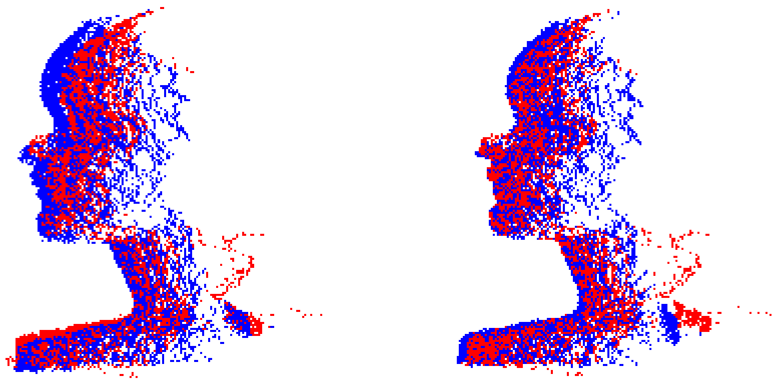
Figure 6.21: Point Clouds Alignment Before and After Registration Refinement

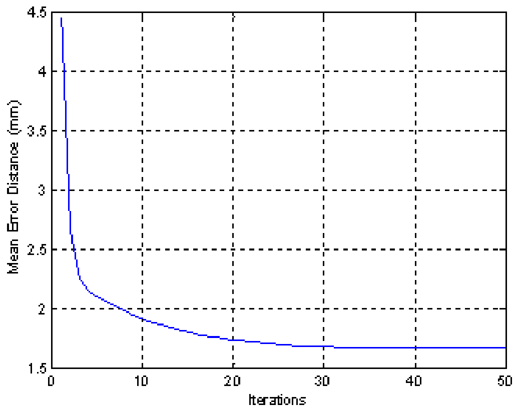

Figure 6.22: Convergence of Mean Error Distance between Two Point Clouds

Chapter 7

3D Modeling

In Chapter 6, we presented an Iterative Closest Surface Patch (ICSP) approach to register three partial shapes of the object into one common coordinate system. The next step is to build a single 3D model from point clouds. To properly render a 3D object on a computer screen, we need a suitable mathematical representation of its surface information. Usually the geometrical shape is parameterized as a set of triangles (termed as triangle mesh) and then the textured appearance is projected onto the wire-frame. 3D modeling task mainly involves iso-surface extraction, mesh generation and texture mapping.

7.1 Surface Extraction

Though registration step minimizes the error distance between overlapping parts of point clouds, there will still be slight misalignment. We adopt a volumetric method to extract the object surface from overlapped point clouds [79].

First, we triangulate each point cloud to obtain a surface mesh. As described in

Chapter 6, a point cloud can be indexed by its corresponding image coordinates. For any pixel $\mathbf{p}(u, v)$ on a 2D grid, it either corresponds to a 3D point $\mathbf{X}_{u,v}$ or is marked as an invalid background pixel. The tessellation starts from the upper left corner of the image. Each time we look at four points (u, v), $(u, v+1)$, $(u+1, v)$ and $(u+1, v+1)$. If two or more points are marked as invalid, then no triangulation will be performed. There exist 6 possible ways of tessellation for the remaining cases (see Figure 7.1). If all four points are valid, the configuration with the shorter diagonal (distance between $\mathbf{X}_{u,v}\mathbf{X}_{u+1,v+1}$ or $\mathbf{X}_{u,v+1}\mathbf{X}_{u+1,v}$ in 3D space) gets selected. The tessellation result of

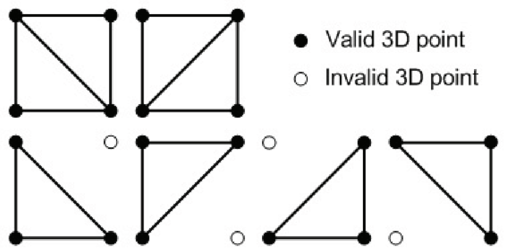

Figure 7.1: Possible Ways of Triangulation

center point cloud is shown in Figure 7.2.

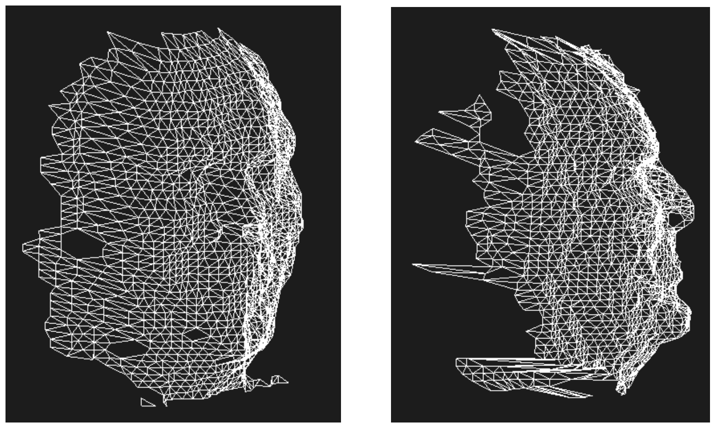

Figure 7.2: Range Surface from Single Point Cloud

Next, we generate a voxel grid covering the whole range of all three range surfaces.

We can calculate a signed distance d_i from every voxel $\mathbf{P}$ to each range surface $\mathbf{S}_i$ (i = 0, 1, 2, denoting the left, center and right range surface). As illustrated in Figure 7.3, we can cast a ray from a camera center $\mathbf{V}_i$ to any voxel $\mathbf{P}$, which would intersect with triangle $\mathbf{P}_{s1}\mathbf{P}_{s2}\mathbf{P}_{s3}$ at $\mathbf{P}_s$, and d_i is defined as

$$d_i = \|\mathbf{P}\mathbf{V}_i\| - \|\mathbf{P}_s\mathbf{V}_i\| \tag{7.1}$$

In this way, all voxels inside the object have positive distances, and those outside have negative distance values. The triangle $\mathbf{P}_{s1}\mathbf{P}_{s2}\mathbf{P}_{s3}$ at $\mathbf{P}_s$ that $||\mathbf{P}\mathbf{V}_i||$ intersects is decided as follows:

First, we project voxel $\mathbf{P}$ to range surface $\mathbf{S}_i$'s corresponding texture image:

$$\lambda\mathbf{p} = \mathbf{C}_i\mathbf{R}_i(\mathbf{P} + \mathbf{t}_i) \tag{7.2}$$

where $\mathbf{R}_i$ and $\mathbf{t}_i$ are rotation matrix and translation vector from reference coordinate system and original camera frame, $\mathbf{C}_i$ is intrinsic camera matrix, λ is an arbitrary scalar and $\mathbf{p} = (u, v, 1)$ is $\mathbf{P}$'s projection. The integer pixels surrounding $\mathbf{p}$ are $(\lfloor u\rfloor, \lfloor v\rfloor)$, $(\lfloor u\rfloor, \lceil v\rceil)$, $(\lceil u\rceil, \lfloor v\rfloor)$ and $(\lceil u\rceil, \lceil v\rceil)$, which can be used to index the corresponding 3D points $\mathbf{X}_{\lfloor u\rfloor,\lfloor v\rfloor}$, $\mathbf{X}_{\lfloor u\rfloor,\lceil v\rceil}$, $\mathbf{X}_{\lceil u\rceil,\lfloor v\rfloor}$ and $\mathbf{X}_{\lceil u\rceil,\lceil v\rceil}$. From the tessellation configuration and $\mathbf{p}$'s relative position to the square, we can easily identify the triangle that ray $\mathbf{P}\mathbf{V}_i$ intersects.

After computing all the signed distance d_i from voxel $\mathbf{P}$ to range surface $\mathbf{S}_i$, we can compute the weighted average:

$$d = \frac{\sum_{i=0}^{2} w_i d_i}{\sum_{i=0}^{2} w_i} \tag{7.3}$$

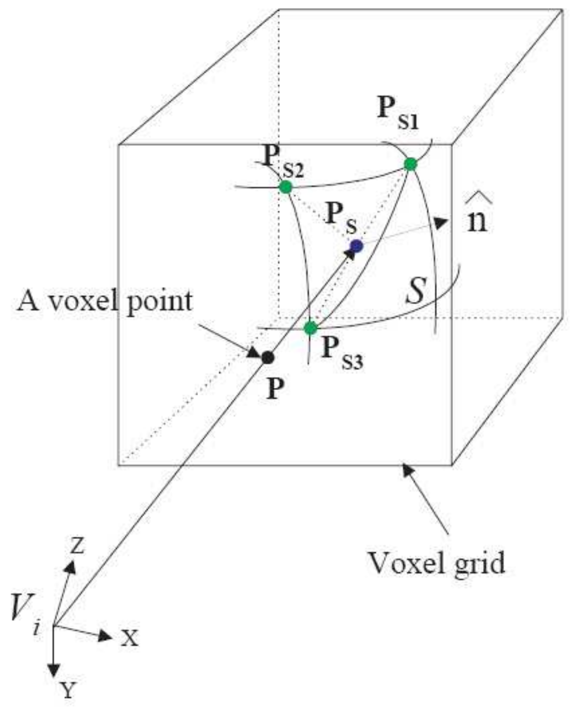

Figure 7.3: Signed Distance Computation

the weight w_i is the angle between viewing direction and surface normal at $\mathbf{P}_s$:

$$w_i = |\mathbf{PV}_i \cdot \hat{n}| \tag{7.4}$$

Naturally, the object surface is generated at the location where the signed distance d is 0.

7.2 Mesh Generation

The Marching Cube algorithm is adopted to generate a surface mesh at zero signed distance. For any cube in the voxel grid (as shown in Figure 7.4), its vertices are either outside of object surface (with negative signed distance) or inside of object surface (with positive signed distance), and the true object surface should intersect the cube edges whose vertices have opposite distance signs.

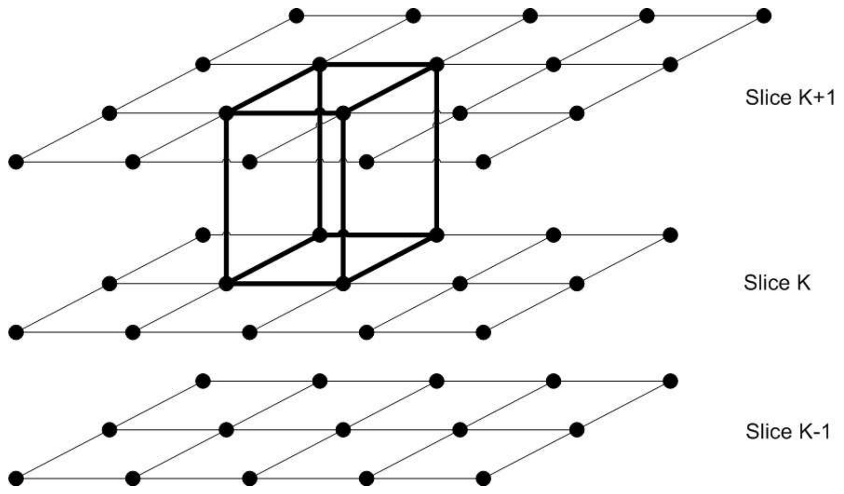

Figure 7.4: Cube of Voxels

Since there are eight vertices in each cube and two states (inside and outside of object surface), there are only $2^8 = 256$ ways a surface can intersect the cube. By enumerating these 256 cases, we can create a table to look up surface-edge intersections given the labeling of cube vertices.

However, enumerating the 256 cases is tedious and error prone. Two different symmetries of the cube reduce the problem from 256 cases to 15 patterns. The first symmetry comes from the vertex sign symmetry, i.e., if the state of all vertices are reversed (from inside the surface to outside or vice versa), the surface intersection remains unchanged. The second symmetry is from cube rotation. Figure 7.5 shows the triangulation of these 15 cases. The simplest pattern, 0, occurs if all vertices are inside (or outside) the surface such that it produces no triangles. Pattern 1 separates one vertex from the other seven and one triangle is generated. Other patterns generate multiple triangles. Permutation of these 14 basic patterns using complementary and rotational symmetry produces the 256 cases.

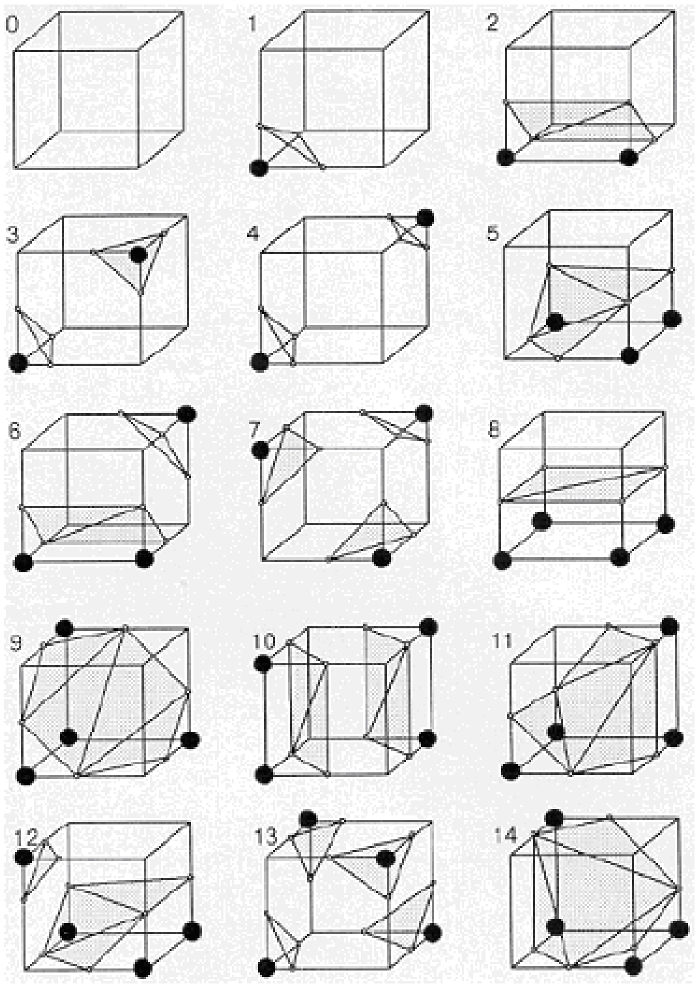

Figure 7.5: Surface Intersection of Voxel Cube

We use a 8 bit char data to index these 256 cases, where each bit indicates the state of one specific vertex (0 if it is outside and 1 if inside). This index serves as a pointer to the edge table and a triangle table. Each entry in the edge table is a 12 bit integer indicating which edge is to be intersected (as shown in Figure 7.6). The triangle table is a 256 × 16 array, where each row vector is the triangle list. Every three entries from the left represent the edge indices of one triangle. The remaining entries are filled with -1. After identifying which edges are to be intersected, we can

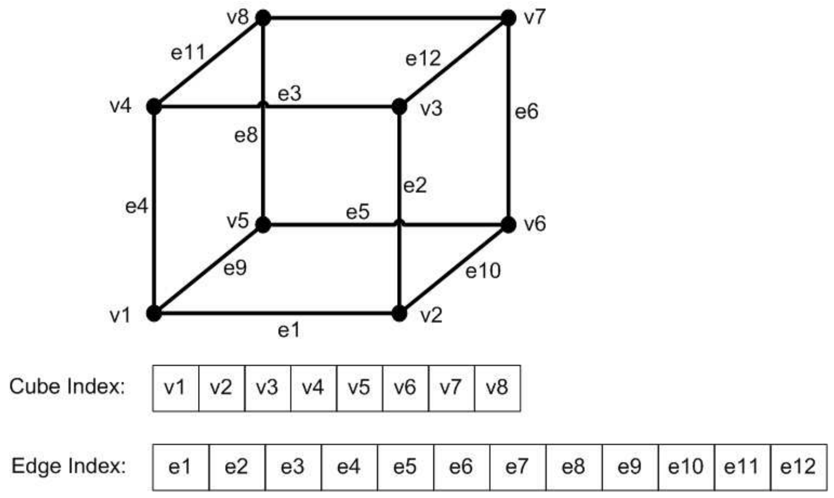

Figure 7.6: Cube Numbering

use interpolation to calculate the position of intersection. If the edge e_i connecting voxels v_m and v_n is to be intersected, and v_m and v_n have signed distances of d_m and d_i, then the interpolated vertex v of surface mesh should be:

$$v = \frac{v_m|d_n| + v_n|d_m|}{|d_m| + |d_n|} \tag{7.5}$$

Such surface-intersection algorithm is carried-out in the whole voxel grid from one cube to another cube to generate a single surface mesh. In our application, we only perform the algorithm in those cubes whose signed distances are below certain threshold value. This would not only improve the speed, but also avoid generating a psuedo-surface at the back of object as our model is not closed. Figure 7.8 shows the surface mesh generated from multiple point clouds (shown in Figure 6.6).

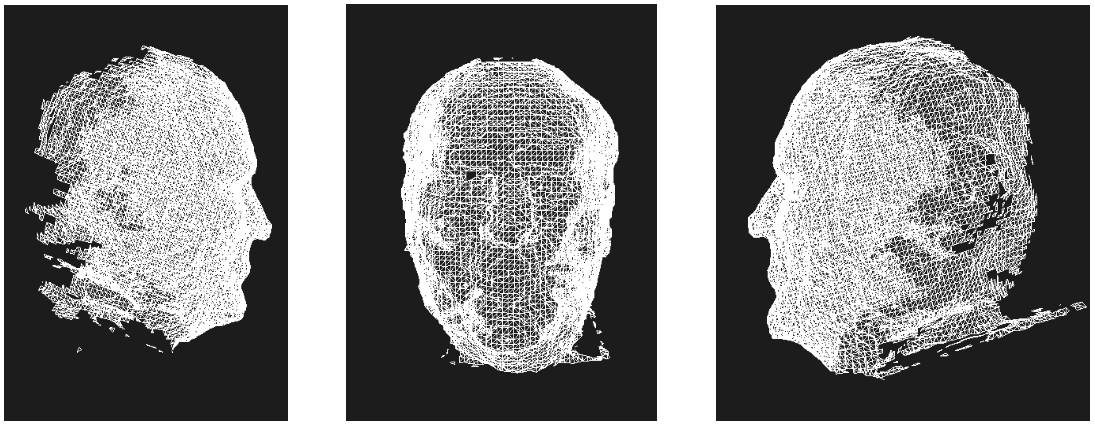

Figure 7.7: Surface Mesh

7.3 Texture Mapping

7.3.1 View Selection

Texture mapping is the last step of 3D modeling. To render a photo-realistic 3D model on computer screen, we wrap the texture images taken by stereo camera (shown in Figure) onto the triangle mesh like gift-wrap paper onto a white box. For

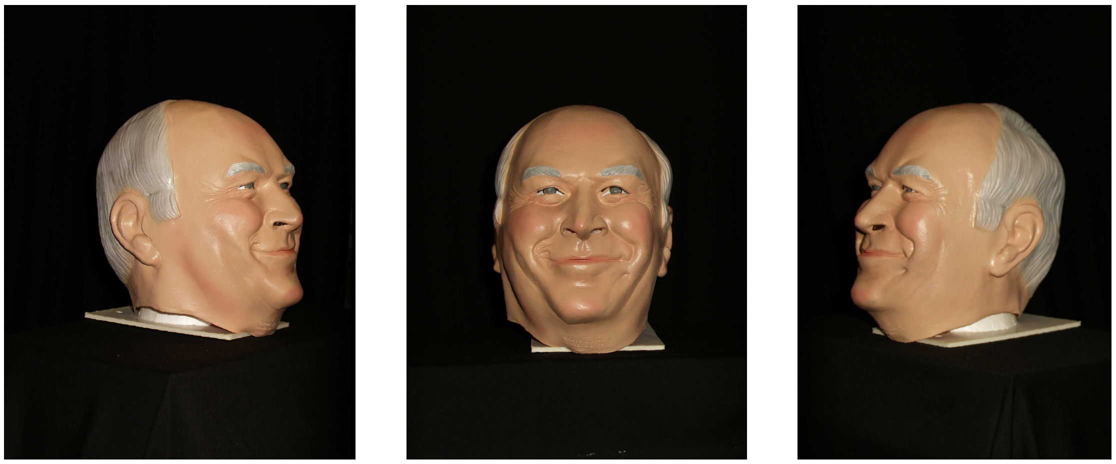

Figure 7.8: Left Rectified Texture Images

every triangle on the mesh, we have to choose one source image among three for texture mapping. Due to occlusion, some triangles on the surface only have one or two valid texture correspondences. To exclude the invalid candidate, we can simply project the vertices back to images and check whether they are within the object area (the segmentation process is described in Chapter 5). Among the valid texture images, we choose the one with best viewing angle.

Suppose $\triangle \mathbf{P}_a\mathbf{P}_b\mathbf{P}_c$ is a triangle on the surface mesh (shown in Figure), the viewing angle α_i between each texture image and triangle is calculated as:

$$\alpha_i = \overrightarrow{\mathbf{P}\mathbf{V}_i} \cdot \hat{n} \tag{7.6}$$

where $\hat{n}$ is the normal vector of triangle $\triangle \mathbf{P}_a\mathbf{P}_b\mathbf{P}_c$:

$$\hat{n} = \overrightarrow{\mathbf{P}_c\mathbf{P}_a} \cdot \overrightarrow{\mathbf{P}_c\mathbf{P}_b} \tag{7.7}$$

and $\mathbf{V}_i$ is camera center, $\mathbf{P}$ is the center of triangle $\triangle \mathbf{P}_a\mathbf{P}_b\mathbf{P}_c$:

$$\mathbf{P} = \frac{\mathbf{P}_a + \mathbf{P}_b + \mathbf{P}_c}{3} \tag{7.8}$$

and the image patch t_i with maximal α_i is chosen to wrap onto $\triangle \mathbf{P}_a\mathbf{P}_b\mathbf{P}_c$ in texture mapping.

With OpenGL, texture mapping can be easily implemented. It automatically stretches and rotates the 2D image to fit the 3D shape. With hardware support, this expensive computation can be completed in a short time. However, due to the inherent differences between cameras (such as noise level, color tone, brightness),

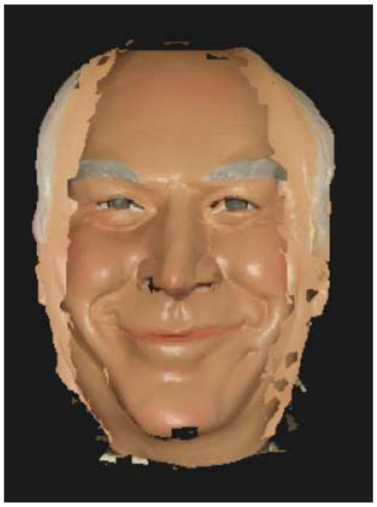

Figure 7.9: Direct Texture Mapping

directly stitching different images onto single 3D model would present serious blocking artifacts (as shown in Figure 7.9).

7.3.2 Texture Blending

To generate a photo-realistic 3D model, it is necessary to normalize the color of different images before texture mapping. Assume $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, ..., $\mathbf{P}_n$ are vertices on a surface mesh that lie between different texture sources. As shown in Figure 7.10, the triangles on the left use image I_a for texture mapping and the ones on the right use image I_b. Ideally, the corresponding points of $\mathbf{P}_i$ in two images $\mathbf{p}_{ai}$ and $\mathbf{p}_{bi}$ should have same colors, while in practice, they often present different RGB values. To normalize the color tone between them, we have to first establish the correspondence between two color spaces.

First, we construct three 256×256 lookup tables for three color channels respectively. The x-direction entries indicate color values in Image I_a and y-direction

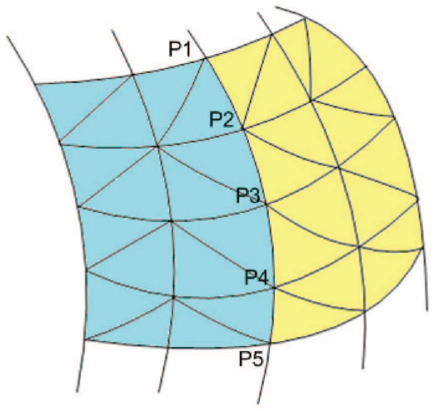

Figure 7.10: Texture Stitching

entries represent those in Image I_b. Each color pair (R_{ai}, R_{bi}), (G_{ai}, G_{bi}) and (B_{ai}, B_{bi}) has one vote at the corresponding entry. By selecting the entries getting the most votes along vertical direction, we can establish a one-to-one mapping between $[R_a, G_a, B_a] \rightarrow [R_b, G_b, B_b]$. Due to limited color distribution of the adjacent vertices $\mathbf{P}_i$ and noise disturbance, the mapping does not cover the whole color range from 0 to 255, and it is not smooth either. To establish a full smooth mapping, we can approximate it using a quadratic function:

$$R_b = k_{1r}{R_a}^2 + k_{2r}R_a \tag{7.9}$$

$$G_b = k_{1g}{G_a}^2 + k_{2g}G_a \tag{7.10}$$

$$B_b = k_{1b}{B_a}^2 + k_{2b}B_a \tag{7.11}$$

We omit the constant in quadratic equation because zero value from one color space should be mapped to zero in another space. If some entries are mapped to values larger than 255, they should be cropped to 255. Usually, the valid texture area in one image do not appear to be extremely bright or extremely dark, so the above crop usually do not cause problems in practice. The coefficients of quadratic equation k_{1r} and k_{2r}can be estimated through polynomial fitting of the selected color pairs (R_{ai}, R_{bi}). The same procedures are applied to (G_{ai}, G_{bi}) and (B_{ai}, B_{bi}) to obtain k_{1g}, k_{2g}, k_{1b} and k_{2b}. Figure 7.11 illustrates the quadratic fitting results.

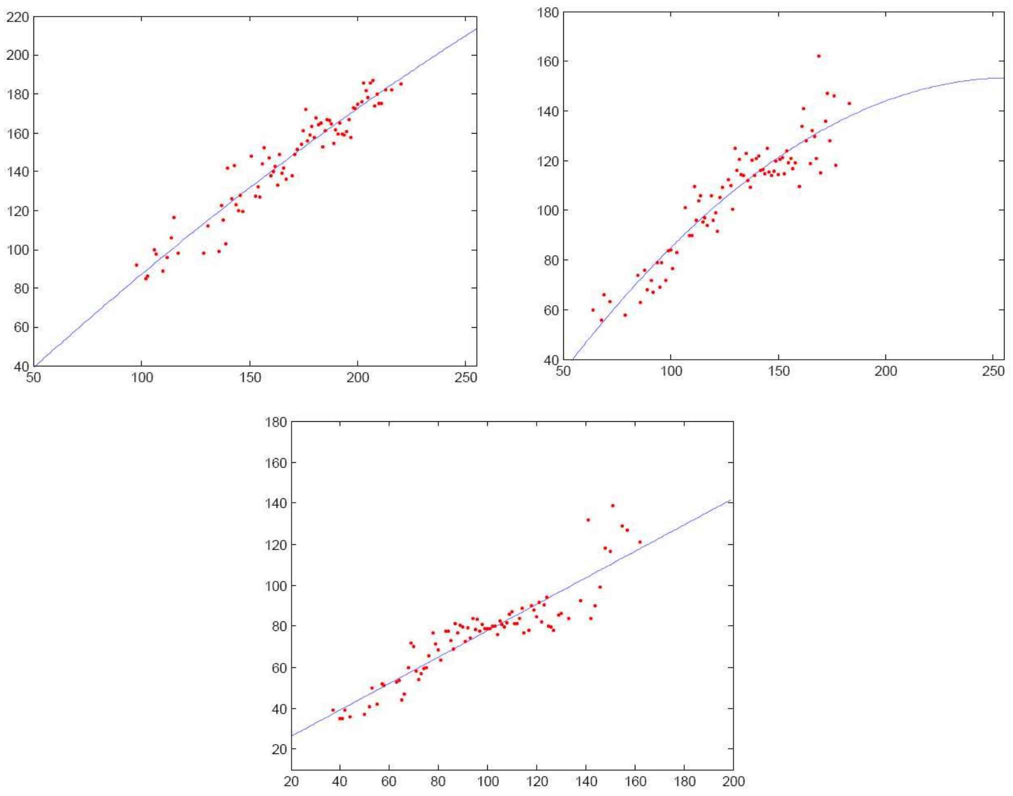

Figure 7.11: Mapping between Color Space (R, G, B Channel) of Two Images

We can then apply the mapping to image I_a to correct its color tone. After calculating the new value for each pixel using equations (7.9-7.11), we can obtain a color-normalized new image. For our system, we correct the color tone of left and right texture image with respect to center image respectively. Figure 7.12 shows the normalized color images.

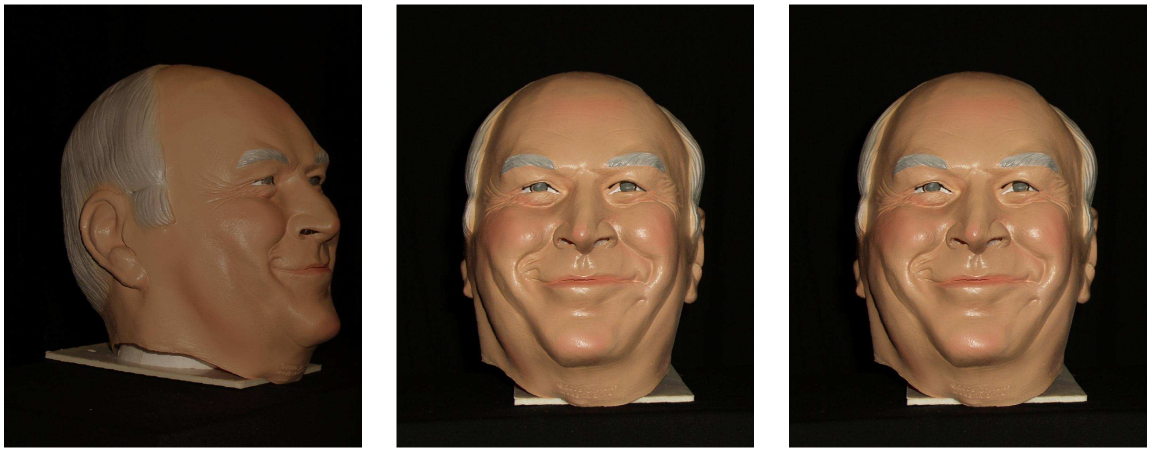
Figure 7.12: Normalized Texture Images

The global color normalization scheme targets at correcting the overall color tone of the whole image. Its underlying assumption is that the difference comes from each camera's unique internal parameters. In practice, viewing angles, reflectance and lighting condition also play important roles in the appearance of object in texture images. After global normalization, there still presents some discontinuities along the edges between different texture sources. We can smoothen the discontinuities through blurring techniques in the local neighborhood.

As described before, $\mathbf{p}_{ai}$ and $\mathbf{p}_{bi}$ $(i = 1, 2, \cdots, n)$ are two groups of image points corresponding to the same vertices on object surface. The discontinuities on the 3D textured model is mainly due to the color differences along two linked edge groups: $\mathbf{p}_{ai}\mathbf{p}_{a(i+1)}$ and $\mathbf{p}_{bi}\mathbf{p}_{b(i+1)}$. Using linear interpolation, we can replace the color of all the pixels on $\mathbf{p}_{ai}\mathbf{p}_{a(i+1)}$ with their counterparts on $\mathbf{p}_{bi}\mathbf{p}_{b(i+1)}$ $(i = 1, 2, \cdots, n)$. We then dilate the single pixel edge to a k pixel width transition band. For every pixel inside the transition band, we blend its original pixel values and the new values through a distance weighted function:

$$\mathbf{C}_j = \frac{(k - d_{ji})\mathbf{C}_{oj} + d_{ji}\mathbf{C}_{nj}}{k} \tag{7.12}$$

where $\mathbf{C}_j$ is the blended color of pixel $\mathbf{p}_j$, d_{ji} is the horizontal distance from pixel $\mathbf{p}_j$ to edge pixel $\mathbf{i}$, $\mathbf{C}_{nj}$ is the new color dilated from edge, and $\mathbf{C}_{oj}$ is $\mathbf{p}_j$'s original RGB value.

Such blending scheme can blur the edge discontinuities on the surface and also keep certain texture details from the original image. Similar to global normalization, we process the left and right texture image with respect to the center image. Figure 7.13 shows an close-up view of the local edge smoothing effects.

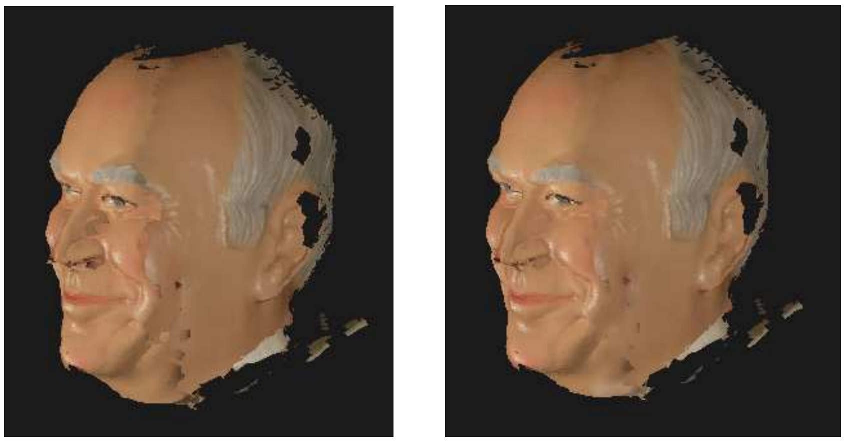

Figure 7.13: Texture Mapping Result Before and After Local Edge Smoothing

The final result of 3D photo-realistic modeling is shown in Figure 7.14. From three pattern stereo images and texture stereo images, we build a 3D model of the object through stereo matching, registration, integration and texture mapping. The multi-stereo camera is calibrated and fixed before image capturing. There is much room for further improvement, but overall, the result is satisfactory.

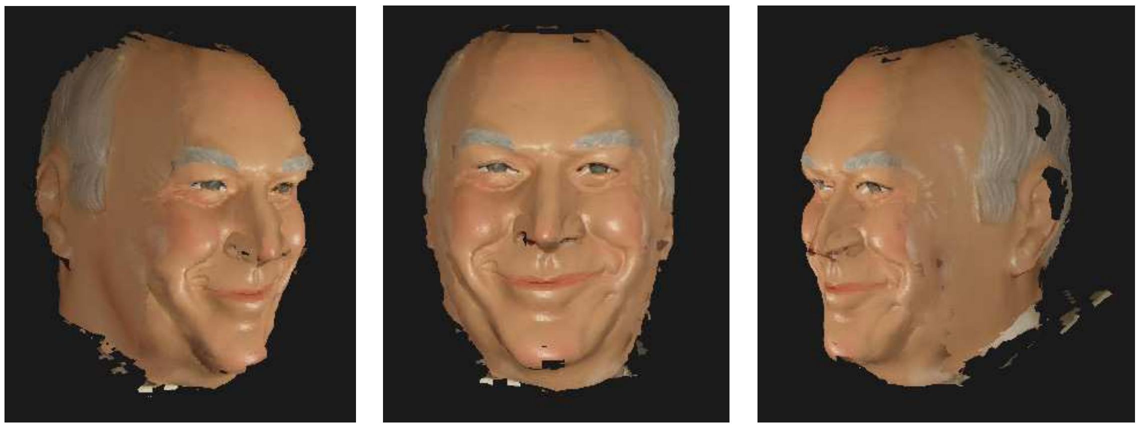
Figure 7.14: Final Textured 3D Model

7.4 Summary

In this Chapter, we focused on the 3D modeling techniques in Computer Graphics literature. We first introduced a volumetric method to extract the iso-surface from several layered point clouds. It computes a signed distance from every voxel to each range surface generated from different point clouds. Then a weighted average of these distances is computed where the weights depend on the viewing angle. We then use Marching Cube algorithm to generate a triangle mesh at voxels with zero signed distances. To render a photo-realistic 3D model on computer screen, we have to wrap the texture image onto the wire-frame model. From among three different texture images, we choose the one with the best viewing direction for texture mapping. To avoid the blocking artifacts in texture stitching, global color normalization followed by local edge smoothing is performed. The final rendered result is satisfactory.

Chapter 8

Conclusions

8.1 Summary

In this dissertation, we presented some novel techniques for image-based 3D sensing and modeling.

First we presented a Depth-from-Defocus (DFD) technique used in camera autofocusing and 3D shape recovery. The camera captures two images of the object with different preset lens positions. From different levels of blur, we retrieve the depth information of object and thus obtain the suitable lens setting for autofocusing. We extended this method to cameras operating in macro-mode, where image magnification can not be neglected. We developed a magnification normalization method to establish the correct correspondence between different images. We also capture one more image to improve the autofocusing accuracy if the object is placed in a certain sensitive distance range. Experimental results on a real SLR camera system are presented to demonstrate the effectiveness and robustness of our technique. In

Chapter 3, we extended our DFD technique to shift-variant cases. The convolution model we adopt in DFD assumes the camera to be a linear shift-invariant system, which is only valid when the object is planar and parallel to focal plane. Our new Shape from Shift-variant Blurring technique relaxes the previous assumption. It models the actual imaging process through a local integral equation and inverts the process to recover the shape parameters, not only the depth but also the slopes in horizontal and vertical directions. We carried-out the experiments on both simulation data and real camera system. Preliminary results on objects with simple shape indicate that our novel approach is useful in practical applications.

We presented another approach to 3D sensing techniques- stereopsis. Compared with single-viewing setting (such as DFD), stereopsis retrieves the depth information from two views. We introduced a novel stereo camera designed recently by some researchers which is able to capture a pair of stereo images using a single lens and a single sensor. With two sets of parallel mirrors in the stereo adaptor attached in front of the lens, the camera splits the incoming lights as if they were captured by different cameras. It is also programmed to shoot two images consecutively in a short time. During the first shooting, a pattern projector is turned on to project a random dot pattern onto the object surface; and the second image is an ordinary textured image. Chapter 4 presents a new camera calibration algorithm tailored for this special architecture. We utilize the physical constraint that the stereo pair has the same intrinsic parameters and derive a closed-form solution. We then refine the initial solution by minimizing the reprojected errors iteratively. Corner detection and pre-lens distortion techniques are also included to improve the accuracy. After estimating all camera parameters, the depth map of object can be calculated through triangulation. In Chapter 5, we presented a fast stereo matching technique based on Gaussian pyramid structure. The correspondence search is carried out in multi-resolution levels to decrease the number of comparisons. We also limit the search window size based on practical system setting. Simple image warping and segmentation techniques are applied to further improve the speed.

Finally, we describe our multi-stereo camera system, where three stereo cameras are placed about 1.3 m away from the object and 45 degrees apart from each other. Each camera can reconstruct partial shape of the object. In Chapter 6, we present a new registration technique to bring different point clouds into a common coordinate system. Similar to traditional Iterative Closest Point (ICP) algorithm, our method also computes the optimal transformation matrices between two corresponding point sets in a least square sense. When identifying the correspondence, we propose a new Iterative Closest Surface Patch (ICSP) approach which tries to find a pair of matched surface patches instead of points. Such a scheme is found to be more robust and faster than the standard technique. After registration, we have to build a single 3D model from the layered point clouds. In Chapter 7, we first introduce a volumetric method to extract the iso-surface of object at voxels with zero signed distances. The Marching Cube algorithm is implemented to generate a triangle mesh. The final texture mapping step includes view selection and texture blending. Among three texture images, the one with best viewing direction is chosen as the "wrap paper". Direct stitching of different images onto a single model would produce serious blocking artifacts. We use a global color normalization method followed by local edge smoothing to generate a photo-realistic model.

8.2 Future Work

This research can be extended and improved further in the future. Some possible topics are listed below:

Image Restoration

Image restoration (or soft focusing) is the process to deblur the image through post-processing techniques. In autofocusing, we can acquire a new focused image with lens system being set at correct positions. If the camera system is not under full control or the image is blurred because of object movement, hard focusing is not applicable. From our theoretical framework, the focused image can be computed from blurred images through deconvolution techniques. More elaborate implementation methods need to be developed to improve the restoration results.

For shift-variant cases, image restoration is even more important. For imaging systems with limited depth of field, such as microscope or bar-code scanner, there usually exists serious shift-variant blurring across the whole image. Moving lens may just focus at another part of the object. Our new theory on inverting the shift-variant blurring can be applied to image restoration and the experimental results are very promising. With advanced post-processing techniques, low-end lens systems like cell phone camera or webcam can replace the high-end imaging equipments to acquired high quality images.

Depth Recovery

In this dissertation, we recover the depth map through a single approach, such as Depth from Defocus (DFD) or stereopsis. In some applications, different techniques can be combined together to achieve better results. In endoscopy, a video sequence is acquired where the images are seriously blurred and distorted. Applying DFD or Structure from Motion (SFM) only may not achieve accurate results, as SFM usually requires accurate feature matching and DFD assumes images are perfectly aligned. We can apply SFM first to obtain an initial estimation and then apply shift-variant image restoration techniques to improve image sharpness. They can be applied alternatively or together in some iterative process to improve the accuracy of depth map.

Auto-registration

Our current registration approach requires the calibration pattern being placed facing both cameras to calculate the initial transformation matrices. Though it is performed only once in the beginning, such manual procedure is tedious. Instead of using a specific calibration pattern, coarse registration can be done by observing images of the object acquired by two cameras. We can first detect common features (such as corners of eyes and lips) in different images and recover relative camera positions through epipolar geometry. In refinement step, both 2D texture features and 3D surface features can be employed to identify correspondences. This should improve the accuracy of both geometric and photometric alignment.

Texture Blending

The appearance of our textured 3D model is satisfactory but the color discontinuities are still visible. To further improve the result, several approaches can be explored. In global color normalization, we can establish the mapping in other kinds of color space (such as Lab) instead of RGB space. RGB color model is device dependent and does not depict the natural color perception process. Correcting each channel separately may introduce some color distortion due to its non-orthogonality. The current view-selection scheme can also be improved. As it selects texture patch based on viewing direction, it may split texture images at specular highlighted or fine detailed areas. This makes seamless texture mapping very difficult. Instead, We can split the image at texture-less areas where enforced color transition looks more natural.

Bibliography

[1] Zhengyou Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[2] Roger Y Tsai, “A versatile camera calibration technique for high-accuracy 3d machine metrology using off-the-shelf tv cameras and lenses,” *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 4, pp. 323–344, 1987.

[3] R.I. Hartley, “An algorithm for self calibration from several views,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1994, pp. 908–912.

[4] Q.-T Luong and O. Faugeras, “Self-calibration of a moving camera from point correspondences and fundamental matrices,” *International Journal of Computer Vision*, vol. 32, no. 3, pp. 261–289, March 1997.

[5] Richard Hartley and Andrew Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2003.

[6] Frdric Devernay and Olivier Faugeras, “Straight lines have to be straight,” *Machine Vision and Applications*, vol. 13, no. 1, pp. 14–24, Aug. 2001.

[7] B. Prescott and G. F. McLean, "Line-based correction of radial lens distortion," *Graphical Models and Image Processing*, vol. 59, no. 1, pp. 39–47, Jan. 1997.

[8] M. Ahmed and A. Farag, "Nonmetric calibration of camera lens distortion: differential methods and robust estimation," *IEEE Transactions on Image Processing*, vol. 14, no. 8, pp. 1215–1230, Aug. 2005.

[9] A.W. Fitzgibbon, "Simultaneous linear estimation of multiple view geometry and lens distortion," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001, vol. 1, pp. 125–132.

[10] Richard I. Hartley and Sing Bing Kang, "Parameter-free radial distortion correction with centre of distortion estimation," in *IEEE International Conference on Computer Vision*, 2005, vol. 2, pp. 1834 – 1841.

[11] R.I. Hartley, "In defence of the 8-point algorithm," in *IEEE International Conference on Computer Vision*, Jun 1995, pp. 1064–1070.

[12] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3d model acquisition," *ACM Trans. on Graphics (SIGGRAPH 2002)*, 2002.

[13] Li Zhang, Brian Curless, and Steven M. Seitz, "Rapid shape acquisition using color structured light and multi-pass dynamic programming," in *The 1st IEEE International Symposium on 3D Data Processing, Visualization, and Transmission*, June 2002, pp. 24–36.

[14] Pei-sen Huang Song Zhang, "High-resolution, real-time three-dimensional shape measurement," *Optical Engineering*, vol. 45, no. 12, 2006.

[15] A. Fusiello, V. Roberto, and E. Trucco, "Efficient stereo with multiple windowing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, Puerto Rico, June 1997, pp. 858–863, IEEE Computer Society Press.

[16] R. C. Gonzales and R. E. Woods, *Digital Image Processing*, Addison Wesly, 1992.

[17] R.C. Bolles, H.H. Baker, and M.J. Hannah, "The jisct stereo evaluation," *IUW*, vol. 93, pp. 263–274.

[18] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," Tech. Rep., Cambridge, MA, USA, 1976.

[19] J Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[20] Ramin Zabih and John Woodfill, "Non-parametric local transforms for computing visual correspondence," in *ECCV '94: Proceedings of the Third European Conference-Volume II on Computer Vision*, London, UK, 1994, pp. 151–158, Springer-Verlag.

[21] D. Scharstein, "Matching images by comparing their gradient fields," in *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision and Image Processing., Proceedings of the 12th IAPR International Conference on*, Oct 1994, vol. 1, pp. 572–575 vol.1.

[22] Stephen T. Barnard, "Stochastic stereo matching over scale," *International Journal of Computer Vision*, vol. 3, no. 1, pp. 17032, May 1989.

[23] Daniel Scharstein and Richard Szeliski, "Stereo matching with non-linear diffusion," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 0, pp. 343, 1996.

[24] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 11, pp. 1222–1239, Nov 2001.

[25] C. Lawrence Zitnick and Takeo Kanade, "A cooperative algorithm for stereo matching and occlusion detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 7, pp. 675–684, 2000.

[26] N. Ahuja J. Weng and T. S. Huang, "Matching two perspective views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 806–825, Aug. 1992.

[27] Peter J. Burt and Edward H. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. COM-31,4, pp. 532–540, 1983.

[28] James R. Bergen, P. Anandan, Keith J. Hanna, and Rajesh Hingorani, "Hierarchical model-based motion estimation," in *ECCV '92: Proceedings of the Second European Conference on Computer Vision*, London, UK, 1992, pp. 237–252, Springer-Verlag.

[29] Kenneth Wong and Roberto Cipolla, "Structure and motion from silhouettes," in *IEEE International Conference on Computer Vision*.

[30] David A. Forsyth and Jean Ponce, *Computer Vision A Modern Approach*, Prentice Hall, 2002.

[31] Q. Zheng and R. Chellappa, "Estimation of illuminant direction, albedo, and shape from shading," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, Jun 1991, pp. 540–545.

[32] O.E. Vega and Y.H. Yang, "Shading logic: a heuristic approach to recover shape from shading," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 6, pp. 592–597, Jun 1993.

[33] K.M. Lee and C.-C.J. Kuo, "Shape from shading with a linear triangular element surface model," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 8, pp. 815–822, Aug 1993.

[34] M. Bichsel and A.P. Pentland, "A simple algorithm for shape from shading," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, Jun 1992, pp. 459–465.

[35] P. Dupuis and J. Oliensis, "Direct method for reconstructing shape from shading," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, Jun 1992, pp. 453–458.

[36] A. P. Pentland, "Local shading analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 6, pp. 170–187, 1984.

[37] Chia-Hoang Lee and Azriel Rosenfeld, “Improved methods of estimating shape from shading using the light source coordinate system,” pp. 323–347, 1989.

[38] A. Pentland, “Shape information from shading: A theory about human perception,” in *Computer Vision., Second International Conference on*, Dec 1988, pp. 404–413.

[39] Ping sing Tsai and Mubarak Shah, “Shape from shading using linear approximation,” *Image and Vision Computing*, vol. 12, no. 8, pp. 487–498, 1994.

[40] M. Subbarao, T. Chio, and A. Nikzad, “Focusing techniques,” *Optical Engineering*, vol. 32(11), pp. 2824–2836, 1993.

[41] S.K. Nayar and Y. Nakagawa, “Shape from focus,” *PAMI*, vol. 16, no. 8, pp. 824–831, August 1994.

[42] Murali Subbarao and Jenn-Kwei Tyan, “Selecting the optimal focus measure for autofocusing and depth-from-focus,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 864–870, 1998.

[43] Y. Xiong and S.A. Shafer, “Depth from focusing and defocusing,” in *DARPA93*, 1993, pp. 967–.

[44] A. P. Pentland, “A new sense for depth of field,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 4, pp. 523–531, 1987.

[45] M. Subbarao and N. Gurumoorthy, "Depth recovery from blurred edges," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1988, pp. 498–503.

[46] M. Subbarao and T.-C. Wei, "Depth from defocus and rapid autofocusing: a practical approach," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, Jun 1992, pp. 773–776.

[47] J. Ens and P. Lawrence, "An investigation of methods for determining depth from focus," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 2, pp. 97–108, Feb 1993.

[48] M. Watanabe and S. Nayar, "Rational filters for passive depth from defocus," Tech. Rep. CUCS-035-95, Dept. of Computer Science, Columbia University, Sept. 1995.

[49] S.K. Nayar, M. Watanabe, and M. Noguchi, "Real-time focus range sensor," in *Computer Vision, 1995. Proceedings., Fifth International Conference on*, Jun 1995, pp. 995–1001.

[50] G. Surya and M. Subbarao, "Depth from defocus by changing camera aperture: a spatial domain approach," in *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on*, Jun 1993, pp. 61–67.

[51] M. Subbarao and G. Surya, "Depth from defocus: A spatial domain approach," *IJCV*, vol. 13, no. 3, pp. 271–294, December 1994.

[52] Tao Xian and Murali Subbarao, “Performance evaluation of different depth from defocus (dfd) techniques,” in *Proceedings of SPIE, Two- and Three-Dimensional Methods for Inspection and Metrology III*, 2005, vol. 6000.

[53] P.Favaro and S. Soatto, “A geometric approach to shape from defocus,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 406–417, 2005.

[54] P. Favaro, A. Mennucci, and S. Soatto, “Observing shape from defocused images,” *International Journal of Computer Vision*, vol. 52, no. 1, pp. 25–43, 2003.

[55] Djemel Ziou and Francois Deschenes, “Depth from defocus estimation in spatial domain,” *Computer Vision and Image Understanding: CVIU*, vol. 81, no. 2, pp. 143–165, 2001.

[56] A. N. Rajagopalan and S. Chaudhuri, “Optimal recovery of depth from defocused images using an MRF model,” in *ICCV*, 1998, pp. 1047–1052.

[57] A. N. Rajagopalan and Subbasis Chaudhuri, “A recursive algorithm for maximum likelihood-based identification of blur from multiple observations,” *IEEE Transactions on Image Processing*, vol. 7, pp. 1075–1079, 1998.

[58] M. Subbarao, “Spatial-domain convolution/deconvolution transform,” Tech. Rep. 91.07.03, Computer Vision Laboratory, Dept. of Electrical Engineering, State University of New York, 1991.

[59] Murali Subbarao, Tse-Chung Wei, and Gopal Surya, “Focused image recovery from two defocused images recorded with different camera settings,” *IEEE Transactions on Image Processing*, vol. 4, no. 12, pp. 1613–1628, 1995.

[60] D. Ziou, “Passive depth from defocus using a spatial domain approach,” in *ICCV ’98: Proceedings of the Sixth International Conference on Computer Vision*, Washington, DC, USA, 1998, p. 799, IEEE Computer Society.

[61] F. Deschênes, D. Ziou, and P. Fuchs, “A homotopy-based approach for computing defocus blur and affine transform simultaneously,” *Pattern Recogn.*, vol. 41, no. 7, pp. 2263–2282, 2008.

[62] M. Subbarao, U.S. Patent Application No. 11/235724, Filed on 9/26/05, U.S. Patent Application No. 11/242191, Filed on 10/03/2005, and U.S. Patent Application No. 11/450024, 6/10/2006. *Rao Transforms...* , U.S. Copyright No. TX 6-195-821, June 1, 2005. See http://www.integralresearch.net.

[63] M. Subbarao, Y. Kang, S. Dutta, and X. Tu, “Localized and computationally efficient approach to shift-variant image deblurring,” in *IEEE International Conference on Image Processing*, 2008.

[64] Xue Tu, M. Subbarao, and Youn-Sik Kang, “A new approach to 3d shape recovery of local planar surface patches from shift-variant blurred images,” in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, Dec. 2008, pp. 1–5.

[65] Paolo Favaro, Stefano Soatto, Martin Burger, and Stanley Osher, "Shape from defocus via diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 518 – 531, May 2008.

[66] Vinay P. Namboodiri, Subhasis Chaudhuri, and Sunil Hadap, "Regularized depth from defocus," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, Oct. 2008, pp. 1520–1523.

[67] K. Pulli, "Multiview registration for large data sets," in *3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on*, 1999, pp. 160–168.

[68] F. Werner T. Fitzgibbon A. Zisserman, A. Schaffalitzky, "Automated reconstruction from multiple photographs," in *IEEE International Conference on Image Processing*, 2002, vol. 3, pp. 517–520.

[69] Paul J. Besl and Neil D. Mckay, "A method for registration of 3d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, Feburary 1992.

[70] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992.

[71] Zhengyou Zhang, "Iterative point matching for registration of free-form curves and surfaces," *Int. J. Comput. Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[72] Robert Bergevin, Marc Soucy, Herv Gagnon, and Denis Laurendeau, "Towards a general multi-view registration technique," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 5, pp. 540–547, 1996.

[73] R. Benjemaa and F. Schmitt, "Fast global registration of 3d sampled surfaces using a multi-z-buffer technique," *3D Digital Imaging and Modeling, International Conference on*, vol. 0, pp. 113, 1997.

[74] G. Blais and M.D. Levine, "Registering multiview range data to create 3d computer objects," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 17, no. 8, pp. 820–824, Aug 1995.

[75] O D Faugeras and M Hebert, "The representation, recognition, and locating of 3-d objects," *Int. J. Rob. Res.*, vol. 5, no. 3, pp. 27–52, 1986.

[76] Berthold K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[77] Andrew Johnson and Sing Bing Kang, "Registration and integration of textured 3-d data," in *Image and Vision Computing*, 1996, pp. 234–241.

[78] Fausto Bernardini, Ioana M. Martin, and Holly Rushmeier, "High-quality texture reconstruction from multiple scans," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 4, pp. 318–332, 2001.

[79] Brian Curless and Marc Levoy, "A volumetric method for building complex models from range images," *Computer Graphics*, vol. 30, no. Annual Conference Series, pp. 303–312, 1996.

[80] A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt, "Reliable surface reconstruction from multiple range images," *Lecture Notes in Computer Science*, vol. 1064, pp. 117–??, 1996.

[81] Mark K. Wheeler, Yoichi Sato, and Katsushi Ikeuchi, "Consensus surfaces for modeling 3d objects from multiple range images," pp. 77–92, 2001.

[82] Jeremy S. de Bonet and Paul Viola, "Roxels: Responsibility weighted 3d volume reconstruction," 1999, vol. 1, p. 418.

[83] Andrew W. Fitzgibbon, Geoff Cross, and Andrew Zisserman, "Automatic 3d model construction for turn-table sequences," in *SMILE'98: Proceedings of the European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, London, UK, 1998, pp. 155–170, Springer-Verlag.

[84] Steven M. Seitz and Charles R. Dyer, "Photorealistic scene reconstruction by voxel coloring," in *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, Washington, DC, USA, 1997, p. 1067, IEEE Computer Society.

[85] H. Y. Lin and M. Subbarao, "Three-dimensional model acquisition using rotational stereo and image focus analysis," in *Proceedings of SPIE*, 2001, vol. 4189, pp. 201–210.

[86] Greg Turk and Marc Levoy, "Zippered polygon meshes from range images," in *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1994, pp. 311–318, ACM.

[87] M. Rutishauser, M. Stricker, and M. Trobina, "Merging range images of arbitrarily shaped objects," in *Computer Vision and Pattern Recognition, 1994.*

Proceedings CVPR '94., 1994 IEEE Computer Society Conference on, Jun 1994, pp. 573–580.

[88] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle, "Surface reconstruction from unorganized points," *Computer Graphics*, vol. 26, no. 2, pp. 71–78, 1992.

[89] Soon-Yong Park and M. Subbarao, "Automatic 3d model reconstruction using voxel coding and pose integration," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, 2002, vol. 2, pp. 533–536.

[90] K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro, and W. Stuetzle, "Robust meshes from multiple range maps," in *NRC '97: Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, Washington, DC, USA, 1997, p. 205, IEEE Computer Society.

[91] P.W. Rander, P.J. Narayanan, and T. Kanade, "Recovery of dynamic scene structure from multiple image sequences," in *Multisensor Fusion and Integration for Intelligent Systems, 1996. IEEE/SICE/RSJ International Conference on*, Dec 1996, pp. 305–312.

[92] Sundar Vedula, Peter Rander, Peter R, Hideo Saito, and Takeo Kanade, "Modeling, combining, and rendering dynamic real-world events from image sequences," in *4th conference virtual systems and multimedia*, 1998, vol. 1, pp. 326–332.

[93] Kari Pulli, Michael Cohen, Tom Duchamp, Hugues Hoppe, Linda Shapiro, and Werner Stuetzle, "View-based rendering: Visualizing real objects from scanned

range and color data," in *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, Julie Dorsey and Phillipp Slusallek, Eds., New York, NY, 1997, pp. 23–34, Springer Wien.

[94] Kenji Matsushita and Toyohisa Kaneko, "Efficient and handy texture mapping on 3D surfaces," in *Computer Graphics Forum (Eurographics '99)*, 1999, vol. 18(3), pp. 349–358.

[95] H. Lensch, W. Heidrich, and H. Seidel, "Automated texture registration and stitching for real world models," in *Proceedings of Pacific Graphics 2000*, pp. 317–327.

[96] Y. Alshawabkeh and N. Haala, "Automatic multi-image photo-texturing of complex 3d scenes," *XVIII CIPA Int. Symposium, Torino, 27 September*, vol. 1, pp. 68–73, 2005.

[97] Nobuyuki Bannai, Robert B. Fisher, and Alexander Agathos, "Multiple color texture map fusion for 3d models," *Pattern Recogn. Lett.*, vol. 28, no. 6, pp. 748–758, 2007.

[98] Adam Baumberg, "Blending images for texturing 3d models," in *Proc. Conf. on British Machine Vision Association*, 2002, pp. 404–413.

[99] Claudio Rocchini, Paolo Cignomi, Claudio Montani, and Roberto Scopigno, "Multiple textures stitching and blending on 3D objects," in *Eurographics Rendering Workshop*, 1999, pp. 119–130.

[100] Kun Zhou, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum, "Texturemontage," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1148–1155, 2005.

[101] M. Subbarao, "Efficient depth recovery through inverse optics," in *Machine Vision for Inspection and Measurement*, H. Freeman, Ed., pp. 101–126. Academic, New York, 1989.

[102] M. Subbarao, "Determining distance from defocused images of simple object," Tech. Rep. 89.07.20, omputer Vision Laboratory, Dept. of Electrical Engineering, State University of New York, 1989.

[103] Jean-Marc Lavest, Marc Viala, and Michel Dhome, "Do we really need an accurate calibration pattern to achieve a reliable camera calibration?," in *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, London, UK, 1998, pp. 158–174, Springer-Verlag.

[104] Richard H. Byrd, Mary E. Hribar, and Jorge Nocedal, "An interior point algorithm for large scale nonlinear programming," *SIAM Journal on Optimization*, vol. 9, pp. 877–900, 1997.

[105] Luca Lucchese and Sanjit Mitra, "Using saddle points for subpixel feature detection in camera calibration targets," in *Asia-Pacific Conferece on Curcuits and Systems*, 2002, vol. 2.

[106] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri, "A compact algorithm for rectification of stereo pairs," *Machine Vision and Applications*, vol. 12, no. 1, pp. 16–22, 2000.